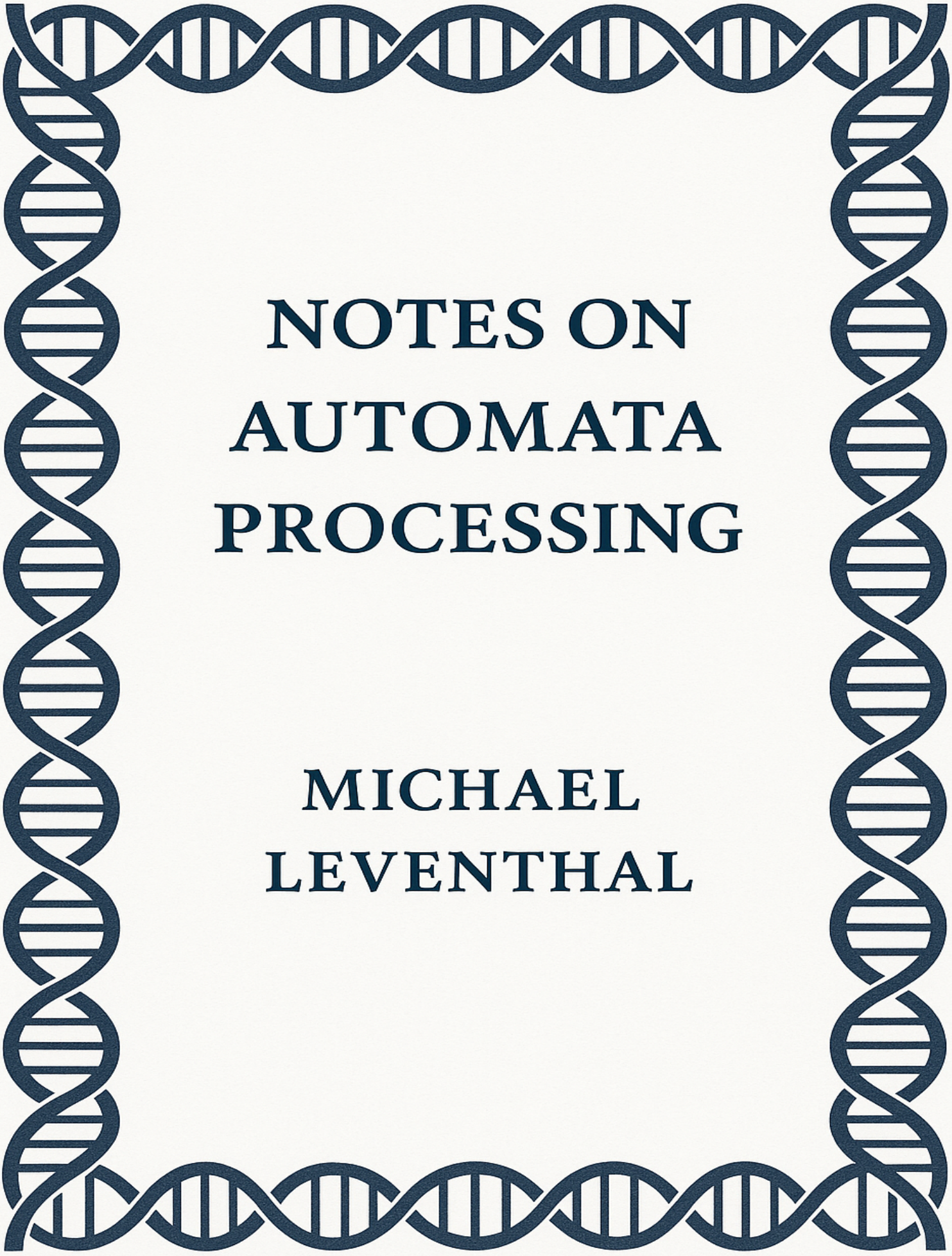


NOTES ON AUTOMATA PROCESSING

MICHAEL LEVENTHAL

Notes on Automata Processing

mleventhal@robotsmali.org, textscience@gmail.com

Notes on Automata Processing
Author's Preface

Between 2010 and 2014 I worked on a novel semiconductor architecture for Micron Corporation for which I conceived its computing paradigm, giving it the name Automata Processing. Micron adopted the name "Automata Processor" for the chip which was unveiled to the world at Supercomputing in 2013 and produced as engineering samples. I developed an XML-based hardware description language for the Automata Processor called the Automata Network Markup Language (ANML) and a visual representation of that language called ANML-G, using the established notation for state-transition diagrams as a starting point. ANML was translatable into Automata Processor machine code by a compiler developed by the Micron team and an ANML-G Workbench was implemented by a third-party developer under contract to Micron as an IDE. Micron discontinued development of the Automata Processor not long after its launch, having never commercialized the technology.

The Automata Processor, being implementable using DRAM process technology, has been associated with Processing-in-Memory (PIM) architectures, although it was conceived more as a workload-specific accelerator rather than a potential solution to the Von Neumann bottleneck. It was targeted at pattern-recognition processes such as virus detection and limited to the use of regular expressions for problem-specific configuration. It was a programmable hardware device, like the FPGA, its arrays of character-level pattern recognition devices being configurable and capable of arbitrary interconnection.

The configurability of the Automata Processor opened up the possibility of using it for a wider range of computing problems. It had other computing elements including combinatorial devices and, most significantly, counters. The counters made it Turing-complete, though not efficiently programmable for all types of problems. Still, there were many interesting problems that could be solved with the Automata Processor architecture once it was unshackled from regular expression languages. ANML gave one the ability to experiment with the potential capabilities of the fabric.

I began to imagine different machines that could be built. I jotted down notes for each machine in a Wiki, eventually assembling a tutorial on the Automata Processor elements, a Cookbook with my machines, and a discussion of practical points of running the machines on the actual hardware. Those notes were never made available to the public.

Despite the Automata Processor never being released as a commercial product, it seems there is still interest in its underlying ideas, with hundreds of research papers citing our work. To the best of my knowledge, there was not a comparable practical, direct implementation of a configurable fabric of finite state automata elements in a semiconductor device before the Automata Processor and there has not been one since. The few odd automata theory enthusiasts that enjoy designing task-specific Turing Machines or NFAs or PDAs may find some of my machines entertaining. I have heard of some interest in the quantum computing community for using NFAs to explore quantum parallel algorithms. Maybe there is a nugget or two in these notes that will inspire some useful idea in the future.

So, I reconstituted my notes as best as I was able - a few machines are missing and there are likely many inconsistencies in my notes but, on the whole, the work seems coherent enough for an interested person to get some value from them.

It would have been lovely to have been able to offer this content with the ANML-G Workbench for playing with the designs. Here are video captures that show the Workbench simulating the operation of ANML machines:

https://robotsmali.org/wp-content/uploads/2025/11/ANMLG_Workbench1.mp4

https://robotsmali.org/wp-content/uploads/2025/11/ANMLG_Workbench2.mp4

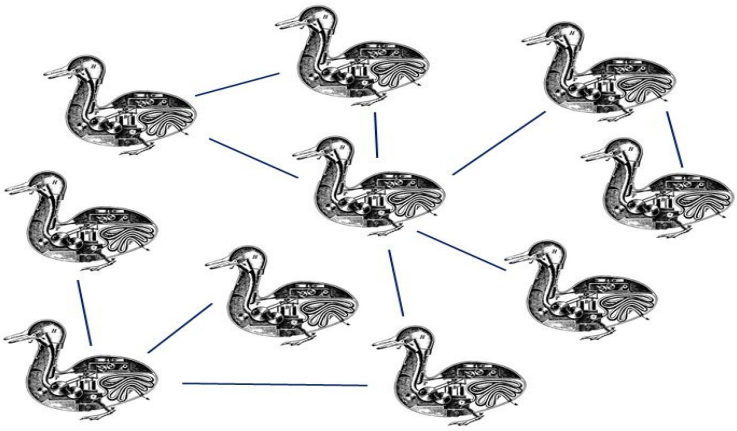
The foundational paper on the Automata Processor, written by me and other members of the team at Micron is here:

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6719386

Michael Leventhal
1 November 2025
Bamako, Mali

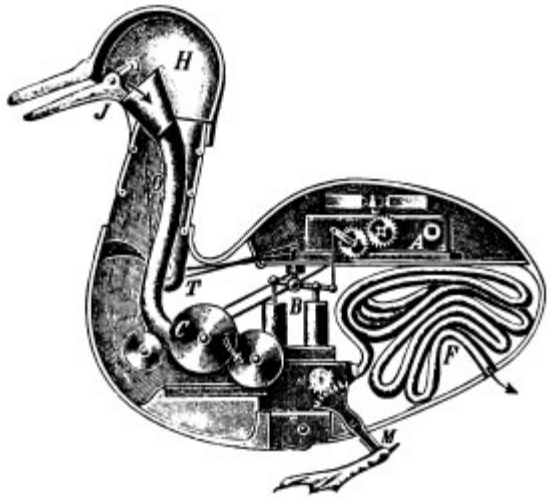
ANML (Automata Network Markup Language) User Guide

ANML (Automata Network Markup Language) / ANML-G User's Guide (ANML-Graphic)



ANML Icon

ANML Icon



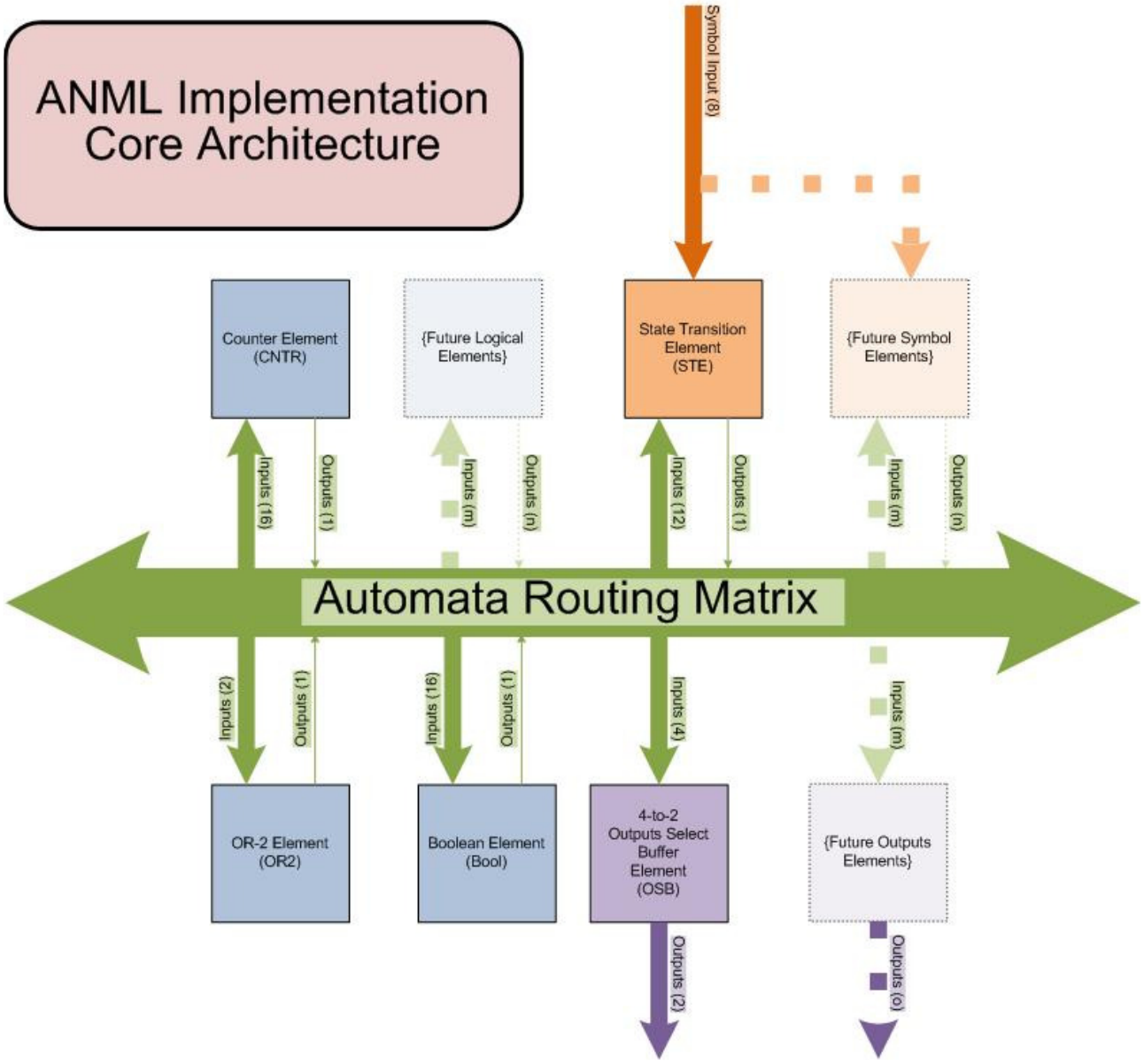
The ANML Icon is taken from the Wikipedia article on Automaton. "The *Digesting Duck* by Jacques de Vaucanson, hailed in 1739 as the first automaton capable of digestion."

"Sans...le canard de Vaucanson vous n'auriez rien qui fit ressouvenir de la gloire de la France."
"Without ... Vaucanson's duck you would have nothing to remind you of the glory of France."

Voltaire

Automata in ANML

Automata in ANML

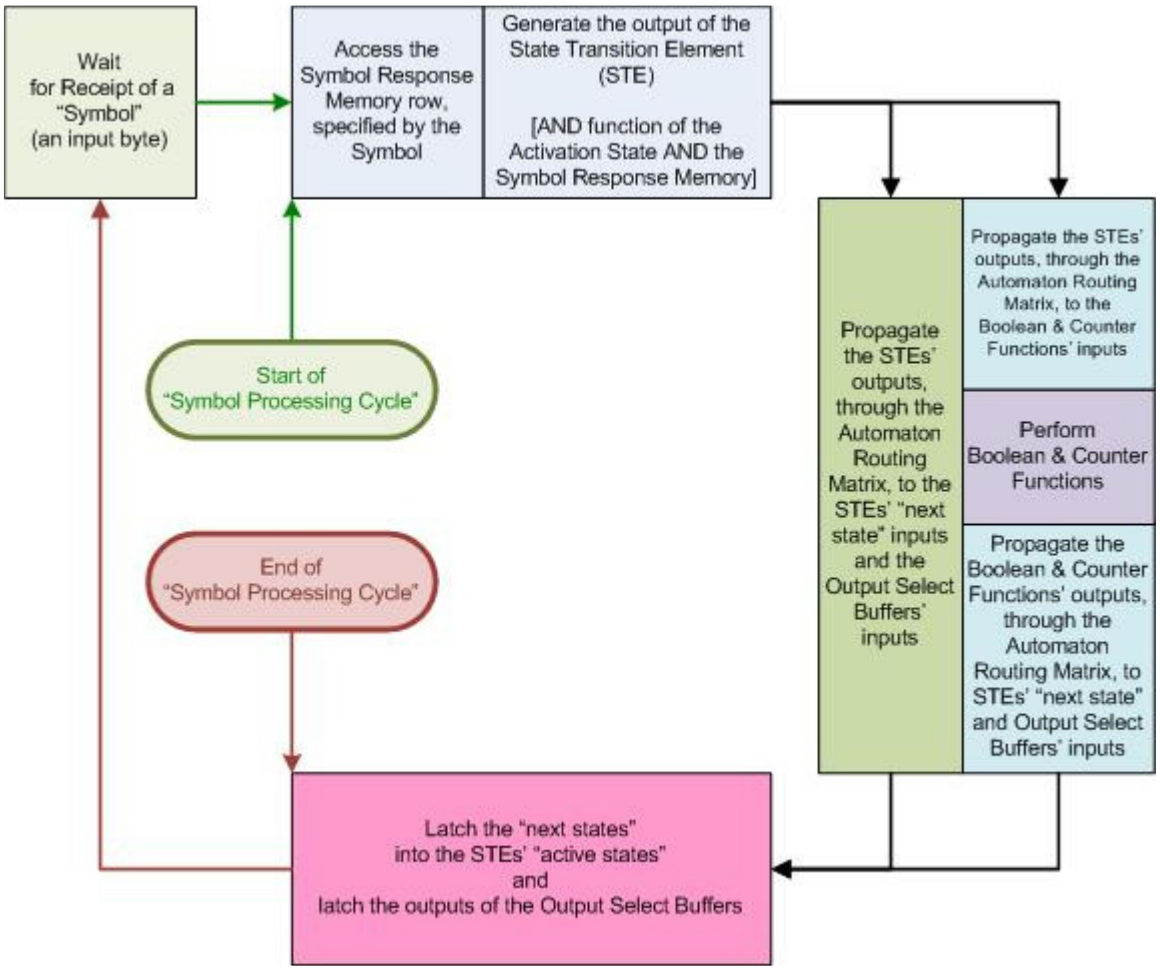


Symbol Cycle

Symbol Cycle

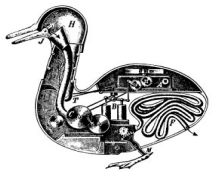
“Symbol Cycle” versus “Syncopated Cycle”

The Automata Processor has a fixed “symbol cycle”, at which it consumes input symbols. It is easy to think of this rate as the “clock” or “beat-rate” of the machine. It is also natural to assume that everything operates on that “clock”. ***It does not!*** There is a ***pipeline*** of events going on, during the time consumed by a “symbol cycle”. Refer to the following diagram:



This makes the distinction clear, between SRAs and the Counter/Boolean Automata – the “symbol consuming” and “results processing” parts of the AutomataProcessor. There is, to use a music term, a “beat” (the “symbol cycle”) and “syncopated beats” (pipeline steps) to this machine. Everything is synchronous – it happens on a fixed beat frequency – but things do not happen at the same time! This becomes important when trying to understand the true operation of counters and Booleans – as well as any future “library” elements.

State-Transition-Element



State-Transition-Element

The state-transition-element is the core feature of ANML as the sole device capable of receiving an input symbol. It also is required to drive the operation of the other elements in ANML such as counters and the combinatorial elements. There are two possible subelements of the state-transition-element specifying actions that may occur on a match event, report-on-match and activate-on-match. Only up to one report-on-match may be associated with a state-transition-element while zero or more activate-on-match may be specified. In the ANML XML coding the report-on-match element must precede any activate-on-match elements.

Syntax

	attribute	type/value	occurence	example
<state-transition-element			0 or more	
	id=	*id*	required unique in macro	"1"
	symbol-set=	*character* *character-class* *bits-enabled* *multiple*	required	"A" "[aA]" "{0:9,21}" "/ab+c/"
	case-insensitive	*boolean*	default: false	"true"
	start=	*"none"* *"start-of-data"* *"all-input"*	default: *none*	"none" "start-of-data" "all-input"
	latch=	*"true"* *"false"*	default: *false*	"true"
>				
<report-on-match/>			0 or 1	
<activate-on-match			0 or more	
	macro=	*"self"* *idref*	default: *self*	"self" "radial"
	element=	*idref*	required	"2"
/></state-transition-element>				

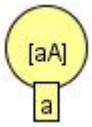
State Transition Element

A Simple Automaton
A Simple Automaton with Input and Output
A Larger Automaton
Self Activating Automaton

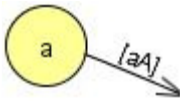
A Simple Automaton

A Simple Automaton

<state-transition-element id="a" symbol-set="[aA]"></state-transition-element>

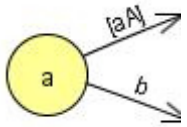


ANML state transition elements

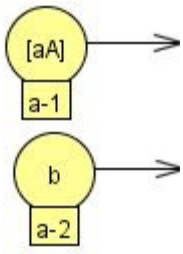


state labelled a with a transition function on the input of either 'a' or 'A'. In ANML this is a complete automaton but in classic finite automata it is not a complete finite state machine, most critically lacking a destination state for the transition function. In ANML we would describe this example as an state transition element labelled with the id "a" that recognizes the character class consisting of either 'a' or 'A'.

The second graph is not the preferred way to represent ANML state transition elements because it encourages a subtle miscomprehension that can lead to design errors. For example, the following state machine, readily created using classic notation, cannot be implemented as shown in ANML automata:



This shows a state that is exited by two different transition values, the character class [aA] and b. An ANML state transition element can only be programmed to recognize a single symbol set so we need one state transition element for each transition value. This is shown below:



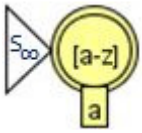
The ANML implementation of the state machine requires two state transition elements for the one state and also requires relabeling of the state identifier into labels associated with the state transition elements. The graph will also be different as whereever we transitioned to state a we will now need transitions to multiple ANML state transition elements. While state machine notation and ANML notation are, in this case, equivalent, the designers task will be simplier in the long run if the notation used is isomorphic its actual implementation in ANML.

This first simple automaton network is not terribly useful because its single state transition element receives no input and is not connected to other state transition elements, and generates no output. The next section will show how to specify a simple automaton network that can receive input and generates output.

A Simple Automaton with Input and Output

A Simple Automaton with Input and Output

<state-transition-element id="a" symbol-set="[a-z]" start="all-input"><report-on-match/></state-transition-element>



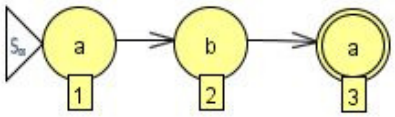
This machine accepts symbols from the input stream and reports when an input symbol matches the symbol-set. In classic finite state automata the automaton would be described as both a start state and an accept state. The triange pointing into the automaton is a standard symbol for designating a start state but ANML embeds the symbol S∞ in the triangle to designate it as an all-input start state transition element. ANML also supports start-of-data start which will be designated by modifying this symbol. An all-input start state transition element remains activated and receives and tests every input symbol against the symbol-set. An important difference between ANML and classic finite state automata is that any ANML state transition element may be designated as an all-input or start-of-data start transition element. In the formal definition of finite automata only a single state may be designated as the start state.

The graphic representation of the automaton also adapts the classic finite automata representation for accept states to indicate that the automaton will report when the input symbol matches the symbol-set. An important difference between ANML automata and common uses of finite automata such as string searching or RegEx is that an output-generating state transition element does not have to be a terminal state; any ANML element can report a match. There are many considerations in output generation and match processing; this will be covered in subsequent sections.

A Larger Automaton

A Larger Automaton

<state-transition-element id="1" symbol-set="a" start="all-input"><activate-on-match element="2"/></state-transition-element>
<state-transition-element id="2" symbol-set="b"><activate-on-match element="3"/></state-transition-element>
<state-transition-element id="3" symbol-set="a"><report-on-match/></state-transition-element>



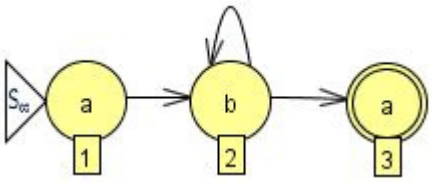
This automaton accepts symbols from the input in state transition element 1 and, on a match, activates state transition element 2. Activation means that the state transition element will accept the next input symbol and take whatever action is specified when the symbol matches the symbol-set. An activation by another state transition element is only enabled for the next input cycle; if no subsequent activation of the state transition element occurs it will be disactivated after processing the current input symbol. If state transition element 2 matches a 'b' state transition element 3 will be activated. If, on the next input cycle state transition element 3 sees an 'a' the match will be reported because reporting is enabled for this state transition element.

State transition element 1 is configured to receive all-input so it will be continuously activated. If this automaton were to be presented with the sequence 'aba' on the final 'a' report output signal would be generated by state transiton element 3 and the final 'a' would also be evaluated and matched by state transition element 1, causing state transition element 2 to be activated for the next input cycle. If the sequence continued with 'ba' a second output report would be generated by state transition element 3.

Self Activating Automaton

Self-Activating Automaton

<state-transition-element id="1" symbol-set="a" start="all-input"><activate-on-match element="2"/></state-transition-element>
<state-transition-element id="2" symbol-set="b">
<activate-on-match element="2"/><activate-on-match element="3"/>
</state-transition-element>
<state-transition-element id="3" symbol-set="a"><report-on-match/></state-transition-element>



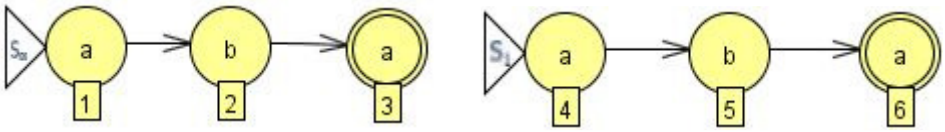
This automaton is identical to the previous one except for an additional activate-on-match signal in state transition element 2 which activates itself. Once state transition element 2 is activated by state transition element 1 it will continue to remain activated as long as the input symbol is a 'b'. Without the self-activation loop the previous automaton would only generate an output signal in state transition element 3 after seeing the sequence 'aba'. This automaton will report output on /ab+a/ where '+' means 'one or more', i.e., a followed by one or more b followed by a.

Input into a Network

Input into a Network

<state-transition-element id="1" symbol-set="a" start="all-input"><activate-on-match element="2"/></state-transition-element>
<state-transition-element id="2" symbol-set="b"><activate-on-match element="3"/></state-transition-element>
<state-transition-element id="3" symbol-set="a"><report-on-match/></state-transition-element>

<state-transition-element id="4" symbol-set="a" start="start-of-data"><activate-on-match element="5"/></state-transition-element>
<state-transition-element id="5" symbol-set="b"><activate-on-match element="6"/></state-transition-element>
<state-transition-element id="6" symbol-set="a"><report-on-match/></state-transition-element>



A symbol is presented to activated state transition elements on each input cycle. Any state transition element may be activated, either through configuration when the automaton network is created or through activation by another state transition element at runtime. When a state transition element is activated it receives the next input symbol and will evaluate it against the symbol-set it is programmed to recognize.

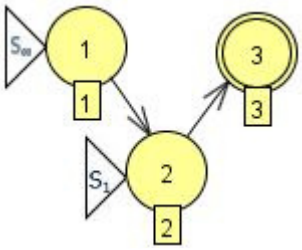
On configuration state transition elements may be designated to accept input under one of two conditions: all-input or start-of-data. All-input state transition elements are always activated and receive a symbol from the input source on each input cycle. Start-of-data state transition elements, like start states in classic finite automata, are activated only on the first symbol of a new data stream. The start of a new data stream condition is controlled at the application programming level.

In the example above the first state transition element of the first automaton receives all input symbols. The second automaton is identical except that the initial state transition element only is activated at start-of-data. If, at start-of-data, the input sequences "baba" is seen the first automaton will generate an output signal from state transition element 3 on the last symbol while the second automaton will not because state transition element 4 remain disactivated after failing to match the first symbol 'b'.

Multiple Input Automaton

Multiple Input Automaton

<state-transition-element id="1" symbol-set="1" start="all-input"><activate-on-match element="2"/></state-transition-element>
<state-transition-element id="2" symbol-set="2" start="start-of-data"><activate-on-match element="3"/></state-transition-element>
<state-transition-element id="3" symbol-set="3"><report-on-match/></state-transition-element>



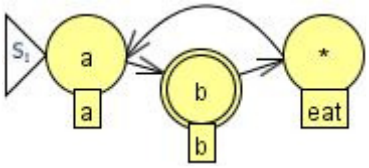
Any state transition element may be designated to respond to input from the input source, whether continuously or at start-of-data and a single automaton may have multiple input state transition elements. This is different from formal finite automata which only permits a single state to be designated as the start state.

In the automaton above state transition element 1 examines all input and state transition element 2 will be activated on the first input symbol at start-of-data and can also be activated by state transition element 1 when it recognizes a symbol. Any input stream containing '123' will generate an output signal from state transition element 3 and the sequence '23' where the 2 is seen at start-of-data will also generate an output report from state transition element 3.

Report Output Automaton

Report Output Automaton

<state-transition-element id="a" symbol-set="a" start="start-of-data"><activate-on-match element="b"/></state-transition-element>
<state-transition-element id="b" symbol-set="b"><report-on-match/><activate-on-match element="eat"/></state-transition-element>
<state-transition-element id="eat" symbol-set="*"><activate-on-match element="a"/></state-transition-element>



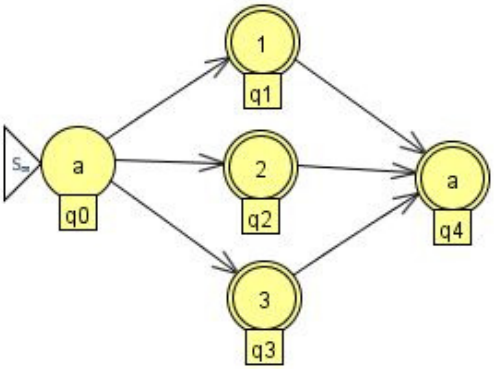
ANML uses the double bubble graphic usual in classic finite state automata to indicate state transition elements which send an output report when the input symbol matches the symbol-set. The ANML report-enabled state transition element is not "final" as used in some applications of finite state automata such as regular expression processing, any state transition element may report output irrespective of its position in an automaton. In the example above state transition element a is automatically activated at start-of-data and if an 'a' followed by a 'b' are then seen an output report will be generated by state transition element b. The automaton continues, however, processing the next input character by activated state transition element eat. eat matches on a wildcard so as long as we are not at end-of-data processing will return to a reactivated state transition element a. This automaton reports match output when it sees an "ab" after start-of-data and will continue to report match output as long as it continues to sees "ab" starting at every third input symbol.

When an state transition element generates a report an ANML device will record the state transition element that caused the report and the current position in the input buffer in the match buffer. This, effectively, gives one a pointer to the sequence of symbols matched by an automaton at the last position of the match. It can be useful to the application have this pointer at various points within the automaton, not just at a "terminal" state transition element in the automaton.

Multiple Report Output Automaton

Multiple Report Output Automaton

<state-transition-element id="q0" symbol-set="a" start="all-input">
<activate-on-match automaton="q1"/>
<activate-on-match automaton="q2"/>
<activate-on-match automaton="q3"/></state-transition-element>
<state-transition-element id="q1" symbol-set="1"><report-on-match/><activate-on-match element="q4"/></state-transition-element>
<state-transition-element id="q2" symbol-set="2"><report-on-match/><activate-on-match element="q4"/></state-transition-element>
<state-transition-element id="q3" symbol-set="3"><report-on-match/><activate-on-match element="q4"/></state-transition-element>
<state-transition-element id="q4" symbol-set="a"><report-on-match/></state-transition-element>



Any ANML state transition element may generate an output report and any number of state transition element in an automaton can be reporting enabled. A semiconductor device implementing ANML may have restrictions either in the total number or distribution of report enabled state-transition-elements but these potential restrictions are not an intrinsic part of ANML. The above automaton recognizes a sequence containing an 'a' followed by either a '1', '2', or '3' followed by an 'a', generating reports on both the second and final matching symbol of the sequence. A typical use of this kind of configuration is to provide "path information". In the above machine there are three possible paths by which we can arrive at state transition element q4. A straightforward way to record which was taken is to generate a report within each possible path segment. State transition elements q1, q2, and q3 all generate a report on the same symbol/input cycle so when a match at state transition element q4 occurs the preceding input cycle will have produced a match from one and only of q1, q2 or q3.

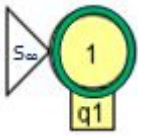
Latched Automaton

Latched Automaton

Latched state transition elements continue to assert external signals, output or activate, once they match a symbol-set until they are reset. The four possible cases, latched output, latched activate-on-match, latched output and activate-on-match, and latched with neither output nor activate-on-match are shown below.

Latched report output

<state-transition-element id="q1" symbol-set="1" start="all-input" latch="true"><report-on-match/></state-transition-element>



Latched report output state transition elements continue to assert the report signal which causes a match report once they match a symbol-set until reset. By contrast, an unlatched report output state transition element asserts the report signal only on the symbol cycle on which the symbol-set is matched.

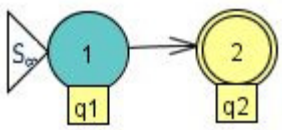
Latched report output state-transition-elements may be useful in automata which need to simultaneously assert at the end of a processing run that some set of conditions have been satisfied at some point in the course of processing. A potential disadvantage of the latched state transition element is that report output information, once reporting of output is latched, is generated at each symbol cycle, increasing the quantity of data that goes to match processing.

The graphic symbol for a latched output report state transition element is like the double bubble output report state transition element except that the area between the two circles is darkly colored.

In the example above, state transition element q1 will report on each symbol cycle once a '1' is seen in the input stream.

Latched activate-on-match

<state-transition-element id="q1" symbol-set="1" start="all-input" latch="true"><activate-on-match element="q2"/></state-transition-element>
<state-transition-element id="q2" symbol-set="2" output="true"></state-transition-element>



Latched state transition elements with activate-on match continue to assert the activation signal once they match a symbol-set until reset. By contrast, an unlatched state transition element with activate-on-match only activates connected state transition elements on the symbol cycle on which the symbol-set is matched.

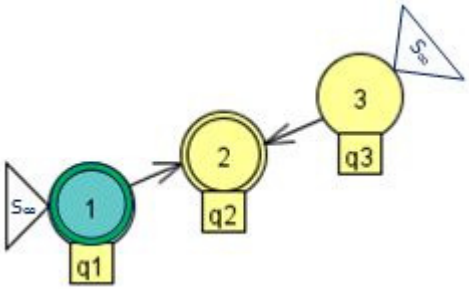
Latched state transition element with activate-on-match may be convenient to use in automata with "don't care" conditions between state transition elements.

A state transition element with latched activate-on-match has a darkly colored bubble in ANML graphic notation.

In the example above, state transition element q1 will activate state transition element q2 once a '1' is seen in the input stream and state transition element q2 will emit output if a '2' subsequently appears in the input stream. In effect, there is a "don't care" condition for the input stream between an appearance of a '1' and subsequent appearance of a '2'.

Latched report output and activate-on-match

<state-transition-element id="q1" symbol-set="1" start="all-input" latch="true"><report-on-match/><activate-on-match element="q2"/></state-transition-element>
<state-transition-element id="q2" symbol-set="2"><report-on-match/></state-transition-element>
<state-transition-element id="q3" symbol-set="3" start="all-input"><activate-on-match element="q2"/></state-transition-element>



Latched report output state transition elements with activate-on-match continue to assert the report signal and to activate connected state transition elements once they match a symbol-set until reset. By contrast, a report output state-transition-element with activate-on-match asserts the report signal and activates connected state transition elements only on the symbol cycle on which the symbol-set is matched.

A state transition element with both and activate-on-match and reporting latched may be useful in some complex automata. In the example above we may wish to know if we reached q2 through q1 but are not concerned that we record reaching q2 through other paths such as that from q3.

A state transition element which latches both report output and activate-on-matches is

indicated by dark coloring in the bubble and a dark ring between the first and second circle used to indicate that the state transition element generates report output.

Latched with neither report output nor activate-on-match

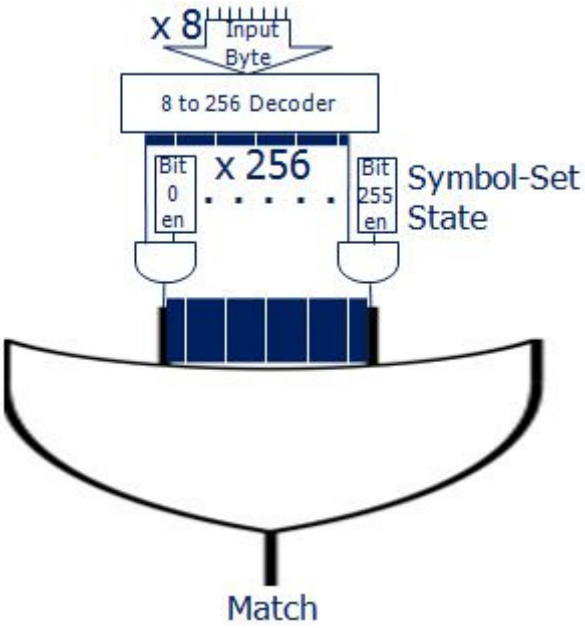
<state-transition-element id="q1" symbol-set="1" latch="true"></state-transition-element>

Despite the latch="true" attribute this state transition element behaves no differently than a state transition elment without the latch attribute (latch="false") since there is neither report-on-match nor activate-on-match to be latched.

Symbol-Set

Symbol-Set

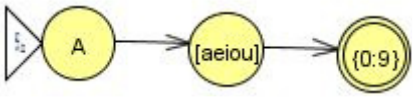
symbol-set="*character* | *character-class* | *bits-enabled* | *multiple*"



ANML machines operate on symbols, with a primary operation being recognition of a symbol against a set of symbols stored in the state transition element. The definition of a symbol is a characteristic of the implementation. In the discussion of symbol-sets in this article we will assume that a symbol is a byte (8-bit) value, both because it will be helpful for this section to think about a specific implementation and also because it is a very likely actual implementation. Therefore, assuming that the input stream consists of 8-bit values each state transition element will be programmable with a 256 position symbol-set value which will be used to determine if connected activate-on-match and report-on-match state transition elements should be activated and generate match report output, if enabled. Any combination of those 256 positions can be set. Byte values input into an activated state transition element will be tested against the symbol-set and if the input value matches any set position a "match" is recognized and activations and report output will be triggered. An abstraction of the circuitry used to store symbol-sets and to test for matches on input symbols is shown in the image above.

<state-transition-element id="q0" symbol-set="A" start="all-input"><activate-on-match automaton="q1"/></state-transition-element>
<state-transition-element id="q1" symbol-set="[aeiou]"><activate-on-match automaton="q2"/></state-transition-element>

<state-transition-element id="q2" symbol-set="{0:9}"><report-on-match/></state-transition-element>



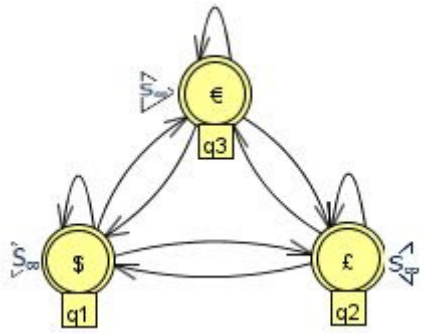
Initial settings of the symbol-set state used to store symbol values for each state transition element are set by the ANML state transition element element attribute *symbol-set*. In the graphic representation the symbol-set value is written in the bubble, not, as in classic finite automata, on the transition arrow as this better reflects that the bubble represents an actual state transition element. The example above shows an automaton which uses the three formats permitted for expressing symbol_sets, character, character-class, and bits-enabled. The syntax of each format is described in detail below. The example automaton recognizes an input sequence beginning with an upper-case 'A' followed by one of the following lower-case letters: 'a', 'e', 'i', 'o', 'u', followed by a symbol with a value between 0 and 9.

The state-transition-element has a boolean attribute case-insensitive which can affect the interpretation of the symbol-set. The case-insensitive attribute has a default value of false so when it does not appear in the state-transition-element symbol-sets are case-sensitive. Case-insensitivity is equivalent to the PCRE modifier /i. It works on character, character-class, and multiple values but not on bits-enabled.

Transitions

Transitions

```
<state-transition-element id="q1" symbol-set="$" start="all-input">
<report-on-match/>
<activate-on-match element="q1"/>
<activate-on-match element="q2"/>
<activate-on-match element="q3"/></state-transition-element>
<state-transition-element id="q2" symbol-set="£" start="all-input">
<report-on-match/>
<activate-on-match element="q1"/>
<activate-on-match element="q2"/>
<activate-on-match element="q3"/></state-transition-element>
<state-transition-element id="q3" symbol-set="€" start="all-input">
<report-on-match/>
<activate-on-match element="q1"/>
<activate-on-match element="q2"/>
<activate-on-match element="q3"/></state-transition-element>
```



Transitions, a term taken from finite automata and state machine design but which has a slightly different meaning there, indicate in ANML what element will be activated for the next input cycle when the input symbol is recognized in the state transition element's symbol-set. Each transition is specified by an activate-on-match subelement of the state transition element, where the containing state transition element is the source or causative element and the element which is activated on the next input cycle is identified by the value of the attribute "element" in the activate-on-match element. The example above shows an automaton with three state transition elements where each state transition element has transitions to the other two state transition elements as well as itself. Each state transition element will therefore have three activate-on-match subelements. In the ANML graphic representation each transition is simply represented by a uni-directional arrow between the source state transition element and the element which is activated on match.

In classic finite state automata the match value is associated with the transition but in ANML transitions are unmarked as they are more properly "activations" which occur on the match

event in the source element. ANML is also not a type of finite state transducer (Moore or Mealy machine) and does not support output values either on the transition or associated with the state transition element. Match report output from the state transition element is generated on elements designated as report-on-match, as is the case for all the state transition elements in the example automaton above, but elements which are not report output enabled do not generate report output. In a very limited sense it is possible to view the operation of an ANML machine as a finite transducer with a capability of outputing either no report output signal on a given clock cycle or a report output signal if the state transition elements are configured as report-on-match. ANML chip implementations may limit the number of report output enabled elements because of routing constraints and there may be other practical limitations that would prevent one from making much use of the finite transducer, even as a finite tranducer with the limitation of having essentially just a binary output value.

ANML transitions are specified only in the source element - an element declares what elements it activates on a match but it does not "know" what elements can activate it.

ANML chip implementations will likely also have limitations on "fan-in" and "fan-out" due to routing constraints, that is, the number of elements that can activate any given elements (fan-in) and the number of elements that can be activated by a source element (fan-out). ANML does not build-in any intrinsic or parameterized maximum fan-in or fan-out values but the designer may wish to be aware of such limitations for the target implementation.

Deterministic and Non-Deterministic Automata

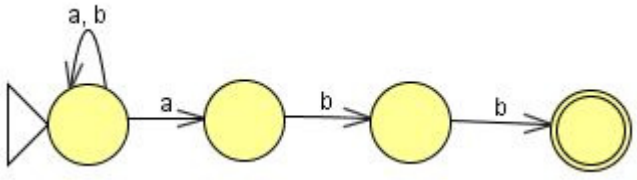
Deterministic and Non-Deterministic Automata

ANML allows for the direct expression of non-deterministic finite automata (NFA) because a source state transition element may simulataneously activate multiple elements on the same symbol-set value. This is equivalent to saying, as one would in classic finite automata, that for an NFA multiple transitions are allowed on the same input symbol.

εNFA (NFAs which also allow ε-transitions, that is, transitions without processing an input symbol), NFA and deterministic finite automata (DFA) all recognize the same class of languages, which is to say that given any automata one can always construct equivalent machines whether as εNFA, NFA or DFA. ANML permits the direct expression of DFA and although it does not allow for ε-transitions the fact that it allows multiple start states effectively enables a trivial mapping from εNFA to an NFA that can be expressed directly in ANML. (When multiple start states are allowed any εNFA can be converted into an NFA with no ε-transitions using the same states as the original εNFA. See the next section εNFA-NFA mapping further discussion)

Shown below is an example of an automaton expressed both as an NFA and its equivalent DFA and the implementation of each in ANML. The NFA and DFA use the standard formal definition of NFA and DFA and are correspondingly graphed showing the transitions as a function of the input. The ANML automaton is shown using ANML graphic notation with the symbol-set shown in the state transition element bubble. A simple algorithm for conversion between NFA and ANML notations is given in the next section.

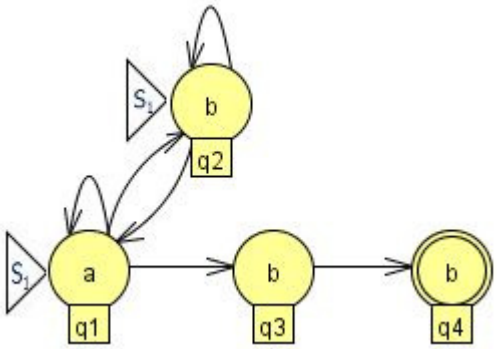
NFA



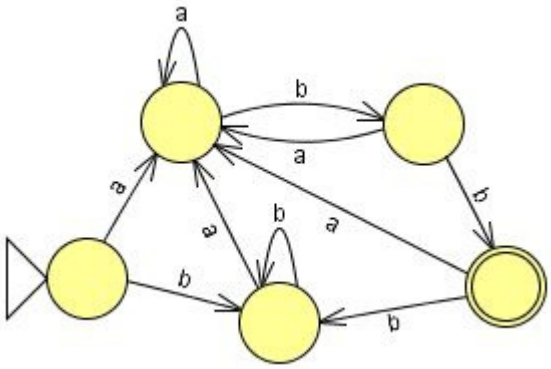
ANML Equivalent to NFA Above

```
<state-transition-element id="q1" symbol-set="a" start="start-of-data">
<activate-on-match automaton="q1"/>
<activate-on-match automaton="q2"/>
<activate-on-match automaton="q3"/></state-transition-element>
<state-transition-element id="q2" symbol-set="b" start="start-of-data">
<activate-on-match automaton="q1"/>
<activate-on-match automaton="q2"/></state-transition-element>
<state-transition-element id="q3" symbol-set="b">
<activate-on-match automaton="q4"/></automaton>
<state-transition-element id="q4" symbol-set="b"><report-on-match/></state-transition-
```

element>

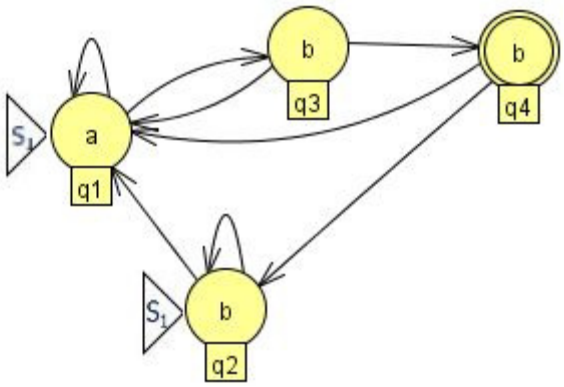


DFA Equivalent to NFA Above



ANML Equivalent to DFA Above

```
<state-transition-element id="q1" symbol-set="a" start="start-of-data">
<activate-on-match element="q1"/>
<activate-on-match element="q3"/></state-transition-element>
<state-transition-element id="q2" symbol-set="b" start="start-of-data">
<activate-on-match element="q1"/>
<activate-on-match element="q2"/></state-transition-element>
<state-transition-element id="q3" symbol-set="b">
<activate-on-match element="q1"/>
<activate-on-match element="q4"/></state-transition-element>
<state-transition-element id="q4" symbol-set="b">
<report-on-match/>
<activate-on-match element="q1"/>
<activate-on-match element="q2"/></state-transition-element>
```


b
b
q3
q4
S_1
a
q1
S_1
b
q2

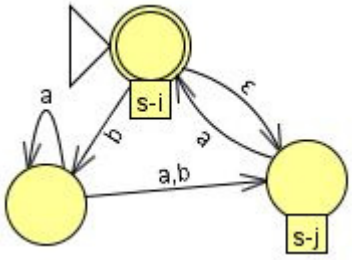
εNFA-NFA mapping further discussion

εNFA-NFA Mapping with Multiple Start States

When multiple start states are allowed, as they are in ANML, any NFA with ε-transitions can be converted into an NFA with no ε-transitions using the same states as the original εNFA. In an NFA the transition function can assign multiple states on the same input symbol. If there is an ε-transition from state S_i to S_j we can add S_j anywhere S_i is the destination of a transition function. If state S_i happens to be a start state we will need to set state S_j to a start state too, potentially resulting in multiple start states, and the ε-transition from state S_i to S_j can be removed.

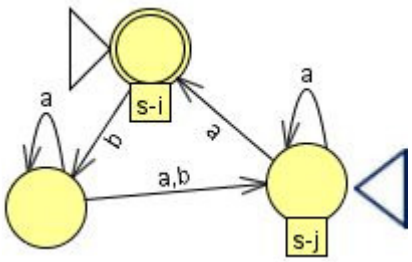
The example below illustrates use of the above algorithm to convert from an εNFA to an NFA with multiple start states.

Here is the original εNFA.



Steps to Convert to NFA without ε-transitions:

1. There is an ε-transition from state s-i to s-j. We must add s-j anywhere s-i is the destination of a transition function. There is a transition from s-j to s-i on the symbol 'a' which must change to a transition from s-j to s-j on the symbol 'a'. This is done by adding a self-loop on s-j on the symbol 'a'.
2. State s-i happens to be a start state so s-j must be set to a start state as well.
3. We can now remove the ε-transition from s-i to s-j. The new NFA, shown below, is equivalent to the original εNFA.



Converting from an NFA to ANML

Converting from an NFA to ANML

1. Write all transitions in the following form

   f(source_state, input_symbol) → *destination_state*

   If *source_state* is a start state enclose it in parentheses, *(source_state)*
   If *destination_state* is an accept state enclose it in parentheses, *(destination_state)*

2. Combine all transition statements where *f(source_state, input_symbol)* is identical and *destination_state*s are of the same type (either not accept state or accept state) into a single statement with multiple *destination_state*s. If *f(source_state, input_symbol)* is identical but one transition statement points to an accept state and another to a not accept state the transitions statements should not be combined and an additional *source_state* will be added in the next step. For example:

   The three statements

   f(source_state, input_symbol) → *destination_state*$_1$
   f(source_state, input_symbol) → *destination_state*$_2$
   f(source_state, input_symbol) → *destination_state*$_3$

   are combined into a single statement as follows

   f(source_state, input_symbol) → *destination_state*$_1$, *destination_state*$_2$, *destination_state*$_3$

3. For all transition statements where *source_state* is identical and the *input_symbol* is different the *source_state* in each statement should be postpended by an incrementing count value in each successive transition statement. If the *source_state* was enclosed in parentheses each rewritten *source_state* should also be enclosed in parentheses.

   These two transition statements

   f(source_state, input_symbol$_1$*)* → *destination_state*
   f(source_state, input_symbol$_2$*)* → *destination_state*

   would be modified into two new transition statements

   f(source_state_0, input_symbol$_1$*)* → *destination_state*

f(source_state_1, input_symbol$_2$) → destination_state

4. The *destination_state*s with value equal to the *source_state*s modifed in the prior step should be changed to match the value of the new *source_state*s, creating multiple *destination_state*s for those transition statements which are modified. For example:
    - source_state$_x$ = destination_state$_y$
    - and source_state$_x$ was changed in two transition statements to source_state$_x$_0 and source_state$_x$_1
    - therefore, everywhere that destination_state$_y$ occurs should be changed to the two destinations with the values of source_state$_x$_0 and source_state$_x$_1
5. If a *destination_state* is never also a *start_state* it does not match an *input_symbol* and can be removed from the transition expression. If such a *destination_state* is also an accept state the *destination_state* value should be removed but the empty parentheses retained.
6. Each transition statement will now become one ANML *state-transition-element* element where the *symbol-set* attribute value is the *input_symbol* and containing one *activate-on-match* element for each *destination_state*. The *id* attribute of the *state-transition-element* element should be the *source_state* value and if the *source_state* is in parentheses the attribute and attribute value *start="start-of-data"* should be included in the *state-transition-element* element. If the *destination_state* of the transition statement is enclosed in parentheses the attribute and attribute value *output="enabled"* should be included in its *state-transition-element* element, including empty *destination_state*s created on the prior step. If, after removal of *destination_state*s in the prior step there are no *destination_state*s in a particular transition expression there will be no *activate-on-match* elements nested in the *state-transition-element* element.

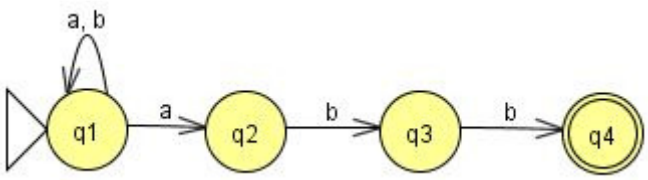
See next section Worked Example of NFA to ANML Conversion

Worked Example of NFA to ANML Conversion

Worked Example of NFA to ANML Conversion

Red text shows what will be transformed in the following step.

NFA



1. Write the NFA as transition expressions

f((q1), a) → q1
f((q1), a) → q2
f((q1), b) → q1
f(q2 , b) → q3
f(q3 , b) → (q4)

2. Combine transition expressions with the same *source_state*, *input_symbol*

f((q1), a) → q1 , q2
f((q1), b) → q1
f(q2 , b) → q3
f(q3 , b) → (q4)

3. Rewrite *source_state* names to uniquely identify by *input_symbol* processed

f((q1-0), a) → q1 , q2
f((q1-1), b) → q1
f(q2 , b) → q3
f(q3 , b) → (q4)

4. Rewrite *destination_state* names corresponding to the changes made to *source_state*

f((q1-0), a) → q1-0 , q1-1 , q2
f((q1-1), b) → q1-0, q1-1
f(q2 , b) → q3
f(q3 , b) → (q4)

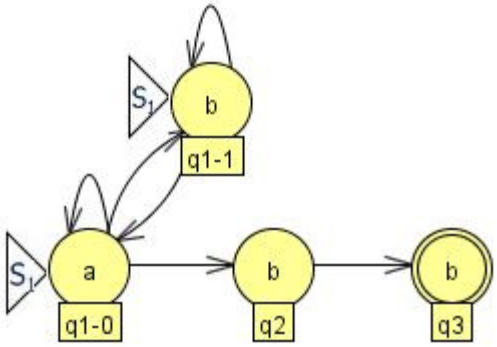
5. Remove *destination_state*s which are never *start_state*s

f((q1-0), a) → q1-0 , q1-1 , q2

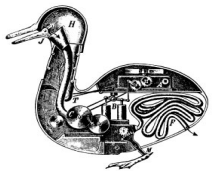
f((q1-1), b) → q1-0, q1-1
f(q2 , b) → q3
f(q3 , b) → ()

6. Each transition expressions now describes an ANML state transition element element.

```
<state-transition-element aid="q1-0" symbol-set="a" start="start-of-data">
  <activate-on-match element="q1-0"/>
  <activate-on-match element="q1-1"/>
  <activate-on-match element="q2"/></state-transition-element>
<state-transition-element aid="q1-1" symbol-set="b" start="start-of-data">
  <activate-on-match element="q1-0"/>
  <activate-on-match element="q1-1"/></state-transition-element>
<state-transition-element aid="q2" symbol-set="b">
  <activate-on-match element="q3"/></state-transition-element>
<astate-transition-element aid="q2" symbol-set="b"><report-on-match/></state-transition-
element>
```



Counter Element



Counter Element

Counters are a special type of automaton element used to count in ANML. They cannot accept symbols from the input source so they always work in conjuction with state transition elements. In the markup language they are simply called *counter*s. There are two possible subelements of the counter specifying actions that may occur on the event of reaching the target count, report-on-target and activate-on-target. Only up to one report-on-target may be associated with a target while zero or more activate-on-target may be specified. In the ANML XML coding the report-on-target element must precede any activate-on-target elements.

Syntax

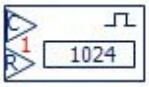
	attribute	type/value	occurence	example
<counter			0 or more	
	countone=	*id*	required unique in macro	"c1"
	reset=	*id*	required unique in macro	"r1"
	target=	*number* *range: 0 to max_target* *(max_target is implementation dependant)*	required	"1024"
	at_target=	*"pulse"* *"latch"* *"roll"*	default: *"pulse"*	"pulse" "latch" "roll"
>				
<report-on-target/>			0 or 1	
<activate-on-target			0 or more	
	macro=	*self* *idref*	default: *self*	"self" "radial"
	element=	*idref*	required	"c2"
/></counter>				

Counter

A Simple Automaton with a Counter Element

A Simple Automaton with a Counter Element

<counter countone="c1" reset="r1" target="1024" at_target="pulse"></counter>



ANML has a special device for counting. The ANML counter element is a stateful machine programmed with a specific target value which raises an activation signal when it has been triggered a number of times equal to the programmed target value after being set at its initial state or after reset.

The ANML counter is initially configured with the target value and a mode of operation which determines the behavior when the count reaches the target. In the example the target value is "1024" and the mode of operation when the count reaches the target is "pulse". The operation modes will be discussed in detail in a later article. There are two inputs to the ANML counter. One input, "countone", causes the internal count stored in the counter state to advance by one. The other input, "reset", causes the counter to return to its initial state, that is, to restore the count value in the counter state such that the counter will not raise an activation signal until it receives input on "countone" a number of times equal to the target value. The "countone" and "reset" inputs are string values representing identifiers, equivalent to the id for the state transition element. They are single bit values which are generated by activate signals from other elements.

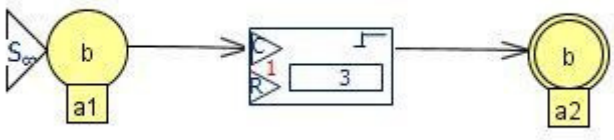
In the graphic representation of the ANML counter a rectangle is used for the counter shape and the inputs "countone" and "reset" are represented by the 'C' and 'R' input connection triangles. The id for both "countone" and "reset" is expressed by text, in red, between the triangles. By convention, the "countone" id will start with a lower case letter 'c' and "reset" with a lower case letter 'r' and both will have a common value for the remainder of the id value which is what is written in the red text next to the input triangles. In the example a '1' is placed between the triangles, indicating that the "countone" value is "c1" and the "reset" value is "r1". The "at_target" operation mode is represented by a symbol in the corner of the rectangle, in the example above, a rising and falling waveform represents "pulse". The "target" is written inside a "window" in the counter rectange.

The counter cannot function unless it is connected to elements driving input. The next article will show how to create automata using counters and state transition elements.

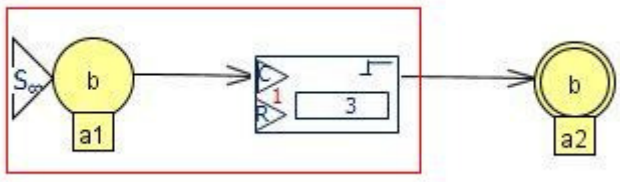
Counting n Occurences

Counting n Occurences

<state-transition-element id="a1" symbol-set="b" start="all-input"><activate-on-match automaton="c1"/></state-transition-element>
<counter countone="c1" reset="r1" target="3" at_target="latch"><activate-on-target element="a2"/>
</counter>
<state-transition-element id="a2" symbol-set="b" output="enabled"></state-transition-element>



The ANML automaton in the example uses a counter to recognize 4 'b's in an input stream (the 'b's do not need to be in a consecutive sequence). A counter does not examine the input symbols and does not match patterns so it must be used in conjunction with state transition elements. We need the first state transition element, a1, to recognize a 'b' and cause the counter to count and we use the final state transition element, a2, to generate output. The counter receives input from the elements connected to it, modifies its internal state and has its output latched all within the same cycle used by the input state transition elements to evaluate the input against its symbol-set. It may be helpful to think of a counter and all the state transition elements that input to it as a single device, as suggested below by redrawing the example with a box around the counter and state transition element a1.

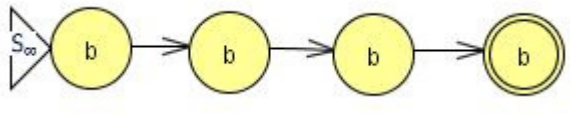


The operation of this automaton is given in detail, cycle-per-cycle, in the table below.

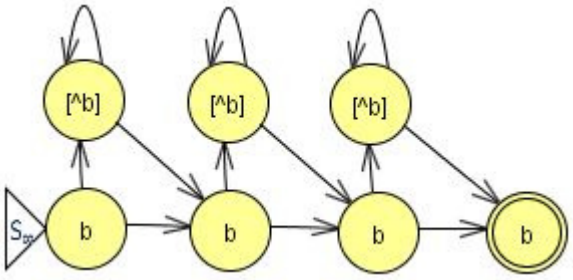
Input will be 4 'b's, designated in sequence as b_1 b_2 b_3 b_4
State transition element a_1 is always activated due all-input start configuation

Cycle	Input Symbol	Actions
1	b_1	$b_1 \rightarrow a_1$ causing activate-on-match C_1, C_1 count advances
2	b_2	$b_2 \rightarrow a_1$ causing activate-on-match C_1, C_1 count advances
3	b_3	$b_3 \rightarrow a_1$ causing activate-on-match c1 C_1 count advances to target causing activate-on-target a2
4	b_4	$b_4 \rightarrow a_1$ causing activate-on-match c_1 C_1 count remains at target (no reset, but a2 is re-activated due to C_1 "latch" setting) $b_4 \rightarrow a_2$ causing output

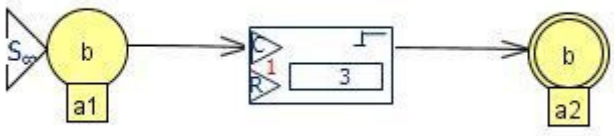
This example could be readily implemented using only state transition elements, and given the low target value a state transition element-only implementation would probably be a better use of resources on a semiconductor implementation. However, the equivalent state transition element-only design is NOT the following very simple automaton:



It is the somewhat more complicated automaton below since our example allows non-'b' symbols interspersed with the 'b's.



Below the operation of our counter machine on a input stream with a mixture of 'b's and non-'b's is illustrated cycle by cycle.



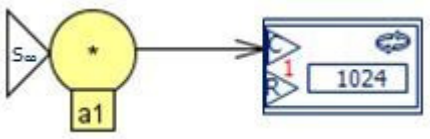
Input b_1 X X b_2 b_3 b_4 X b_5

Cycle	Input Symbol	Actions
1	b_1	$b_1 \rightarrow a_1$ causing activate-on-match c_1, C_1 count advances
2	X	
3	X	
4	b_2	$b_2 \rightarrow a_1$ causing activate-on-match c_1, C_1 count advances
5	b_3	$b_3 \rightarrow a_1$ causing activate-on-match c_1 C_1 count advances to target causing activate-on-target a2
6	b_4	$b_4 \rightarrow a_1$ causing activate-on-match c_1 C_1 count remains at target (no reset, but a2 is re-activated due to C_1 "latch" setting) $b_4 \rightarrow a_2$ causing output
7	X	C_1 count remains at target (no reset, but a2 is re-activated due to C_1 "latch" setting)
8	b_5	$b_5 \rightarrow a_1$ causing activate-on-match c_1 C_1 count remains at target (no reset, but a2 is re-activated due to C_1 "latch" setting) $b_5 \rightarrow a_2$ causing output

Counter Report Output

Counter Report Output

<state-transition-element id="a1" symbol-set="*" start="all-input"><activate-on-match element="c1"/></state-transition-element>
<counter countone="c1" reset="r1" target="1024" at_target="rollover"><report-on-target/></counter>

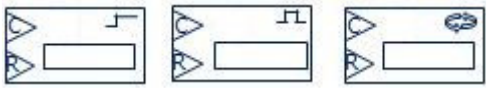


Counters can report on reaching the target. The automaton in the example generates a report on every 1024th input symbol. The report will be associated with the counter countone id c1. The ANML graphic notation indicates an report output generating counter by a double box.

Operation Modes, Latch, Pulse, Roll

Operation Modes: Latch, Pulse, Roll

at_target="latch | pulse | roll"



The *at_target* attribute configures ANML counters for modes of operation, causing different behaviors when the counter counts to the target.

latch persistently activates the elements connected to it through the counter's XML *activate-on-target* elements. The counter value holds at the target. It as if, on each cycle, the counter reached the target value.

pulse activates the elements connected to it through the counter's XML *activate-on-target* elements on the cycle on which the counter value reaches the target and on subsequent cycles does not activate the connected elements. The counter value holds at the target but it is as if the counter "shuts off" after generating a single activation.

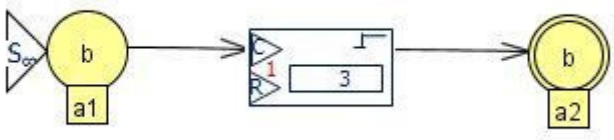
roll activates the elements connected to it through the counter's XML *activate-on-target* elements on the cycle on which the counter value reaches the target and resets the counter value to its initial state, such that it will not activate connected elements until it counts again to the target. This has an effect which is identical to sending an input to *reset* when the target is reached.

In a previous article an automaton was implemented using the *latch* operation mode to count occurences of 'b's. Changing the operation mode to *pulse* or *roll* would change the language recognized by the automaton. The following stream of input symbols will be submitted against each machine, stepping cycle-by-cycle to observe its precise behavior (each occurence of a b is subscripted with its sequence number):

b_1 X X b_2 b_3 b_4 X b_5 b_6 b_7 b_8 b_9

latch

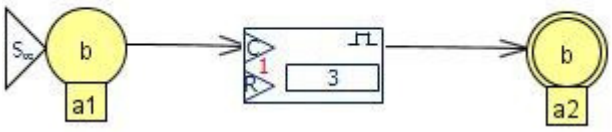
```
<state-transition-element id="a1" symbol-set="b" start="all-input"><activate-on-match
element="c1"/></state-transition-element>
<counter countone="c1" reset="r1" target="3" at_target="latch"><activate-on-target
element="a2"/>
</counter>
<state-transition-element id="a2" symbol-set="b"><report-on-match/></state-transition-
element>
```



The machine generates a match report from a2 on receipt of symbols b4, b5, b6, b7, b8 and b9. See section below for a latch cycle-by-cycle analysis of the machine's operation and for its equivalent implementation using only state transition elements.

pulse

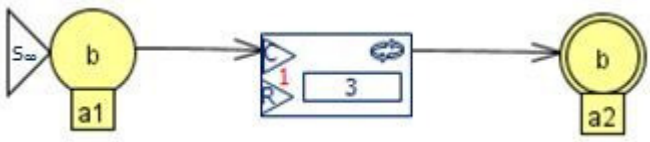
<state-transition-element id="a1" symbol-set="b" start="all-input"><activate-on-match element="c1"/></state-transition-element>
<counter countone="c1" reset="r1" target="3" at_target="pulse"><activate-on-target element="a2"/>
</counter>
<state-transition-element id="a2" symbol-set="b"><report-on-match/></state-transition-element>



The machine generates a match report only at b4, essentially shutting off after the first time the counter activates a2. See section below for a pulse cycle-by-cycle analysis of the machine's operation and for its equivalent implementation using only state transition elements.

Roll

<state-transition-element id="a1" symbol-set="b" start="all-input"><activate-on-match automaton="c1"/></state-transition-element>
<counter countone="c1" reset="r1" target="3" at_target="pulse"><activate-on-target element="a2"/>
</counter>
<state-transition-element id="a2" symbol-set="b"><report-on-match/></state-transition-element>

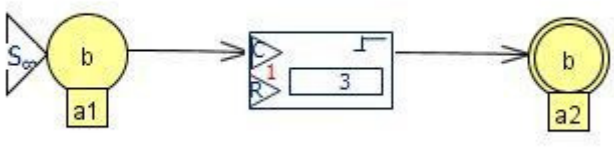


The machine generates a match report from a2 on receipt of b4 and b7. Four 'b's are needed to generate the first match report but only three for the second because b4 is both the last of the first sequence and the first of next sequence. See section below for a roll cycle-by-cycle analysis of the machine's operation and for its equivalent implementation using only state transition elements.

Latch Cycle-by-Cycle and Automata Equivalent

latch

<state-transition-element id="a1" symbol-set="b" start="all-input"><activate-on-match element="c1"/></state-transition-element>
<counter countone="c1" reset="r1" target="3" at_target="latch"><activate-on-target element="a2"/>
<state-transition-element id="a2" symbol-set="b"><report-on-match/></state-transition-element>



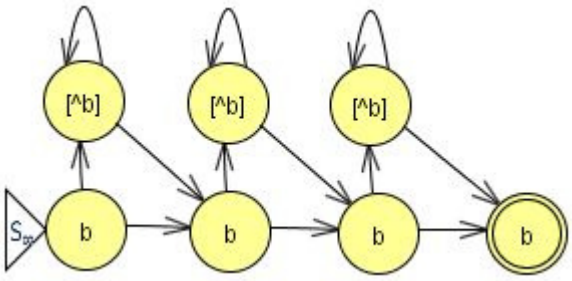
Input Stream

b_1 X X b_2 b_3 b_4 X b_5 b_6 b_7 b_8 b_9

Cycle-by-Cycle Analysis

Cycle	Input Symbol	Actions
1	b_1	$b_1 \rightarrow a_1$ causing activate-on-match c_1, C_1 count advances
2	X	
3	X	
4	b_2	$b_2 \rightarrow a_1$ causing activate-on-match c_1, C_1 count advances
5	b_3	$b_3 \rightarrow a_1$ causing activate-on-match c_1 C_1 count advances to target causing activate-on-target a2
6	b_4	$b_4 \rightarrow a_1$ causing activate-on-match c_1 $b_4 \rightarrow a2$ causing output C_1 count remains at target (no reset) a_2 is activated due to "latch" setting
7	X	C_1 count remains at target (no reset) a_2 is activated due to "latch" setting
8	b_5	$b_5 \rightarrow a_1$ causing activate-on-match c_1 $b_5 \rightarrow a2$ causing output C_1 count remains at target (no reset) a_2 is activated due to "latch" setting
9	b_6	$b_6 \rightarrow a_1$ causing activate-on-match c_1 $b_6 \rightarrow a2$ causing output C_1 count remains at target (no reset) a_2 is activated due to "latch" setting
10	b_7	$b_7 \rightarrow a_1$ causing activate-on-match c_1 $b_7 \rightarrow a_2$ causing output C_1 count remains at target (no reset) a_2 is activated due to "latch" setting
11	b_8	$b_8 \rightarrow a_1$ causing activate-on-match c_1 $b_8 \rightarrow a_2$ causing output C_1 count remains at target (no reset) a_2 is activated due to "latch" setting
12	b_9	$b_9 \rightarrow a_1$ causing activate-on-match c_1 $b_9 \rightarrow a_2$ causing output C_1 count remains at target (no reset) a_2 is activated due to "latch" setting

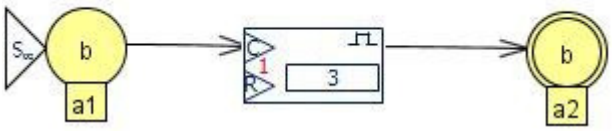
State Transition Element-Only Equivalent



Pulse Cycle-by-Cycle and Automata Equivalent

Pulse

<state-transition-element id="a1" symbol-set="b" start="all-input"><activate-on-match element="c1"/></state-transition-element>
<counter countone="c1" reset="r1" target="3" at_target="pulse"><activate-on-target element="a2"/>
<state-transition-element id="a2" symbol-set="b"><report-on-match/></state-transition-element>



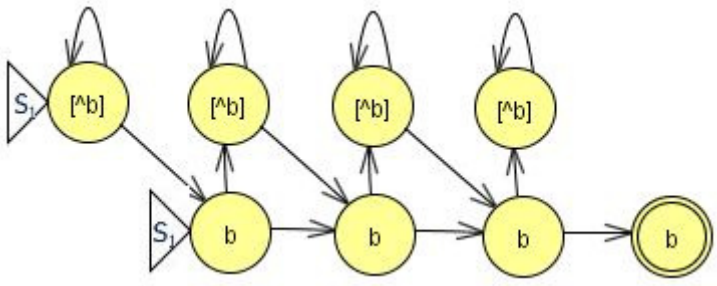
Input Stream

b_1 X X b_2 b_3 b_4 X b_5 b_6 b_7 b_8 b_9

Cycle-by-Cycle Analysis

Cycle	Input Symbol	Actions
1	b_1	$b_1 \rightarrow a_1$ causing activate-on-match c_1, C_1 count advances
2	X	
3	X	
4	b_2	$b_2 \rightarrow a_1$ causing activate-on-match c_1, C_1 count advances
5	b_3	$b_3 \rightarrow a_1$ causing activate-on-match c_1 C_1 count advances to target causing activate-on-target a_2
6	b_4	$b_4 \rightarrow a_1$ causing activate-on-match c_1 $b_4 \rightarrow a_2$ causing output C_1 count remains at target (no reset) a_2 is NOT activated due to "pulse" setting
7	X	C_1 count remains at target (no reset) a_2 is NOT activated due to "pulse" setting
8	b_5	$b_5 \rightarrow a_1$ causing activate-on-match c_1 C_1 count remains at target (no reset) NO output from a_2 because it wasn't activated
9	b_6	$b_6 \rightarrow a_1$ causing activate-on-match c_1 C_1 count remains at target (no reset) NO output from a_2 because it wasn't activated
10	b_7	$b_7 \rightarrow a_1$ causing activate-on-match c_1 C_1 count remains at target (no reset) NO output from a_2 because it wasn't activated
11	b_8	$b_8 \rightarrow a_1$ causing activate-on-match c_1 C_1 count remains at target (no reset) NO output from a_2 because it wasn't activated
12	b_9	$b_9 \rightarrow a_1$ causing activate-on-match c_1 C_1 count remains at target (no reset) NO output from a_2 because it wasn't activated

State Transition Element-Only Equivalent

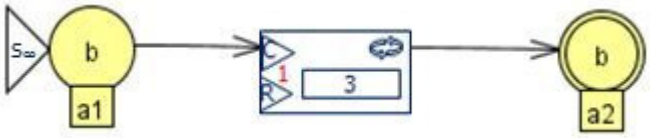


This automaton must use the start-of-data start instead of all-input since it will not recognize subsequent sequences after the counter reaches the target the first time and additional input should not be propagated through the automaton once that occurs. A state transition element with a '[^b]' character class with a start-of-data start must also be added to enable the machine to work in the case where the first symbol is not 'b'. This machine is only language-equivalent instead of simply equivalent to the version using a counter because in the counter version the first state transition element is continually activated because of the

Roll Cycle-by-Cycle and Automata Equivalent

roll

<state-transition-element id="a1" symbol-set="b" start="all-input"><activate-on-match element="c1"/></state-transition-element>
<counter countone="c1" reset="r1" target="3" at_target="roll"><activate-on-target element="a2"/>
<state-transition-element id="a2" symbol-set="b"><report-on-match/></state-transition-element>



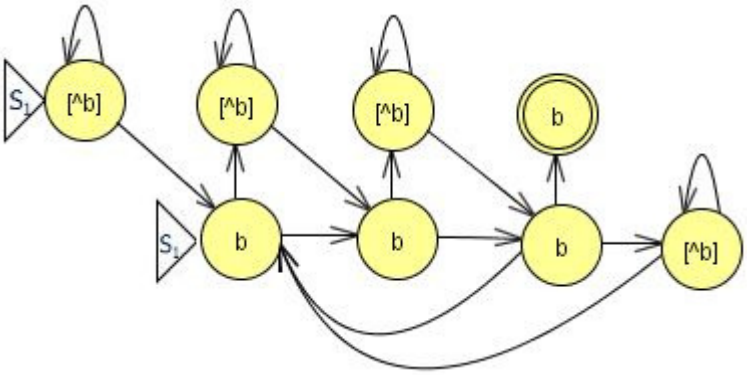
Input Stream

b_1 X X b_2 b_3 b_4 X b_5 b_6 b_7 b_8 b_9

Cycle-by-Cycle Analysis

Cycle	Input Symbol	Actions
1	b_1	$b_1 \rightarrow a_1$ causing activate-on-match c_1, C_1 count advances
2	X	
3	X	
4	b_2	$b_2 \rightarrow a_1$ causing activate-on-match c_1, C_1 count advances
5	b_3	$b_3 \rightarrow a_1$ causing activate-on-match c_1 C_1 count advances to target causing activate-on-target a_2 C_1 count reset
6	b_4	$b_4 \rightarrow a_1$ causing activate-on-match c_1, C_1 count advances No activate-on-target for a_2 due to roll $b_4 \rightarrow$a2 causing output
7	X	
8	b_5	$b_5 \rightarrow a_1$ causing activate-on-match c_1, C_1 count advances NO output from a_2 because it wasn't activated
9	b_6	$b_6 \rightarrow a_1$ causing activate-on-match c_1 C_1 count advances to target causing activate-on-target a_2 C_1 count reset
10	b_7	$b_7 \rightarrow a_1$ causing activate-on-match c_1, C_1 count advances No activate-on-target for a_2 due to roll $b_7 \rightarrow$a2 causing output
11	b_8	$b_8 \rightarrow a_1$ causing activate-on-match c_1, C_1 count advances
12	b_9	$b_9 \rightarrow a_1$ causing activate-on-match c_1 C_1 count advances to target causing activate-on-target a_2 C_1 count reset

State Transition Element-Only Equivalent



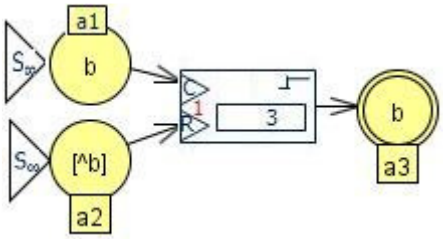
Like the pulse counter machine, this automaton must use the start-of-data start instead of all-input to avoid having 'b's cascade into a2 after the first pass through the counter. However, because of the roll the automaton should restart the count sequence once the first 'b' is seen after the count reset. A state transition element with a '[^b]' character class with a start-of-data start must also be added to enable the automaton to work in the case where the first symbol is not 'b'. This automaton is only language-equivalent instead of simply equivalent to the version using a counter because in the counter version the first state transition element is

continually activated because of the all-input start while in the only-state transition element version the the state transition element which recognizes the first 'b' can only be activated once. The two automata do, however, generate output on the same set of input sequences.

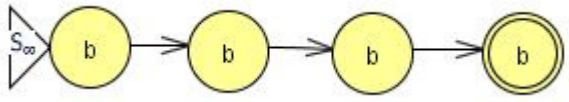
Reset and Counting in Sequence

Reset and Counting in Sequence

<state-transition-element id="a1" symbol-set="b" start="all-input"><activate-on-match element="c1"/></state-transition-element>
<state-transition-element id="a2" symbol-set="[^b]" start="all-input"><activate-on-match element="r1"/></state-transition-element>
<counter countone="c1" reset="r1" target="3" at_target="latch">
<activate-on-target element="a3"/></counter>
<state-transition-element id="a3 symbol-set="b"><report-on-match/></state-transition-element>



In a previous article an automaton was constructed to count 4 'b's but allowed not 'b' symbols to be interspersed between 'b's. This example shows an automaton that counts only sequences of 4 consecutive 'b's using the counter reset, the equivalent of the following state transition element-only implementation:



State transition element a2 examines each input symbol and will reset the counter if a symbol which is not a 'b' is seen. Reset causes the counter value to be restored to the initial value, restarting the count. The exact series of operations, cycle-by-cycle is shown below.

Cycle	Input Symbol	Actions
1	b_1	$b_1 \rightarrow a_1$ causing activate-on-match c_1, C_1 count advances
2	b_2	$b_2 \rightarrow a_1$ causing activate-on-match c_1, C_1 count advances
3	b_3	$b_3 \rightarrow a_1$ causing activate-on-match c_1 C_1 count advances to target causing activate-on-target a_3 C_1 count remains at target (no reset)
4	b_4	$b_4 \rightarrow a_1$ causing activate-on-match c_1 $b_4 \rightarrow a_3$ causing output C_1 count remains at target (no reset) a3 activated
5	b_5	$b_5 \rightarrow a_1$ causing activate-on-match c_1 $b_5 \rightarrow a_3$ causing output C_1 count remains at target (no reset) a3 activated
6	X	$X \rightarrow a_2$ causing activate-on-match R_1 C_1 Count Reset, a_3 is deactivated
7	b_6	$b_6 \rightarrow a_1$ causing activate-on-match c_1, C_1 count advances
8	b_7	$b_7 \rightarrow a_1$ causing activate-on-match c_1, C_1 count advances
9	b_8	$b_8 \rightarrow a_1$ causing activate-on-match c_1 C_1 count advances to target causing activate-on-target a_3 C_1 count remains at target (no reset)
10	b_9	$b_9 \rightarrow a_1$ causing activate-on-match c_1 $b_9 \rightarrow a_3$ causing output C_1 count reamins at target (no reset) a3 activate
11	X	$X \rightarrow a_2$ causing activate-on-match R_1 C_1 Count Reset, a_3 is deactivated
12	b_{10}	$b10 \rightarrow a_1$ causing activate-on-match c_1, C_1 count advances

Using Operation Modes to Match First, Discrete and Overlapping Sequences

Using Operation Modes to Match First, Discrete and Overlapping Sequences

Counter operation modes can be used to control whether an automaton generates output on the first sequence of number of symbols or on each discrete, non-overlapping set or on overlapping sets. For example, matching three 'b's in following stream of 9 'b's

b_1 b_2 b_3 b_4 b_5 b_6 b_7 b_8 b_9

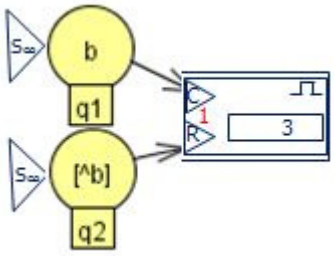
will, for each category, produce the following input

first	output 1X after b_1 b_2 b_3
discrete	output 3X after b_1 b_2 b_3, b_4 b_5 b_6, b_7 b_8 b_9
overlapping	output 7X after b_1 b_2 b_3, b_2 b_3 b_4, b_3 b_4 b_5, b_4 b_5 b_6, b_5 b_6 b_7, b_6 b_7 b_8, b_7 b_8 b_9

First can be accomplished with a pulse counter, discrete with rollover and overlapping with latch. Each of these modes was analyzed in detail in a prior article.

First Sequence Using Pulse Mode

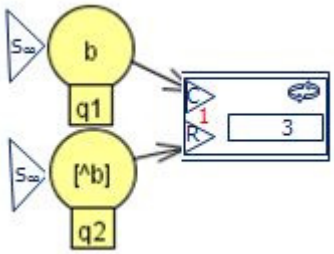
```
<state-transition-element id="q1" symbol-set="b" start="all-input"><activate-on-match
element="c1"/></state-transition-element>
<state-transition-element id="q2" symbol-set="[^b]" start="all-input"><activate-on-match
element="r1"/></state-transition-element>
<counter countone="c1" reset="r1" target="3" at_target="pulse">
<report-on-target/></counter>
```



Discrete Sequences Using Rollover Mode

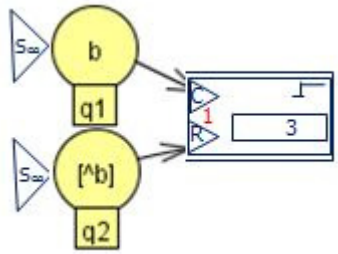
```
<state-transition-element id="q1" symbol-set="b" start="all-input"><activate-on-match
element="c1"/></state-transition-element>
<state-transition-element id="q2" symbol-set="[^b]" start="all-input"><activate-on-match
```

element="r1"/></state-transition-element>
<counter countone="c1" reset="r1" target="3" at_target="roll">
<report-on-target/></counter>



Overlapping Sequences Using Latch Mode

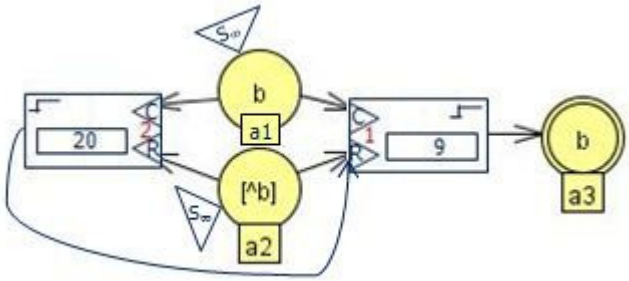
<state-transition-element id="q1" symbol-set="b" start="all-input"><activate-on-match element="c1"/></state-transition-element>
<state-transition-element id="q2" symbol-set="[^b]" start="all-input"><activate-on-match element="r1"/></state-transition-element>
<counter countone="c1" reset="r1" target="3" at_target="latch">
<report-on-target/></counter>



Counting Ranges

Counting Ranges

```
<state-transition-element id="a1" symbol-set="b" start="all-input">
<activate-on-match element="c1"/>
<activate-on-match element="c2"/>
</state-transition-element>
<state-transition-element id="a2" symbol-set="[^b]" start="all-input">
<activate-on-match element="r1"/>
<activate-on-match element="r2"/>
</state-transition-element>
<counter countone="c1" reset="r1" target="9" at_target="latch">
<activate-on-target element="a3"/>
</counter>
<state-transition-element id="a3 symbol-set="b"><report-on-match/></state-transition-
element>
</counter countone="c2" reset="r2" target="20" at_target="latch">
<activate-on-target element="r1"/>
</counter>
```



This example uses two counters to recognize input sequences with between 10 and 20 'b's, inclusive. Counter c1 counts 9 'b's, enabling state transition element a3 to recognize the tenth 'b' and output and thereafter latches the activation signal to a3 so that a3 will continue to output as long as 'b's arrive and there is no reset. Counter counts 20 'b's and then activates the reset r1 on counter c1

Combinatorial Elements

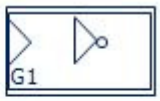
Combinatorial Elements

Combinatorial Elements perform combinatorial logic functions in ANML. Combinatorial logic includes boolean operations like AND and OR and does not have state that is used to compute results such as activations or output. State transition elements have state in the symbol-set and counter elements have state in the current count value.

Inverter Element

Inverter Element

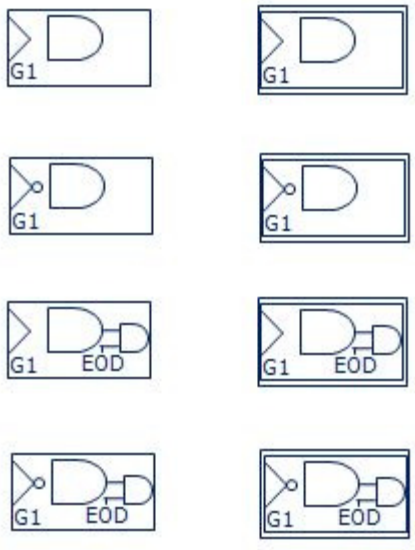
	attribute	type/value	occurence	example
<inverter	id=	*id*	required unique in macro	"b1"
	high-only-on-eod=	*"true"* *"false"*	default: *false*	"true"
>				
<report-on-high/>			0 or 1	
<activate-on-high			0 or more	
	macro=	*"self"* *idref*	default: self	"self" "radial"
	element=	*idref*	required	"q1"
/></inverter>				



AND Element

AND Element

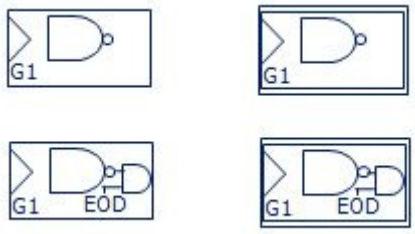
	attribute	type/value	occurence	example
<and	id=	*id*	required unique in macro	"b1"
	negate-inputs=	*"true"* *"false"*	default: *false*	"true"
	high-only-on-eod=	*"true"* *"false"*	default: *false*	"true"
>				
<report-on-high/>			0 or 1	
<activate-on-high			0 or more	
	macro=	*"self"* *idref*	default: self	"self" "radial"
	element=	*idref*	required	"q1"
/></and>				



NAND Element

NAND Element

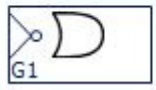
	attribute	type/value	occurence	example
<nand	id=	*id*	required unique in macro	"b1"
	high-only-on-eod=	*"true"* *"false"*	default: *false*	"true"
>				
<report-on-high/>			0 or 1	
<activate-on-high			0 or more	
	macro=	*"self"* *idref*	default: self	"self" "radial"
	element=	*idref*	required	"q1"
/></nand>				

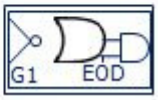


OR Element

OR Element

	attribute	type/value	occurence	example
<or	id=	*id*	required unique in macro	"b1"
	negate-inputs=	*"true"* *"false"*	default: *false*	"true"
	high-only-on-eod=	*"true"* *"false"*	default: *false*	"true"
>				
<report-on-high/>			0 or 1	
<activate-on-high			0 or more	
	macro=	*"self"* *idref*	default: self	"self" "radial"
	element=	*idref*	required	"q1"
/></or>				

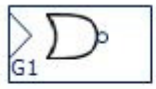


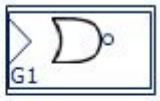


NOR Element

NOR Element

	attribute	type/value	occurence	example
<nor	id=	*id*	required unique in macro	"b1"
	high-only-on-eod=	*"true"* *"false"*	default: *false*	"true"
>				
<report-on-high/>			0 or 1	
<activate-on-high			0 or more	
	macro=	*"self"* *idref*	default: self	"self" "radial"
	element=	*idref*	required	"q1"
/></nor>				





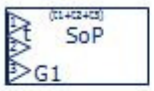
Sum-of-products Element

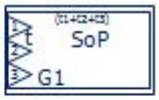
Sum-of-Products Element

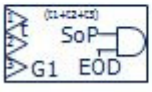
The Sum-of-Products performs a sum of products operation on a set of inputs. The number of product terms which can be summed is an implementation parameter, as is the number of inputs in each product term.

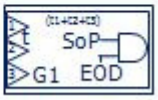
Here is a tutorial on the concept of sum-of-products.

	attribute	type/value	occurence	example
<sum-of-products	id	*id*	required unique in macro	"g1"
	high-only-on-eod=	*"true"* *"false"*	default: *false*	"true"
>				
<product-term			2 to max by implementation parameter	
	id		required unique in macro	"t1"
/>				
<report-on-high/>			0 or 1	
<activate-on-high			0 or more	
	macro=	*"self"* *idref*	default: *self*	"self" "radial"
	element=	*idref*	required	"q1"
/></sum-of-products>				









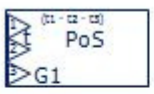
Product-of-sums Element

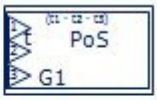
Product-of-Sums Element

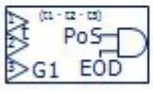
The Product-of-Sums performs a product of sums operation on a set of inputs. The number of summed terms for which we take the product is an implementation parameter, as is the number of inputs in each summing term.

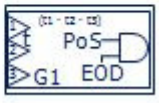
Here is a tutorial on the concept of sum-of-products.

	attribute	type/value	occurence	example
<product-of-sums	id	*id*	required unique in macro	"g1"
	high-only-on-eod=	*"true"* *"false"*	default: *false*	"true"
>				
<sum-term			2 to max by implementation parameter	
	id		required unique in macro	"t1"
/>				
<report-on-high/>			0 or 1	
<activate-on-high			0 or more	
	macro=	*"self"* *idref*	default: self	"self" "radial"
	element=	*idref*	required	"q1"
/></product-of-sums>				









Macro

Macro

Macros are containers for automata, including automata generated from regexes and automata contained in nested macros. They may have input ports and output ports, defined by the inclusion of the *port-definition* subelement and *element-reference*s. A macro without a *port-definition* is an isolated automaton not connected to other macros or elements in an automata network. If an element is included by reference in the port it can be activated by elements outside of the macro in the case of of *in type*, activate elements outside the macro in the case of an *out type* and both in the case of *inout type*.

Macros can be specified as "power-regions" by setting the *power-region* attribute to be *true*. ANML implementations may offer management features allowing macros designated as power-regions to be turned on and off as needed. This could be used to limit the generation of output to activated macros or for user-controlled explicit power management. Power-regions could be implemented as a compiler directive or as a suggestion to place a group of macros in a sector which may be individually turned on or off. Programmatic control at runtime would be enabled through the macro id.

Syntax

	attribute	type/value	occurence	example
<macro			0 or more	
	name=	*string*	required	"atcg seq"
	id=	*id*	required each macro id must be unique	"ca12"
	power-region	*boolean*	default: false	"true"
<port-definition			0 or 1	
<element-reference			0 or more	
	macro	*self* *idref*	default: self	"cell1"
	element	*idref*	required	"a2"
	type	*state-transition-element* *counter* *or* *and* *nand* *nor* *sum-of-products* *product-of-sums*	required	"or"
	activation	*in* *out* *inout* *none*	default: none	
	reporting	*boolean*	default: false	
	start	*start-of-data* *all-input* *none*	default: none	
/></port-definition>				
<macro			0 or more	
<*any automaton element*			0 or more	
<regex			0 or more	
</macro>				

Macro Examples

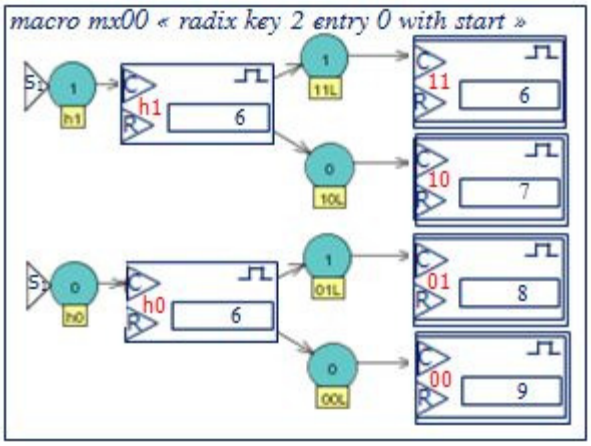
A Macro with no external connections
A Macro with input external connections but no output external connections
Proposals for new Macro structures

A Macro with no external connections

A Macro with No External Connections

```
<macro name="radix key 2 entry 0 with start" id="mx00">
        <port-definition>
          <element-reference element="h1" type="state-transition-element" start="start-of-
data"/>
          <element-reference element="h0" type="state-transition-element" start="start-of-
data"/>
          <element-reference element="c11" type="state-transition-element"
reporting="true"/>
          <element-reference element="c10" type="state-transition-element"
reporting="true"/>
          <element-reference element="c01" type="state-transition-element"
reporting="true"/>
          <element-reference element="c00" type="state-transition-element"
reporting="true"/>
        </port-definition>
        <state-transition-element id="h1" symbol-set="1" start="start-of-data"
latch="true">
          <activate-on-match element="ch1"/>
        </state-transition-element>
        <state-transition-element id="h0" symbol-set="0" start="start-of-data"
latch="true">
          <activate-on-match element="ch0"/>
        </state-transition-element>
        <counter countone="ch1" reset="rh1" target="6" at_target="pulse">
          <activate-on-target element="11L"/>
          <activate-on-target element="10L"/>
        </counter>
        <counter countone="ch0" reset="rh0" target="6" at_target="pulse">
          <activate-on-target element="01L"/>
          <activate-on-target element="00L"/>
        </counter>
        <state-transition-element id="11L" symbol-set="1" latch="true">
          <activate-on-match element="c11"/>
        </state-transition-element>
        <state-transition-element id="10L" symbol-set="0" latch="true">
          <activate-on-match element="c10"/>
        </state-transition-element>
        <state-transition-element id="01L" symbol-set="1" latch="true">
          <activate-on-match element="c01"/>
        </state-transition-element>
        <state-transition-element id="00L" symbol-set="0" latch="true">
```

```
      <activate-on-match element="c00"/>
    </state-transition-element>
    <counter countone="c11" reset="r11" target="6" at_target="pulse">
      <report-on-target/>
    </counter>
    <counter countone="c10" reset="r10" target="7" at_target="pulse">
      <report-on-target/>
    </counter>
    <counter countone="c01" reset="r01" target="8" at_target="pulse">
      <report-on-target/>
    </counter>
    <counter countone="c00" reset="r00" target="9" at_target="pulse">
      <report-on-target/>
    </counter>
  </macro>
```



A Macro with input external connections but no output external connections

A Macro With Input External Connections but No Output External Connections

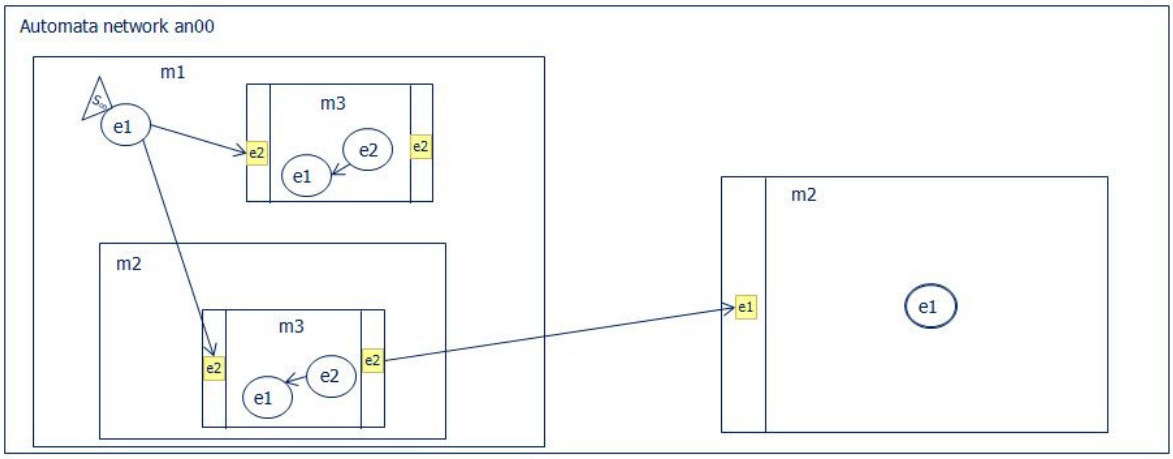
```
<macro name="radix key 2 final" id="mx05">
  <port-definition>
    <element-reference element="C11" type="state-transition-element" reporting="true"/>
    <element-reference element="C10" type="state-transition-element" reporting="true"/>
    <element-reference element="C01" type="state-transition-element" reporting="true"/>
    <element-reference element="C00" type="state-transition-element" reporting="true"/>
    <element-reference element="h1" type="state-transition-element" activation="in"/>
    <element-reference element="h0" type="state-transition-element" activation="in"/>
  </port-definition>
  <state-transition-element id="h1" symbol-set="1" latch="true">
    <activate-on-match element="ch1"/>
  </state-transition-element>
  <state-transition-element id="h0" symbol-set="0" latch="true">
    <activate-on-match element="ch0"/>
  </state-transition-element>
  <counter countone="ch1" reset="rh1" target="6" at_target="pulse">
    <activate-on-target element="C11"/>
    <activate-on-target element="10"/>
  </counter>
  <counter countone="ch0" reset="rh0" target="6" at_target="pulse">
    <activate-on-target element="01"/>
    <activate-on-target element="00"/>
  </counter>
  <state-transition-element id="C11" symbol-set="1">
    <report-on-match/>
  </state-transition-element>
  <state-transition-element id="10" symbol-set="0">
    <activate-on-match element="C10"/>
  </state-transition-element>
  <state-transition-element id="C10" symbol-set="*">
    <report-on-match/>
  </state-transition-element>
  <state-transition-element id="01" symbol-set="1">
    <activate-on-match element="s1"/>
  </state-transition-element>
  <state-transition-element id="s1" symbol-set="*">
    <activate-on-match element="C01"/>
  </state-transition-element>
  <state-transition-element id="C01" symbol-set="*">
    <report-on-match/>
  </state-transition-element>
  <state-transition-element id="00" symbol-set="0">
    <activate-on-match element="s2"/>
  </state-transition-element>
  <state-transition-element id="s2" symbol-set="*">
    <activate-on-match element="s3"/>
```

```
      </state-transition-element>
      <state-transition-element id="s3" symbol-set="*">
         <activate-on-match element="C00"/>
      </state-transition-element>
      <state-transition-element id="C00" symbol-set="*">
         <report-on-match/>
      </state-transition-element>
   </macro>
```

Missing Original ANML-G Representation

Macro With Incoming External Connections

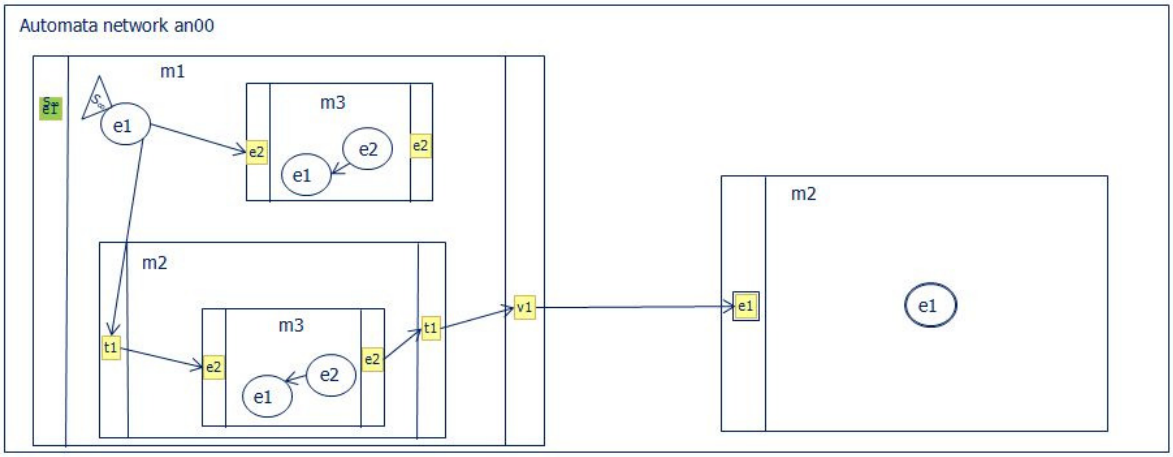
Proposals for new Macro structures



Click here to download the XML coding

```
<?xml version="1.0"?>
<automata-network name="string" id="an00" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="file:///c:/ANML/cookbook/ANML_v6.xsd">
   <macro name="string" id="m1">
      <state-transition-element id="e1" symbol-set="string" start="all-input">
         <activate-on-match element="m3.e2"/>
         <activate-on-match element="m2.m3.e2"/>
      </state-transition-element>
      <macro name="string" id="m2">
         <macro name="string" id="m3">
            <port-definition>
               <element-reference element="e2" type="inout">
                  <activate-on-match element="an00.m2.e1"/>
               </element-reference>
            </port-definition>
            <state-transition-element id="e1" symbol-set="string"/>
            <state-transition-element id="e2" symbol-set="string">
               <activate-on-match element="e1"/>
            </state-transition-element>
         </macro>
      </macro>
      <macro name="string" id="m3">
         <port-definition>
            <element-reference element="e2" type="inout"/>
         </port-definition>
         <state-transition-element id="e1" symbol-set="string"/>
         <state-transition-element id="e2" symbol-set="string">
            <activate-on-match element="e1"/>
         </state-transition-element>
      </macro>
   </macro>
   <macro name="string" id="m2">
      <port-definition>
         <element-reference element="e1" type="in"/>
      </port-definition>
      <state-transition-element id="e1" symbol-set="string">
         <report-on-match/>
      </state-transition-element>
   </macro>
</automata-network>
```

A



Click here to download the XML coding

```
<?xml version="1.0"?>
<automata-network name="Black Box Macro Example" id="an00" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="file:///c:/ANML/cookbook/ANML_v6.xsd">
   <macro name="string" id="m1">
      <port-definition>
         <element-reference id="S∞e1" element="e1" start="start-all-input"/>
         <element-reference id="v1" element="t1" activation="out">
            <activate-on-match element="e1"/>
         </element-reference>
      </port-definition>
      <state-transition-element id="e1" symbol-set="string" start="all-input">
         <activate-on-match element="e2"/>
         <activate-on-match element="t1"/>
      </state-transition-element>
      <macro name="string" id="m2">
         <port-definition>
            <element-reference id="t1" element="e2" activation="inout">
               <activate-on-match element="v1"/>
            </element-reference>
         </port-definition>
         <macro name="string" id="m3">
            <port-definition>
               <element-reference id="e2" element="e2" activation="inout">
                  <activate-on-match element="t1"/>
               </element-reference>
            </port-definition>
            <state-transition-element id="e1" symbol-set="string"/>
            <state-transition-element id="e2" symbol-set="string">
               <activate-on-match element="e1"/>
            </state-transition-element>
         </macro>
      </macro>
      <macro name="string" id="m3">
         <port-definition>
            <element-reference element="e2" activation="inout"/>
         </port-definition>
         <state-transition-element id="e1" symbol-set="string"/>
         <state-transition-element id="e2" symbol-set="string">
            <activate-on-match element="e1"/>
         </state-transition-element>
      </macro>
   </macro>
   <macro name="string" id="m2">
      <port-definition>
         <element-reference id="e1" element="e1" activation="in" reporting="true"/>
      </port-definition>
      <state-transition-element id="e1" symbol-set="string">
         <report-on-match/>
      </state-transition-element>
   </macro>
</automata-network>
```

Regex

Regex

Regex is an XML element which may be contained in macros. Regex will be interpreted by the compiler, resulting in the creation of an ANML automata. The details of regex automata are opaque in ANML, although the regex can be associated with an in, out or inout port in the macro port-definition through the regex id. Output of a regex (match) is also associated with the regex id in the output buffer.

The exact regex syntax supported is an implementation detail.

	attribute	type/value	occurence	example
<regex	id=	*id*	required unique in macro	"b1"
>				
regular expression		*string*		
</regex>				

Automata-Network

Automata-Network

automata-network contains zero or more automatons composed from macros and automata elements. It is the root element for an ANML network description.

Syntax

	attribute	type/value	occurence	example
<automata-network				
	name=	string	required	"an118"
	id=	id	required	"an1"
<macro			0 or more	
<*any automaton element*			0 or more	
</automata-network>				

Cookbook

Cookbook

This section show how to use ANML to implement various automata.

Copying Counter Values
Reverse Counter

Variation
Reverse Counter without a Counter Element

Multipying a Counter by 2

Implementing a Bit Stack with the 2X Counter

missing XOR

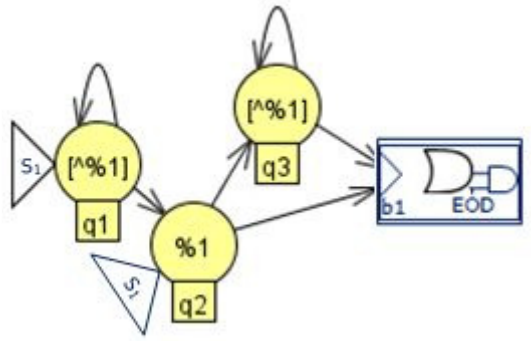
Short pattern recognition with mismatches using the Automata Processor as an accelerator
Fixed-length pattern recognition with serial-to-parallel shift-register match aggregation

Flagging Occurences of Exactly One Value in a Stream of Input

Flagging Occurences of Exactly One Value in a Stream of Input

This automaton reports when one and only occurence of a value is seen in a data stream delimited by start-of-data and end-of-data. The ANML and ANML-G representations show the macro template where a variable (%1) is used to indicate where substitution of the actual value desired should be made.

ANML-G Macro



ANML Template

```
<?xml version="1.0"?>
<automata-network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="file:///C:/ANML/cookbook/ANML_v6.xsd"
id="an001" name="0-255 One Value Counter">
   <macro id="m001-%1" name="Single 0-255 One Value Counter">
      <state-transition-element id="q1" start="start-of-data" symbol-set="[^%1]">
         <activate-on-match element="q1"/>
         <activate-on-match element="q2"/>
      </state-transition-element>
      <state-transition-element id="q2" symbol-set="%1" start="start-of-data">
         <activate-on-match element="q3"/>
         <activate-on-match element="b1"/>
      </state-transition-element>
      <state-transition-element id="q3" symbol-set="[^%1]">
         <activate-on-match element="q3"/>
         <activate-on-match element="b1"/>
      </state-transition-element>
      <or id="b1" high-only-on-eod="true">
         <report-on-high/>
      </or>
```

```
   </macro>
</automata-network>
```

An implementation of this circuit as 256 independent macro automatons within a logic-region was created using a perl script (missing).

This automaton detects 4 mutually exclusive conditions under which one and only one value %1 in a stream of symbols bounded by a single start-of-date (S_1) and an end-of-data (EOD) signal:

1. A single %1
2. A single %1 followed by one or more ^%1
3. One or more ^%1 followed by a single %1
4. One or more ^%1 followed by a single %1 followed by one or more %1

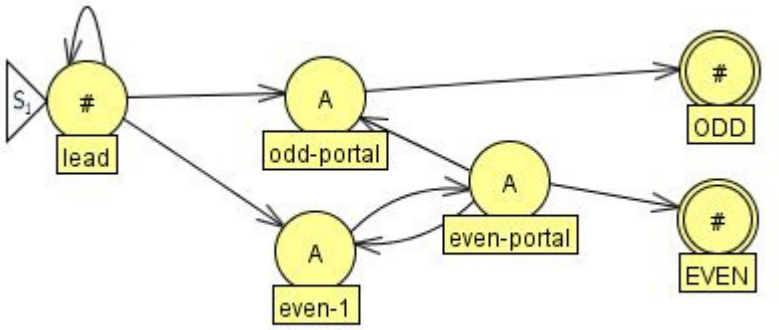
Any other situation will not produce an report-on-high match at b1.

Odd or Even Symbol Count

Odd or Even Symbol Count

This automaton assumes input in the form of one or more leading '#' followed by an unbroken sequence of 'A' followed by a symbol '#' terminating the sequence. It simply determines whether the number of 'A's is odd or even. If ODD a report event occurs on the ODD state-transition-element and if it is EVEN a report event occurs on the EVEN state-transition-element.

ANML-G Macro



ANML Template

```
<?xml version="1.0"?>
<macro name="odd or even symbol count" id="m01" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="file:///c:/ANML/cookbook/ANML_v4.xsd">
   <!-- input in the form of #+A+#+ -->
   <state-transition-element id="lead" symbol-set="#" start="start-of-data">
      <activate-on-match element="lead"/>
      <activate-on-match element="odd-portal"/>
      <activate-on-match element="even-1"/>
   </state-transition-element>
   <state-transition-element id="odd-portal" symbol-set="A">
      <activate-on-match element="ODD"/>
   </state-transition-element>
   <state-transition-element id="even-1" symbol-set="A">
      <activate-on-match element="even-portal"/>
   </state-transition-element>
   <state-transition-element id="even-portal" symbol-set="A">
      <activate-on-match element="even-1"/>
      <activate-on-match element="EVEN"/>
      <activate-on-match element="odd-portal"/>
   </state-transition-element>
   <state-transition-element id="EVEN" symbol-set="#">
      <report-on-match/>
   </state-transition-element>
```

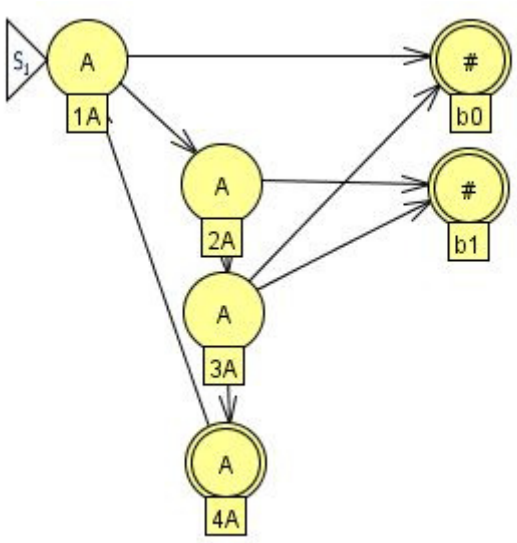
```
    <state-transition-element id="ODD" symbol-set="#">
            <report-on-match/>
    </state-transition-element>
</macro>
```

Counter with 2 bit Display and Carryover

Counter with 2-bit Display and Carryover

This example counts the number of 'A's in a stream terminated by a '#', reporting the result up to 3 in binary and reporting an output event each time the count rolls over at 4. The binary count is represented by simultaneous report events on b0 and b1 when the terminating '#' is seen and a count rollover by a report event on 4A each time the count reaches the rollover value.

ANML-G Macro



ANML Template

```
<?xml version="1.0"?>
<macro name="counter with 2-bit display" id="m01" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="file:///c:/ANML/cookbook/ANML_v4.xsd">
   <!-- input in the form of A+# -->
   <state-transition-element id="1A" symbol-set="A" start="start-of-data">
      <activate-on-match element="2A"/>
      <activate-on-match element="b0"/>
   </state-transition-element>
   <state-transition-element id="2A" symbol-set="A">
      <activate-on-match element="3A"/>
      <activate-on-match element="b1"/>
   </state-transition-element>
   <state-transition-element id="3A" symbol-set="A">
      <activate-on-match element="4A"/>
      <activate-on-match element="b0"/>
      <activate-on-match element="b1"/>
   </state-transition-element>
   <state-transition-element id="4A" symbol-set="A">
```

```
      <report-on-match/>
      <activate-on-match element="1A"/>
   </state-transition-element>
   <state-transition-element id="b0" symbol-set="#">
      <report-on-match/>
   </state-transition-element>
   <state-transition-element id="b1" symbol-set="#">
      <report-on-match/>
   </state-transition-element>
</macro>
```

Here is a table showing counting from 1 to 12 'A's and report events.

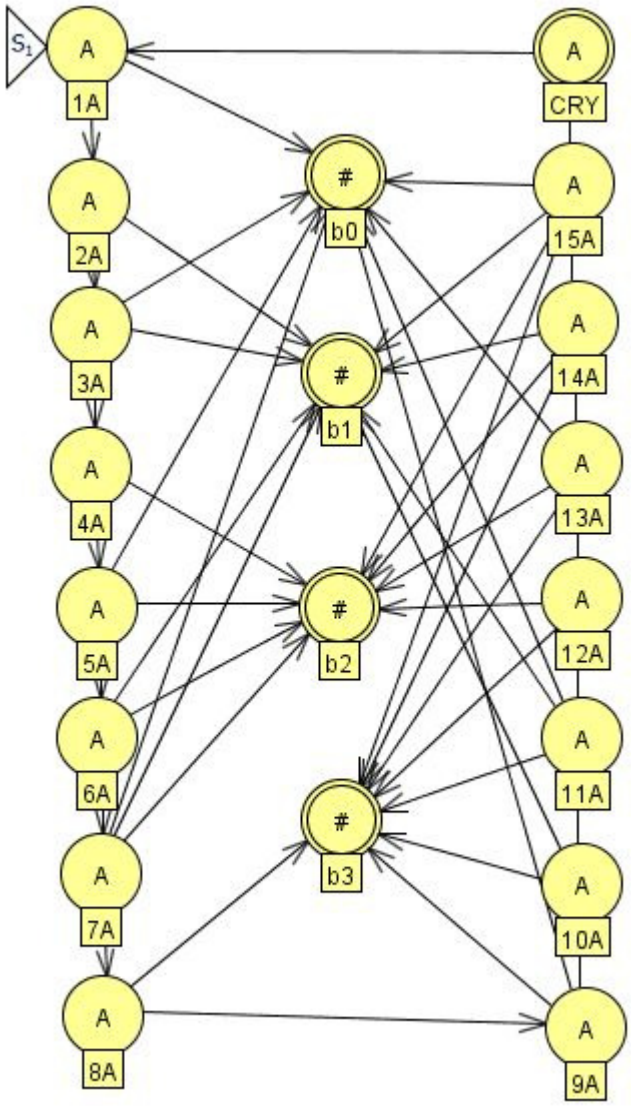
stream	report events
A#	b0 at #
AA#	b1 at #
AAA#	b0 b1 at #
AAAA#	4A at 4th A
AAAAA#	4A at 4th A b0 at #
AAAAAA#	4A at 4th A b1 at #
AAAAAAA#	4A at 4th A b0 b1 at #
AAAAAAAA#	4A at 4th A 4A at 8th A
AAAAAAAAA#	4A at 4th A 4A at 8th A b0 at #
AAAAAAAAAA#	4A at 4th A 4A at 8th A b1 at #
AAAAAAAAAAA#	4A at 4th A 4A at 8th A b0 b1 at #
AAAAAAAAAAAA#	4A at 4th A 4A at 8th A 4A at 12th A

Counter with 4 bit Display and Carryover

Counter with 4-bit Display and Carryover

The operation of this automaton is identical to the Counter with 2-bit Display and Carryover described in the previous article, except that there are 4 bits of display for a base count up to 15 and a carryover match at each interval of 16.

ANML-G Macro



ANML Template

```
<?xml version="1.0"?>
<macro name="counter with 4-bit display" id="m01"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="file:///c:/ANML/cookbook/ANML_v4.xsd">
   <!-- input in the form of A+# -->
   <state-transition-element id="1A" symbol-set="A" start="start-of-data">
      <activate-on-match element="2A"/>
      <activate-on-match element="b0"/>
   </state-transition-element>
   <state-transition-element id="2A" symbol-set="A">
      <activate-on-match element="3A"/>
      <activate-on-match element="b1"/>
   </state-transition-element>
   <state-transition-element id="3A" symbol-set="A">
      <activate-on-match element="4A"/>
      <activate-on-match element="b0"/>
      <activate-on-match element="b1"/>
   </state-transition-element>
   <state-transition-element id="4A" symbol-set="A">
      <activate-on-match element="5A"/>
      <activate-on-match element="b2"/>
   </state-transition-element>
   <state-transition-element id="5A" symbol-set="A">
      <activate-on-match element="6A"/>
      <activate-on-match element="b0"/>
      <activate-on-match element="b2"/>
   </state-transition-element>
   <state-transition-element id="6A" symbol-set="A">
      <activate-on-match element="7A"/>
      <activate-on-match element="b1"/>
      <activate-on-match element="b2"/>
   </state-transition-element>
   <state-transition-element id="7A" symbol-set="A">
      <activate-on-match element="8A"/>
      <activate-on-match element="b0"/>
      <activate-on-match element="b1"/>
      <activate-on-match element="b2"/>
   </state-transition-element>
   <state-transition-element id="8A" symbol-set="A">
      <activate-on-match element="9A"/>
      <activate-on-match element="b3"/>
   </state-transition-element>
   <state-transition-element id="9A" symbol-set="A">
      <activate-on-match element="10A"/>
      <activate-on-match element="b0"/>
      <activate-on-match element="b3"/>
   </state-transition-element>
   <state-transition-element id="10A" symbol-set="A">
      <activate-on-match element="11A"/>
      <activate-on-match element="b1"/>
      <activate-on-match element="b3"/>
```

```
  </state-transition-element>
  <state-transition-element id="11A" symbol-set="A">
    <activate-on-match element="12A"/>
    <activate-on-match element="b0"/>
    <activate-on-match element="b1"/>
    <activate-on-match element="b3"/>
  </state-transition-element>
  <state-transition-element id="12A" symbol-set="A">
    <activate-on-match element="13A"/>
    <activate-on-match element="b2"/>
    <activate-on-match element="b3"/>
  </state-transition-element>
  <state-transition-element id="13A" symbol-set="A">
    <activate-on-match element="14A"/>
    <activate-on-match element="b0"/>
    <activate-on-match element="b2"/>
    <activate-on-match element="b3"/>
  </state-transition-element>
  <state-transition-element id="14A" symbol-set="A">
    <activate-on-match element="15A"/>
    <activate-on-match element="b1"/>
    <activate-on-match element="b2"/>
    <activate-on-match element="b3"/>
  </state-transition-element>
  <state-transition-element id="15A" symbol-set="A">
    <activate-on-match element="CRY"/>
    <activate-on-match element="b0"/>
    <activate-on-match element="b1"/>
    <activate-on-match element="b2"/>
    <activate-on-match element="b3"/>
  </state-transition-element>
  <state-transition-element id="CRY" symbol-set="A">
    <report-on-match/>
    <activate-on-match element="1A"/>
  </state-transition-element>
  <state-transition-element id="b0" symbol-set="#">
    <report-on-match/>
  </state-transition-element>
  <state-transition-element id="b1" symbol-set="#">
    <report-on-match/>
  </state-transition-element>
  <state-transition-element id="b2" symbol-set="#">
    <report-on-match/>
  </state-transition-element>
  <state-transition-element id="b3" symbol-set="#">
     <report-on-match/>
  </state-transition-element>
</macro>
```

Counter of Non-continguous Symbols with 4 bit Display and Carryover

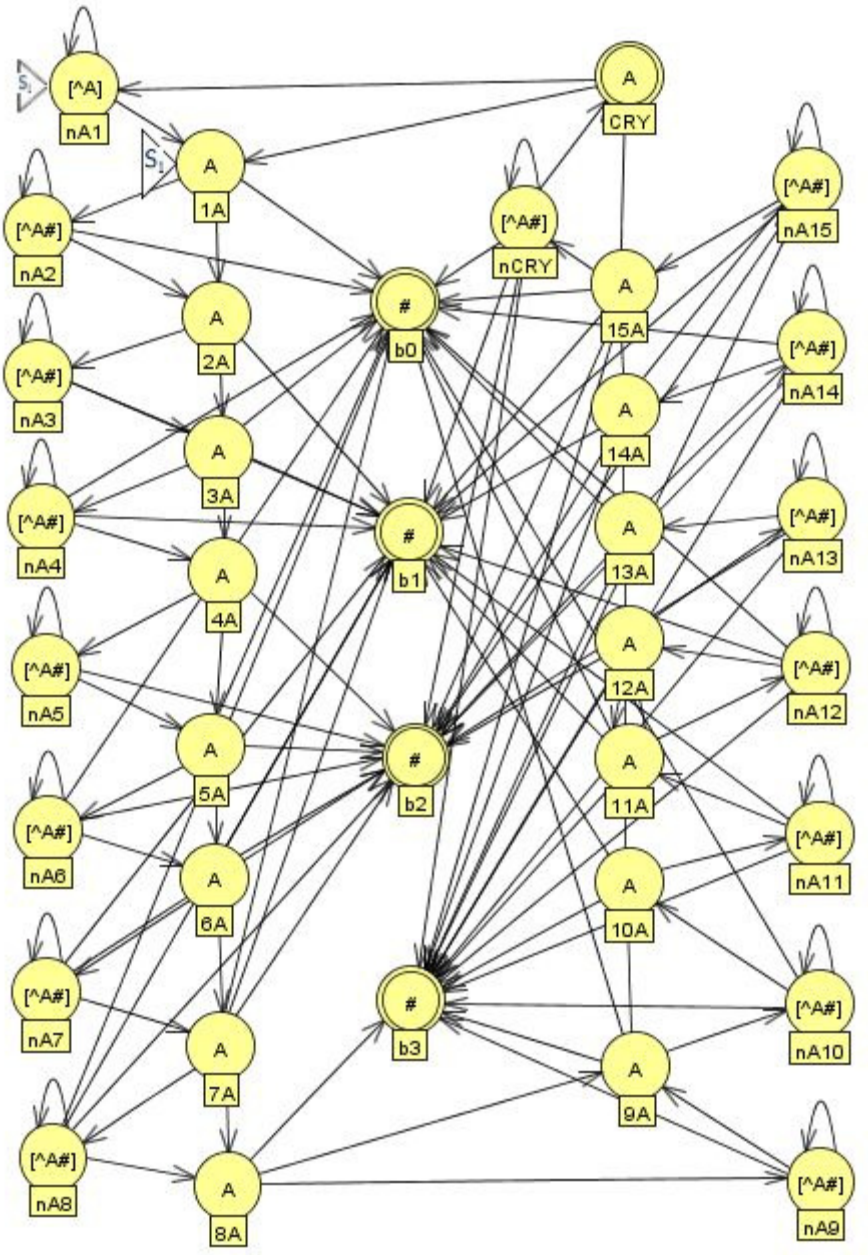
Counter of Non-continguous Symbols with 4 bit Display and Carryover

This example is an extension of the previous counter. This counter allows any symbol to appear in the stream but only counts the target symbol, 'A' in this implementation. For example,

tAté"è'è'A0@²#~{[A#

will display a count of 3.

ANML-G Macro



ANML Template

It may be inconvenient that this automaton uses '#' as the signal to report the count. One alternate way to implement this function is use AND elements with EOD enabled in place of the state-transition-elements b0-b3. A potential advantage of using the '#' is that it can be used as a separator between streams. In this case additional logic will be needed to restart processing after the display event.

```
<?xml version="1.0" ?>
- <macro name="counter (non-contiguous symbols)with 4-bit display" id="m01"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="file:///c:/ANML/cookbook/ANML_v4.xsd"> <!--
  input in the form of [^#]+#  -->
  - <state-transition-element id="1A" symbol-set="A" start="start-of-data">
      <activate-on-match element="2A" />
      <activate-on-match element="b0" />
    </state-transition-element>
  - <state-transition-element id="2A" symbol-set="A">
      <activate-on-match element="3A" />
      <activate-on-match element="b1" />
    </state-transition-element>
  - <state-transition-element id="3A" symbol-set="A">
      <activate-on-match element="4A" />
      <activate-on-match element="b0" />
      <activate-on-match element="b1" />
    </state-transition-element>
  - <state-transition-element id="4A" symbol-set="A">
      <activate-on-match element="5A" />
      <activate-on-match element="b2" />
    </state-transition-element>
  - <state-transition-element id="5A" symbol-set="A">
      <activate-on-match element="6A" />
      <activate-on-match element="b0" />
      <activate-on-match element="b2" />
    </state-transition-element>
  - <state-transition-element id="6A" symbol-set="A">
      <activate-on-match element="7A" />
      <activate-on-match element="b1" />
      <activate-on-match element="b2" />
    </state-transition-element>
  - <state-transition-element id="7A" symbol-set="A">
      <activate-on-match element="8A" />
      <activate-on-match element="b0" />
      <activate-on-match element="b1" />
      <activate-on-match element="b2" />
    </state-transition-element>
  - <state-transition-element id="8A" symbol-set="A">
      <activate-on-match element="9A" />
      <activate-on-match element="b3" />
    </state-transition-element>
  - <state-transition-element id="9A" symbol-set="A">
      <activate-on-match element="10A" />
      <activate-on-match element="b0" />
      <activate-on-match element="b3" />
    </state-transition-element>
  - <state-transition-element id="10A" symbol-set="A">
      <activate-on-match element="11A" />
      <activate-on-match element="b1" />
      <activate-on-match element="b3" />
    </state-transition-element>
  - <state-transition-element id="11A" symbol-set="A">
      <activate-on-match element="12A" />
      <activate-on-match element="b0" />
      <activate-on-match element="b1" />
```

```
    <activate-on-match element="b3" />
  </state-transition-element>
- <state-transition-element id="12A" symbol-set="A">
    <activate-on-match element="13A" />
    <activate-on-match element="b2" />
    <activate-on-match element="b3" />
  </state-transition-element>
- <state-transition-element id="13A" symbol-set="A">
    <activate-on-match element="14A" />
    <activate-on-match element="b0" />
    <activate-on-match element="b2" />
    <activate-on-match element="b3" />
  </state-transition-element>
- <state-transition-element id="14A" symbol-set="A">
    <activate-on-match element="15A" />
    <activate-on-match element="b1" />
    <activate-on-match element="b2" />
    <activate-on-match element="b3" />
  </state-transition-element>
- <state-transition-element id="15A" symbol-set="A">
    <activate-on-match element="CRY" />
    <activate-on-match element="b0" />
    <activate-on-match element="b1" />
    <activate-on-match element="b2" />
    <activate-on-match element="b3" />
  </state-transition-element>
- <state-transition-element id="CRY" symbol-set="A">
    <report-on-match />
    <activate-on-match element="1A" />
  </state-transition-element>
  <!-- Recognize not target symbol characters not counted  -->
- <state-transition-element id="nA1" symbol-set="[^A]">
    <activate-on-match element="nA1" />
    <activate-on-match element="1A" />
  </state-transition-element>
- <state-transition-element id="nA2" symbol-set="[^A#]">
    <activate-on-match element="nA2" />
    <activate-on-match element="2A" />
    <activate-on-match element="b0" />
  </state-transition-element>
- <state-transition-element id="nA3" symbol-set="[^A#]">
    <activate-on-match element="nA3" />
    <activate-on-match element="3A" />
    <activate-on-match element="b1" />
  </state-transition-element>
- <state-transition-element id="nA4" symbol-set="[^A#]">
    <activate-on-match element="nA4" />
    <activate-on-match element="4A" />
    <activate-on-match element="b0" />
    <activate-on-match element="b1" />
  </state-transition-element>
- <state-transition-element id="nA5" symbol-set="[^A#]">
    <activate-on-match element="nA5" />
    <activate-on-match element="5A" />
    <activate-on-match element="b2" />
  </state-transition-element>
```

```
- <state-transition-element id="nA6" symbol-set="[^A#]">
    <activate-on-match element="nA6" />
    <activate-on-match element="6A" />
    <activate-on-match element="b0" />
    <activate-on-match element="b2" />
  </state-transition-element>
- <state-transition-element id="nA7" symbol-set="[^A#]">
    <activate-on-match element="nA7" />
    <activate-on-match element="7A" />
    <activate-on-match element="b1" />
    <activate-on-match element="b2" />
  </state-transition-element>
- <state-transition-element id="nA8" symbol-set="[^A#]">
    <activate-on-match element="nA8" />
    <activate-on-match element="8A" />
    <activate-on-match element="b0" />
    <activate-on-match element="b1" />
    <activate-on-match element="b2" />
  </state-transition-element>
- <state-transition-element id="nA9" symbol-set="[^A#]">
    <activate-on-match element="nA9" />
    <activate-on-match element="9A" />
    <activate-on-match element="b3" />
  </state-transition-element>
- <state-transition-element id="nA10" symbol-set="[^A#]">
    <activate-on-match element="nA10" />
    <activate-on-match element="10A" />
    <activate-on-match element="b0" />
    <activate-on-match element="b3" />
  </state-transition-element>
- <state-transition-element id="nA11" symbol-set="[^A#]">
    <activate-on-match element="nA11" />
    <activate-on-match element="11A" />
    <activate-on-match element="b1" />
    <activate-on-match element="b3" />
  </state-transition-element>
- <state-transition-element id="nA12" symbol-set="[^A#]">
    <activate-on-match element="nA12" />
    <activate-on-match element="12A" />
    <activate-on-match element="b0" />
    <activate-on-match element="b1" />
    <activate-on-match element="b3" />
  </state-transition-element>
- <state-transition-element id="nA13" symbol-set="[^A#]">
    <activate-on-match element="nA13" />
    <activate-on-match element="13A" />
    <activate-on-match element="b2" />
    <activate-on-match element="b3" />
  </state-transition-element>
- <state-transition-element id="nA14" symbol-set="[^A#]">
    <activate-on-match element="nA14" />
    <activate-on-match element="14A" />
    <activate-on-match element="b0" />
    <activate-on-match element="b2" />
    <activate-on-match element="b3" />
  </state-transition-element>
```

```
- <state-transition-element id="nA15" symbol-set="[^A#]">
    <activate-on-match element="nA15" />
    <activate-on-match element="15A" />
    <activate-on-match element="b1" />
    <activate-on-match element="b2" />
    <activate-on-match element="b3" />
  </state-transition-element>
- <state-transition-element id="nCRY" symbol-set="[^A#]">
    <activate-on-match element="CRY" />
    <activate-on-match element="nCRY" />
    <activate-on-match element="b0" />
    <activate-on-match element="b1" />
    <activate-on-match element="b2" />
    <activate-on-match element="b3" />
  </state-transition-element>
  <!--  End of stream marker and output final count   -->
- <state-transition-element id="b0" symbol-set="#">
    <report-on-match />
  </state-transition-element>
- <state-transition-element id="b1" symbol-set="#">
    <report-on-match />
  </state-transition-element>
- <state-transition-element id="b2" symbol-set="#">
    <report-on-match />
  </state-transition-element>
- <state-transition-element id="b3" symbol-set="#">
    <report-on-match />
  </state-transition-element>
</macro>
```

Radix Sort on 2 Keys

Radix Sort on 2 Keys

This automaton does a 2 key radix sort (see wikipedia article). It takes a stream of binary symbols ('0' or '1') representing the keys and reports the sort order by bucket group. For example, the set of values 0, 1, 3, 0, 2, 3 (in binary symbols 00, 01, 11, 00, 10, 11) is presented to the automaton as the stream

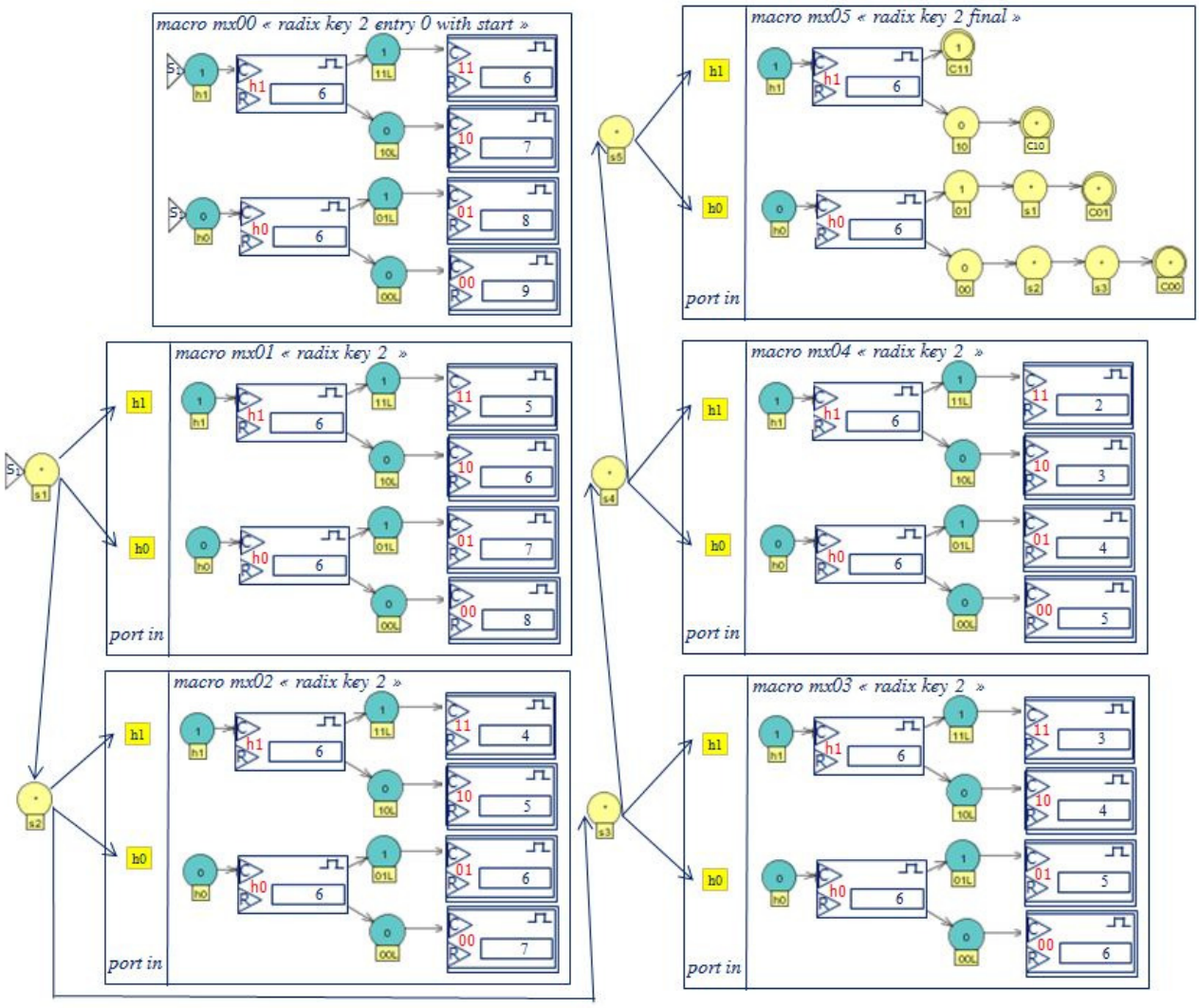
001011011001###

taking the first binary symbol from each value for the first six symbols followed by the second binary symbol from each value for the next six symbols. Three extra characters are added to flush the results on the last symbol cycle. The results are reported in the last 4 symbol cycles corresponding the four possible buckets of a 2 bit value:

final cycle minus 3: report third value (3)
final cycle minus 2: report fifth value (2)
final cycle minus 1: report second value (1)
final cycle: report first and fourth values (0)

One macro is needed for each possible input value. In the example below an automaton has been created for a stream of six values so six macros are used.

ANML-G Macro



ANML Template

A 2 key radix sort has 4 possible 'buckets', i.e., 0, 1, 2 or 3. This automaton simply identifies which bucket each individual value goes in and uses counters to "time" the output of each number into the cycle in which all values which go into the bucket are output. There is nothing in the design of the automaton to ensure that the sorting is stable, although some automata network implementions may produce a stable sort order.

One macro is used for each value being sorted. One way to use this architecture is to place and route a circuit for a largest number of values an implementation can support and to reconfigure the counter values for the number of values in each dataset. This design can be expanded to support greater number of keys following the same design principle used in this example.

The operation of the example automaton is traced through in the table below:

input stream, decimal: 0, 1, 3, 0, 2, 3
input stream, binary: 00 01 11 00 10 11
input stream, radix (MSD): 0 0 1 0 1 1 0 1 1 0 0 1 # # #

cycle: 1 input: 0 start-of-data

active states	event	activate	count	output
mx00 h1				
mx00 h0	match	mx00 ch0	mx00 ch0 1/6	
an01 s1	match	mx01 h1 mx01 h0 an01 s2		

cycle: 2 input: 0

active states	event	activate	count	output
mx00 h0	latch	mx00 ch0	mx00 ch0 2/6	
mx01 h1				
mx01 h0	match	mx01 ch0	mx01 ch0 1/6	
an01 s2	match	mx02 h1 mx02 h0 an01 s3		

cycle: 3 input: 1

active states	event	activate	count	output
mx00 h0	latch	mx00 ch0	mx00 ch0 3/6	
mx01 h0	latch	mx01 ch0	mx01 ch0 2/6	
mx02 h1	match	mx02 ch1	mx02 ch1 1/6	
mx02 h0				
an01 s3	match	mx03 h1 mx03 h0 an01 s4		

cycle: 4 input: 0

active states	event	activate	count	output
mx00 h0	latch	mx00 ch0	mx00 ch0 4/6	
mx01 h0	latch	mx01 ch0	mx01 ch0 3/6	
mx02 h1	latch	mx02 ch1	mx02 ch1 2/6	
mx03 h1				
mx03 h0	match	mx03 ch0	mx03 ch0 1/6	
an01 s4	match	mx04 h1 mx04 h0 an01 s5		

cycle: 5 input: 1

active states	event	activate	count	output
mx00 h0	latch	mx00 ch0	mx00 ch0 5/6	
mx01 h0	latch	mx01 ch0	mx01 ch0 4/6	
mx02 h1	latch	mx02 ch1	mx02 ch1 3/6	
mx03 h0	latch	mx03 ch0	mx03 ch0 2/6	
mx04 h1	match	mx04 ch1	mx04 ch1 1/6	
mx04 h0				
an01 s5	match	mx05 h1 mx05 h0		

cycle: 6 input: 1

active states	event	activate	count	output
mx00 h0	latch	mx00 ch0	mx00 ch0 6/6	
mx01 h0	latch	mx01 ch0	mx01 ch0 5/6	
mx02 h1	latch	mx02 ch1	mx02 ch1 4/6	
mx03 h0	latch	mx03 ch0	mx03 ch0 3/6	
mx04 h1	latch	mx04 ch1	mx04 ch1 2/6	
mx05 h1	match	mx05 ch1	mx05 ch1 1/6	
mx05 h0				
mx00 ch0	target	mx00 01L mx00 00L		

cycle: 7 input: 0

active states	event	activate	count	output
mx01 h0	latch	mx01 ch0	mx01 ch0 6/6	
mx02 h1	latch	mx02 ch1	mx02 ch1 5/6	
mx03 h0	latch	mx03 ch0	mx03 ch0 4/6	
mx04 h1	latch	mx04 ch1	mx04 ch1 3/6	
mx05 h1	latch	mx05 ch1	mx05 ch1 2/6	
mx00 01L				
mx00 00L	match	mx00 c00	mx00 c00 1/9	
mx01 ch0	target	mx01 01L mx01 00L		

cycle: 8 input: 1

active states	event	activate	count	output
mx02 h1	latch	mx02 ch1	mx02 ch1 6/6	
mx03 h0	latch	mx03 ch0	mx03 ch0 5/6	
mx04 h1	latch	mx04 ch1	mx04 ch1 4/6	
mx05 h1	latch	mx05 ch1	mx05 ch1 3/6	
mx00 00L	latch	mx00 c00	mx00 c00 2/9	
mx01 01L	match	mx01 c01	mx01 c01 1/7	
mx01 00L				
mx02 ch1	target	mx02 11L mx02 10L		

cycle: 9 input: 1

active states	event	activate	count	output
mx03 h0	latch	mx03 ch0	mx03 ch0 6/6	
mx04 h1	latch	mx04 ch1	mx04 ch1 5/6	
mx05 h1	latch	mx05 ch1	mx05 ch1 4/6	
mx00 00L	latch	mx00 c00	mx00 c00 3/9	
mx01 01L	latch	mx01 c01	mx01 c01 2/7	
mx02 11L	match	mx02 c11	mx02 c11 1/4	
mx02 10L				
mx03 ch0	target	mx03 01L mx03 00L		

cycle: 10 input: 0

active states	event	activate	count	output
mx04 h1	latch	mx04 ch1	mx04 ch1 6/6	
mx05 h1	latch	mx05 ch1	mx05 ch1 5/6	
mx00 00L	latch	mx00 c00	mx00 c00 4/9	
mx01 01L	latch	mx01 c01	mx01 c01 3/7	
mx02 11L	latch	mx02 c11	mx02 c11 2/4	
mx03 01L				
mx03 00L	match	mx03 c00	mx03 c00 1/6	
mx04 ch1	target	mx04 11L mx04 10L		

cycle: 11 input: 0

active states	event	activate	count	output
mx05 h1	latch	mx05 ch1	mx05 ch1 6/6	
mx00 00L	latch	mx00 c00	mx00 c00 5/9	
mx01 01L	latch	mx01 c01	mx01 c01 4/7	
mx02 11L	latch	mx02 c11	mx02 c11 3/4	
mx03 00L	latch	mx03 c00	mx03 c00 2/6	
mx04 11L				
mx04 10L	match	mx04 c10	mx04 c10 1/3	
mx05 ch1	target	mx05 C11 mx05 10		

cycle: 12 input: 1

active states	event	activate	count	output
mx00 00L	latch	mx00 c00	mx00 c00 6/9	
mx01 01L	latch	mx01 c01	mx01 c01 5/7	
mx02 11L	latch	mx02 c11	mx02 c11 4/4	mx02 c11
mx03 00L	latch	mx03 c00	mx03 c00 3/6	
mx04 10L	latch	mx04 c10	mx04 c10 2/3	
mx05 C11	match			mx05 C11
mx05 10				

cycle: 13 input: #

active states	event	activate	count	output
mx00 00L	latch	mx00 c00	mx00 c00 7/9	
mx01 01L	latch	mx01 c01	mx01 c01 6/7	
mx03 00L	latch	mx03 c00	mx03 c00 4/6	
mx04 10L	latch	mx04 c10	mx04 c10 3/3	mx04 c10

cycle: 14 input: #

active states	event	activate	count	output
mx00 00L	latch	mx00 c00	mx00 c00 8/9	
mx01 01L	latch	mx01 c01	mx01 c01 7/7	mx01 c01
mx03 00L	latch	mx03 c00	mx03 c01 5/6	

cycle: 15 input: #

active states	event	activate	count	output
mx00 00L	latch	mx00 c00	mx00 c00 9/9	mx c00
mx03 00L	latch	mx03 c00	mx03 c00 6/6	mx03 c00

```
<?xml version="1.0" ?>
- <automata-network name="an-radixsort-keys2elements6" id="an01"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="file:///c:/ANML/cookbook/ANML_v6.xsd">
  - <state-transition-element id="s1" symbol-set="*" start="start-of-data">
      <activate-on-match macro="mx01" element="h1" />
      <activate-on-match macro="mx01" element="h0" />
      <activate-on-match element="s2" />
    </state-transition-element>
  - <state-transition-element id="s2" symbol-set="*">
      <activate-on-match macro="mx02" element="h1" />
      <activate-on-match macro="mx02" element="h0" />
      <activate-on-match element="s3" />
    </state-transition-element>
  - <state-transition-element id="s3" symbol-set="*">
      <activate-on-match macro="mx03" element="h1" />
      <activate-on-match macro="mx03" element="h0" />
      <activate-on-match element="s4" />
    </state-transition-element>
  - <state-transition-element id="s4" symbol-set="*">
      <activate-on-match macro="mx04" element="h1" />
      <activate-on-match macro="mx04" element="h0" />
      <activate-on-match element="s5" />
    </state-transition-element>
  - <state-transition-element id="s5" symbol-set="*">
      <activate-on-match macro="mx05" element="h1" />
      <activate-on-match macro="mx05" element="h0" />
    </state-transition-element>
  - <macro name="radix key 2 entry 0 with start" id="mx00">
    - <port-definition>
        <element-reference element="h1" type="state-transition-element" start="start-of-
          data" />
        <element-reference element="h0" type="state-transition-element" start="start-of-
          data" />
        <element-reference element="c11" type="state-transition-element"
          reporting="true" />
        <element-reference element="c10" type="state-transition-element"
          reporting="true" />
        <element-reference element="c01" type="state-transition-element"
          reporting="true" />
        <element-reference element="c00" type="state-transition-element"
          reporting="true" />
      </port-definition>
    - <state-transition-element id="h1" symbol-set="1" start="start-of-data" latch="true">
        <activate-on-match element="ch1" />
      </state-transition-element>
    - <state-transition-element id="h0" symbol-set="0" start="start-of-data" latch="true">
        <activate-on-match element="ch0" />
      </state-transition-element>
    - <counter countone="ch1" reset="rh1" target="6" at_target="pulse">
        <activate-on-target element="11L" />
        <activate-on-target element="10L" />
      </counter>
    - <counter countone="ch0" reset="rh0" target="6" at_target="pulse">
        <activate-on-target element="01L" />
        <activate-on-target element="00L" />
```

```
    </counter>
  - <state-transition-element id="11L" symbol-set="1" latch="true">
      <activate-on-match element="c11" />
    </state-transition-element>
  - <state-transition-element id="10L" symbol-set="0" latch="true">
      <activate-on-match element="c10" />
    </state-transition-element>
  - <state-transition-element id="01L" symbol-set="1" latch="true">
      <activate-on-match element="c01" />
    </state-transition-element>
  - <state-transition-element id="00L" symbol-set="0" latch="true">
      <activate-on-match element="c00" />
    </state-transition-element>
  - <counter countone="c11" reset="r11" target="6" at_target="pulse">
      <report-on-target />
    </counter>
  - <counter countone="c10" reset="r10" target="7" at_target="pulse">
      <report-on-target />
    </counter>
  - <counter countone="c01" reset="r01" target="8" at_target="pulse">
      <report-on-target />
    </counter>
  - <counter countone="c00" reset="r00" target="9" at_target="pulse">
      <report-on-target />
    </counter>
  </macro>
- <macro name="radix key 2" id="mx01">
  - <port-definition>
      <element-reference element="h1" type="state-transition-element"
        activation="in" />
      <element-reference element="h0" type="state-transition-element"
        activation="in" />
      <element-reference element="c11" type="state-transition-element"
        reporting="true" />
      <element-reference element="c10" type="state-transition-element"
        reporting="true" />
      <element-reference element="c01" type="state-transition-element"
        reporting="true" />
      <element-reference element="c00" type="state-transition-element"
        reporting="true" />
    </port-definition>
  - <state-transition-element id="h1" symbol-set="1" latch="true">
      <activate-on-match element="ch1" />
    </state-transition-element>
  - <state-transition-element id="h0" symbol-set="0" latch="true">
      <activate-on-match element="ch0" />
    </state-transition-element>
  - <counter countone="ch1" reset="rh1" target="6" at_target="pulse">
      <activate-on-target element="11L" />
      <activate-on-target element="10L" />
    </counter>
  - <counter countone="ch0" reset="rh0" target="6" at_target="pulse">
      <activate-on-target element="01L" />
      <activate-on-target element="00L" />
    </counter>
  - <state-transition-element id="11L" symbol-set="1" latch="true">
```

```
    <activate-on-match element="c11" />
  </state-transition-element>
- <state-transition-element id="10L" symbol-set="0" latch="true">
    <activate-on-match element="c10" />
  </state-transition-element>
- <state-transition-element id="01L" symbol-set="1" latch="true">
    <activate-on-match element="c01" />
  </state-transition-element>
- <state-transition-element id="00L" symbol-set="0" latch="true">
    <activate-on-match element="c00" />
  </state-transition-element>
- <counter countone="c11" reset="r11" target="5" at_target="pulse">
    <report-on-target />
  </counter>
- <counter countone="c10" reset="r10" target="6" at_target="pulse">
    <report-on-target />
  </counter>
- <counter countone="c01" reset="r01" target="7" at_target="pulse">
    <report-on-target />
  </counter>
- <counter countone="c00" reset="r00" target="8" at_target="pulse">
    <report-on-target />
  </counter>
</macro>
- <macro name="radix key 2" id="mx02">
- <port-definition>
    <element-reference element="h1" type="state-transition-element"
      activation="in" />
    <element-reference element="h0" type="state-transition-element"
      activation="in" />
    <element-reference element="c11" type="state-transition-element"
      reporting="true" />
    <element-reference element="c10" type="state-transition-element"
      reporting="true" />
    <element-reference element="c01" type="state-transition-element"
      reporting="true" />
    <element-reference element="c00" type="state-transition-element"
      reporting="true" />
  </port-definition>
- <state-transition-element id="h1" symbol-set="1" latch="true">
    <activate-on-match element="ch1" />
  </state-transition-element>
- <state-transition-element id="h0" symbol-set="0" latch="true">
    <activate-on-match element="ch0" />
  </state-transition-element>
- <counter countone="ch1" reset="rh1" target="6" at_target="pulse">
    <activate-on-target element="11L" />
    <activate-on-target element="10L" />
  </counter>
- <counter countone="ch0" reset="rh0" target="6" at_target="pulse">
    <activate-on-target element="01L" />
    <activate-on-target element="00L" />
  </counter>
- <state-transition-element id="11L" symbol-set="1" latch="true">
    <activate-on-match element="c11" />
  </state-transition-element>
```

```
- <state-transition-element id="10L" symbol-set="0" latch="true">
    <activate-on-match element="c10" />
  </state-transition-element>
- <state-transition-element id="01L" symbol-set="1" latch="true">
    <activate-on-match element="c01" />
  </state-transition-element>
- <state-transition-element id="00L" symbol-set="0" latch="true">
    <activate-on-match element="c00" />
  </state-transition-element>
- <counter countone="c11" reset="r11" target="4" at_target="pulse">
    <report-on-target />
  </counter>
- <counter countone="c10" reset="r10" target="5" at_target="pulse">
    <report-on-target />
  </counter>
- <counter countone="c01" reset="r01" target="6" at_target="pulse">
    <report-on-target />
  </counter>
- <counter countone="c00" reset="r00" target="7" at_target="pulse">
    <report-on-target />
  </counter>
</macro>
- <macro name="radix key 2" id="mx03">
- <port-definition>
    <element-reference element="h1" type="state-transition-element"
      activation="in" />
    <element-reference element="h0" type="state-transition-element"
      activation="in" />
    <element-reference element="c11" type="state-transition-element"
      reporting="true" />
    <element-reference element="c10" type="state-transition-element"
      reporting="true" />
    <element-reference element="c01" type="state-transition-element"
      reporting="true" />
    <element-reference element="c00" type="state-transition-element"
      reporting="true" />
  </port-definition>
- <state-transition-element id="h1" symbol-set="1" latch="true">
    <activate-on-match element="ch1" />
  </state-transition-element>
- <state-transition-element id="h0" symbol-set="0" latch="true">
    <activate-on-match element="ch0" />
  </state-transition-element>
- <counter countone="ch1" reset="rh1" target="6" at_target="pulse">
    <activate-on-target element="11L" />
    <activate-on-target element="10L" />
  </counter>
- <counter countone="ch0" reset="rh0" target="6" at_target="pulse">
    <activate-on-target element="01L" />
    <activate-on-target element="00L" />
  </counter>
- <state-transition-element id="11L" symbol-set="1" latch="true">
    <activate-on-match element="c11" />
  </state-transition-element>
- <state-transition-element id="10L" symbol-set="0" latch="true">
    <activate-on-match element="c10" />
```

```
    </state-transition-element>
  - <state-transition-element id="01L" symbol-set="1" latch="true">
      <activate-on-match element="c01" />
    </state-transition-element>
  - <state-transition-element id="00L" symbol-set="0" latch="true">
      <activate-on-match element="c00" />
    </state-transition-element>
  - <counter countone="c11" reset="r11" target="3" at_target="pulse">
      <report-on-target />
    </counter>
  - <counter countone="c10" reset="r10" target="4" at_target="pulse">
      <report-on-target />
    </counter>
  - <counter countone="c01" reset="r01" target="5" at_target="pulse">
      <report-on-target />
    </counter>
  - <counter countone="c00" reset="r00" target="6" at_target="pulse">
      <report-on-target />
    </counter>
  </macro>
- <macro name="radix key 2" id="mx04">
  - <port-definition>
      <element-reference element="h1" type="state-transition-element"
        activation="in" />
      <element-reference element="h0" type="state-transition-element"
        activation="in" />
      <element-reference element="c11" type="state-transition-element"
        reporting="true" />
      <element-reference element="c10" type="state-transition-element"
        reporting="true" />
      <element-reference element="c01" type="state-transition-element"
        reporting="true" />
      <element-reference element="c00" type="state-transition-element"
        reporting="true" />
    </port-definition>
  - <state-transition-element id="h1" symbol-set="1" latch="true">
      <activate-on-match element="ch1" />
    </state-transition-element>
  - <state-transition-element id="h0" symbol-set="0" latch="true">
      <activate-on-match element="ch0" />
    </state-transition-element>
  - <counter countone="ch1" reset="rh1" target="6" at_target="pulse">
      <activate-on-target element="11L" />
      <activate-on-target element="10L" />
    </counter>
  - <counter countone="ch0" reset="rh0" target="6" at_target="pulse">
      <activate-on-target element="01L" />
      <activate-on-target element="00L" />
    </counter>
  - <state-transition-element id="11L" symbol-set="1" latch="true">
      <activate-on-match element="c11" />
    </state-transition-element>
  - <state-transition-element id="10L" symbol-set="0" latch="true">
      <activate-on-match element="c10" />
    </state-transition-element>
  - <state-transition-element id="01L" symbol-set="1" latch="true">
```

```
      <activate-on-match element="c01" />
    </state-transition-element>
  - <state-transition-element id="00L" symbol-set="0" latch="true">
      <activate-on-match element="c00" />
    </state-transition-element>
  - <counter countone="c11" reset="r11" target="2" at_target="pulse">
      <report-on-target />
    </counter>
  - <counter countone="c10" reset="r10" target="3" at_target="pulse">
      <report-on-target />
    </counter>
  - <counter countone="c01" reset="r01" target="4" at_target="pulse">
      <report-on-target />
    </counter>
  - <counter countone="c00" reset="r00" target="5" at_target="pulse">
      <report-on-target />
    </counter>
  </macro>
- <macro name="radix key 2 final" id="mx05">
  - <port-definition>
      <element-reference element="C11" type="state-transition-element"
        reporting="true" />
      <element-reference element="C10" type="state-transition-element"
        reporting="true" />
      <element-reference element="C01" type="state-transition-element"
        reporting="true" />
      <element-reference element="C00" type="state-transition-element"
        reporting="true" />
      <element-reference element="h1" type="state-transition-element"
        activation="in" />
      <element-reference element="h0" type="state-transition-element"
        activation="in" />
    </port-definition>
  - <state-transition-element id="h1" symbol-set="1" latch="true">
      <activate-on-match element="ch1" />
    </state-transition-element>
  - <state-transition-element id="h0" symbol-set="0" latch="true">
      <activate-on-match element="ch0" />
    </state-transition-element>
  - <counter countone="ch1" reset="rh1" target="6" at_target="pulse">
      <activate-on-target element="C11" />
      <activate-on-target element="10" />
    </counter>
  - <counter countone="ch0" reset="rh0" target="6" at_target="pulse">
      <activate-on-target element="01" />
      <activate-on-target element="00" />
    </counter>
  - <state-transition-element id="C11" symbol-set="1">
      <report-on-match />
    </state-transition-element>
  - <state-transition-element id="10" symbol-set="0">
      <activate-on-match element="C10" />
    </state-transition-element>
  - <state-transition-element id="C10" symbol-set="*">
      <report-on-match />
    </state-transition-element>
```

```
  – <state-transition-element id="01" symbol-set="1">
      <activate-on-match element="s1" />
    </state-transition-element>
  – <state-transition-element id="s1" symbol-set="*">
      <activate-on-match element="C01" />
    </state-transition-element>
  – <state-transition-element id="C01" symbol-set="*">
      <report-on-match />
    </state-transition-element>
  – <state-transition-element id="00" symbol-set="0">
      <activate-on-match element="s2" />
    </state-transition-element>
  – <state-transition-element id="s2" symbol-set="*">
      <activate-on-match element="s3" />
    </state-transition-element>
  – <state-transition-element id="s3" symbol-set="*">
      <activate-on-match element="C00" />
    </state-transition-element>
  – <state-transition-element id="C00" symbol-set="*">
      <report-on-match />
    </state-transition-element>
  </macro>
</automata-network>
```

Fuzzy Matching (In Order) 8 Symbol Patterns

Fuzzy Matching (In Order) on 8 Symbol Patterns

This automaton performs fuzzy matching, reporting for each input symbol set a match percentage above a threshold. The match algorithm is simple, each input symbol is compared to a pattern symbol in the same position and if it matches the match percentage is incremented. This is equivalent to a Hamming distance. For this example the number of symbols in each input set is 8 and the pattern width is correspondingly 8. The example tests each input symbol set against 8 patterns.

For example, we could have the following 8 patterns:

```
0: wracking
1: gorillas
2: apraxias
3: practice
4: radicals
5: archduke
6: agnostic
7: guerilla
```

and let's say we have have the follow 3 input symbol sets (initiated with a '#" and separated by '####')

#wracking####godzilla####billards####

and a threshold value of 62.5% (minimum value to report match).

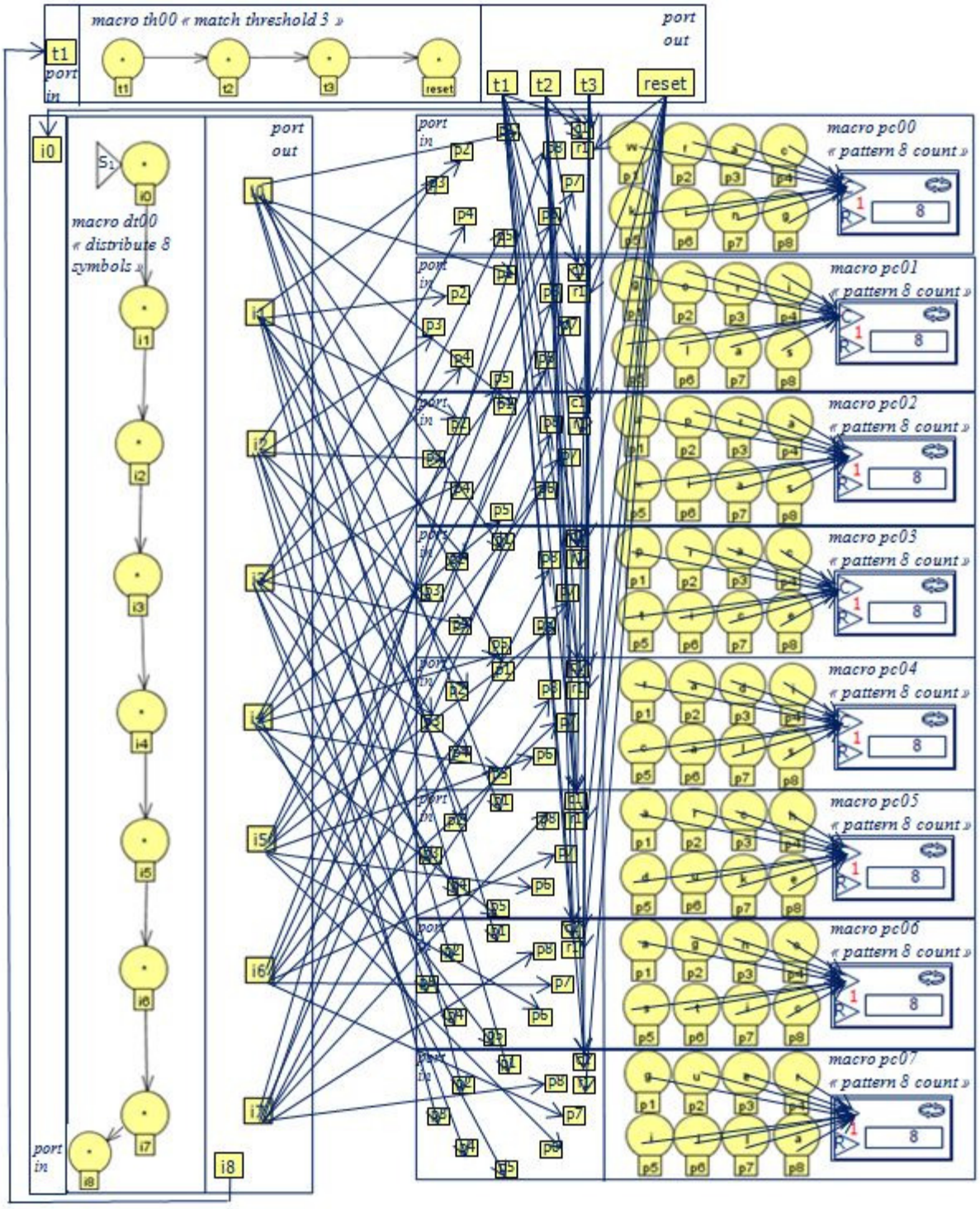
We will report the following fuzzy match scores against the patterns:

input	w	r	a	c	k	i	n	g	score
pattern0	w	r	a	c	k	i	n	g	
match0	w	r	a	c	k	i	n	g	8/8 100%
pattern1	g	o	r	i	l	l	a	s	
match1									below threshold
pattern2	a	p	r	a	x	i	a	s	
match2						i			below threshold
pattern3	p	r	a	c	t	i	c	e	
match3		r	a	c		i			below threshold
pattern4	r	a	d	i	c	a	l	s	
match4									below threshold
pattern5	a	r	c	h	d	u	k	e	
match5		r							below threshold
pattern6	a	g	n	o	s	t	i	c	
match6									below threshold
pattern7	g	u	e	r	i	l	l	a	
match7									below threshold

input	g	o	d	z	i	l	l	a	score
pattern0	w	r	a	c	k	i	n	g	
match0									below threshold
pattern1	g	o	r	i	l	l	a	s	
match1	g	o	d			l			below threshold
pattern2	a	p	r	a	x	i	a	s	
match2									below threshold
pattern3	p	r	a	c	t	i	c	e	
match3									below threshold
pattern4	r	a	d	i	c	a	l	s	
match4			d				l		below threshold
pattern5	a	r	c	h	d	u	k	e	
match5									below threshold
pattern6	a	g	n	o	s	t	i	c	
match6									below threshold
pattern7	g	u	e	r	i	l	l	a	
match7	g				i	l	l	a	5/8 62.5%

input	b	i	l	l	a	r	d	s	score
pattern0	w	r	a	c	k	i	n	g	
match0									below threshold
pattern1	g	o	r	i	l	l	a	s	
match1								s	below threshold
pattern2	a	p	r	a	x	i	a	s	
match2								s	below threshold
pattern3	p	r	a	c	t	i	c	e	
match3									below threshold
pattern4	r	a	d	i	c	a	l	s	
match4								s	below threshold
pattern5	a	r	c	h	d	u	k	e	
match5									below threshold
pattern6	a	g	n	o	s	t	i	c	
match6									below threshold
pattern7	g	u	e	r	i	l	l	a	
match7									below threshold

ANML-G Macro



ANML Template

The automaton is constructed with three different types of macros: "distribute 8 symbols", "match threshold 3", and "pattern 8 count". The distribute macro orders input to the pattern macros and the threshold macro determine the fuzzy match percentage for each pattern in the pattern macros.

macro "distribute 8 symbols"

This macro uses automata to distribute input symbols to other macros that will compute on them. The state-transition-elements form a sequential line to activate the set of elements in the target macros designed for each symbol cycle, with a delay of one cycle for start-up. In the input data we will put an extra symbol at the beginning to prime the distribution. There is a corresponding delay state-transition-element at the end of the state-transition-element chain which gives one cycle for the input symbol to pass to the elemens activate in the previous cycle. The initial state-transition-element is a start element on start-of-data and is also an input element in the macro. As a macro input element another element may activate it to continue looping on the next data set in the input stream;

macro "match threshold 3"

This macro implements the threshold value for fuzzy matching and also resets the pattern counters at the end of the computation. Each t# state-transition-element activates the pattern counters in the pattern macros causing the count to advance toward the counter target. At the counter target the counter will output, with the number of symbol cycles spent in the threshold macro indicating the number of symbols cycles not matched during processing of the input. This macro has three counting state-transition-elements - the threshold can be raised or lowered by changing this number. The last counting state-transition-element (t3 in the example) activate the start state-transition-element in the distribute symbols macro. Activation occurs one cycle before the final symbol in the input stream used to trigger counter reset because the distribute macro has a one cycle delay to enable the pattern macro to start computing symbols. The counting state-transition-elements and the reset state-transition-element operation requires that buffering symbols be placed between input symbol sets.

macro "pattern 8 count"

This macro matches the input symbol set against a pattern and keeps count of the number of matches.

The match algorithm is extremely simple in this example, the input symbol set is the same size as the pattern set and each input symbol is matched in the pattern set in order. For example, input symbol set "12345678" matched against a pattern of "12348765" will match 4 positions and would produce a lower match score than the input symbol set "1234---5" since this set matches 5 positions.

It may be possible to modify the macro to implement other fuzzy matching algorithms such as Levenshtein distance.

Each time a state-transition-element for a symbol position is matched the pattern counter is triggered. The number of matches therefore resides in the counter. A perfect match on all 8 symbols will cause the counter to reach the target and to generate output. This may be recognized as a fuzzy match percentage of 100%. Lower match percentages are determined by the match threshold macro which will continue to trigger the counter until an output event occurs or the threshold limit is reached. The threshold macro also resets the counters for the next input symbol set.

The table below steps through the operation of the example automaton on the input symbol set given at the top of the article.

input symbol	active states	activated	count	output
#	dt00 i0	dt00 i1 pc00-pc07 p1		
w	dt00 i1 pc00-pc07 p1	dt00 i2 pc00-pc07 p2	pc00 1/8	
r	dt00 i2 pc00-pc07 p2	dt00 i3 pc00-pc07 p3	pc00 2/8 pc03 1/8 pc05 1/8	
a	dt00 i3 pc00-pc07 p3	dt00 i4 pc00-pc07 p4	pc00 3/8 pc03 2/8	
c	dt00 i4 pc00-pc07 p4	dt00 i5 pc00-pc07 p5	pc00 4/8 pc03 3/8	
k	dt00 i5 pc00-pc07 p5	dt00 i6 pc00-pc07 p6	pc00 5/8	
i	dt00 i6 pc00-pc07 p6	dt00 i7 pc00-pc07 p7	pc00 6/8 pc02 1/8 pc03 4/8	
n	dt00 i7 pc00-pc07 p7	dt00 i8 pc00-pc07 p8	pc00 7/8	
g	dt00 i8 pc00-pc07 p8	th00 t1	pc00 8/8	pc00 100%
#	th00 t1	th00 t2 pc00-pc07 c1	pc00 1/8 pc01 1/8 pc02 2/8 pc03 5/8 pc04 1/8 pc05 2/8 pc06 1/8 pc07 1/8	
#	th00 t2	th00 t3 pc00-pc07 c1	pc00 2/8 pc01 2/8 pc02 3/8 pc03 6/8 pc04 2/8 pc05 3/8 pc06 2/8 pc07 2/8	
#	th00 t3	dt00 i0 th00 reset pc00-pc07 c1	pc00 3/8 pc01 3/8 pc02 4/8 pc03 7/8 pc04 3/8 pc05 4/8 pc06 3/8 pc07 3/8	
#	dt00 i0 th00 reset	dt00 i1 pc00-pc07 p1	pc00 0/8 pc01 0/8 pc02 0/8 pc03 0/8 pc04 0/8 pc05 0/8 pc06 0/8 pc07 0/8	
g	dt00 i1 pc00-pc07 p1	dt00 i2 pc00-pc07 p2	pc01 1/8 pc07 1/8	
o	dt00 i2 pc00-pc07 p2	dt00 i3 pc00-pc07 p3	pc01 2/8	
d	dt00 i3	dt00 i4	pc01 3/8	

	pc00-pc07 p3	pc00-pc07 p4	pc04 1/8	
z	dt00 i4 pc00-pc07 p4	dt00 i5 pc00-pc07 p5		
i	dt00 i5 pc00-pc07 p5	dt00 i6 pc00-pc07 p6	pc07 2/8	
l	dt00 i6 pc00-pc07 p6	dt00 i7 pc00-pc07 p7	pc01 4/8 pc07 3/8	
l	dt00 i7 pc00-pc07 p7	dt00 i8 pc00-pc07 p8	pc04 2/8 pc07 4/8	
a	dt00 i8 pc00-pc07 p8	th00 t1	pc07 5/8	
#	th00 t1	th00 t2 pc00-pc07 c1	pc00 1/8 pc01 5/8 pc02 1/8 pc03 1/8 pc04 3/8 pc05 1/8 pc06 1/8 pc07 6/8	
#	th00 t2	th00 t3 pc00-pc07 c1	pc00 2/8 pc01 6/8 pc02 2/8 pc03 2/8 pc04 4/8 pc05 2/8 pc06 2/8 pc07 7/8	
#	th00 t3	dt00 i0 th00 reset pc00-pc07 c1	pc00 3/8 pc01 7/8 pc02 2/8 pc03 2/8 pc04 4/8 pc05 2/8 pc06 2/8 pc07 8/8	pc07 62.5%
#	dt00 i0 th00 reset	dt00 i1 pc00-pc07 p1	pc00 0/8 pc01 0/8 pc02 0/8 pc03 0/8 pc04 0/8 pc05 0/8 pc06 0/8 pc07 0/8	
b	dt00 i1 pc00-pc07 p1	dt00 i2 pc00-pc07 p2		
i	dt00 i2 pc00-pc07 p2	dt00 i3 pc00-pc07 p3		
l	dt00 i3 pc00-pc07 p3	dt00 i4 pc00-pc07 p4		
l	dt00 i4 pc00-pc07 p4	dt00 i5 pc00-pc07 p5		
a	dt00 i5 pc00-pc07 p5	dt00 i6 pc00-pc07 p6		
r	dt00 i6 pc00-pc07 p6	dt00 i7 pc00-pc07 p7		
d	dt00 i7 pc00-pc07 p7	dt00 i8 pc00-pc07 p8		
s	dt00 i8	th00 t1	pc01 1/8	

	pc00-pc07 p8		pc02 1/8 pc04 1/8	
#	th00 t1	th00 t2 pc00-pc07 c1	pc00 1/8 pc01 2/8 pc02 2/8 pc03 1/8 pc04 2/8 pc05 1/8 pc06 1/8 pc07 1/8	
#	th00 t2	th00 t3 pc00-pc07 c1	pc00 2/8 pc01 3/8 pc02 3/8 pc03 2/8 pc04 3/8 pc05 2/8 pc06 2/8 pc07 2/8	
#	th00 t3	dt00 i0 th00 reset pc00-pc07 c1	pc00 3/8 pc01 4/8 pc02 4/8 pc03 3/8 pc04 4/8 pc05 3/8 pc06 3/8 pc07 3/8	
#	dt00 i0 th00 reset	dt00 i1 pc00-pc07 p1	pc00 0/8 pc01 0/8 pc02 0/8 pc03 0/8 pc04 0/8 pc05 0/8 pc06 0/8 pc07 0/8	

```
<?xml version="1.0" encoding="UTF-8" ?>
- <automata-network id="an01" name="fuzzy match pattern 8"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="C:/ANML/cookbook/ANML_v6.xsd">
  - <macro id="th00" name="match threshold 3">
    - <port-definition>
        <element-reference element="t1" type="state-transition-element"
          activation="inout" />
        <element-reference element="t2" type="state-transition-element"
          activation="out" />
        <element-reference element="t3" type="state-transition-element"
          activation="out" />
        <element-reference element="reset" type="state-transition-element"
          activation="out" />
      </port-definition>
    - <state-transition-element id="t1" symbol-set="*">
        <activate-on-match element="t2" />
        <activate-on-match element="c1" macro="pc00" />
        <activate-on-match element="c1" macro="pc01" />
        <activate-on-match element="c1" macro="pc02" />
        <activate-on-match element="c1" macro="pc03" />
        <activate-on-match element="c1" macro="pc04" />
        <activate-on-match element="c1" macro="pc05" />
        <activate-on-match element="c1" macro="pc06" />
        <activate-on-match element="c1" macro="pc07" />
      </state-transition-element>
    - <state-transition-element id="t2" symbol-set="*">
        <activate-on-match element="t3" />
        <activate-on-match element="c1" macro="pc00" />
        <activate-on-match element="c1" macro="pc01" />
        <activate-on-match element="c1" macro="pc02" />
        <activate-on-match element="c1" macro="pc03" />
        <activate-on-match element="c1" macro="pc04" />
        <activate-on-match element="c1" macro="pc05" />
        <activate-on-match element="c1" macro="pc06" />
        <activate-on-match element="c1" macro="pc07" />
      </state-transition-element>
    - <state-transition-element id="t3" symbol-set="*">
        <activate-on-match element="reset" />
        <activate-on-match macro="dt00" element="i0" />
        <activate-on-match element="c1" macro="pc00" />
        <activate-on-match element="c1" macro="pc01" />
        <activate-on-match element="c1" macro="pc02" />
        <activate-on-match element="c1" macro="pc03" />
        <activate-on-match element="c1" macro="pc04" />
        <activate-on-match element="c1" macro="pc05" />
        <activate-on-match element="c1" macro="pc06" />
        <activate-on-match element="c1" macro="pc07" />
      </state-transition-element>
    - <state-transition-element id="reset" symbol-set="*">
        <activate-on-match macro="pc00" element="r1" />
        <activate-on-match macro="pc01" element="r1" />
```

```
      <activate-on-match macro="pc02" element="r1" />
      <activate-on-match macro="pc03" element="r1" />
      <activate-on-match macro="pc04" element="r1" />
      <activate-on-match macro="pc05" element="r1" />
      <activate-on-match macro="pc06" element="r1" />
      <activate-on-match macro="pc07" element="r1" />
    </state-transition-element>
  </macro>
- <macro id="dt00" name="distribute 8 symbols">
  - <port-definition>
      <element-reference element="i0" type="state-transition-element"
        activation="inout" start="start-of-data" />
      <element-reference element="i1" type="state-transition-element"
        activation="out" />
      <element-reference element="i2" type="state-transition-element"
        activation="out" />
      <element-reference element="i3" type="state-transition-element"
        activation="out" />
      <element-reference element="i4" type="state-transition-element"
        activation="out" />
      <element-reference element="i5" type="state-transition-element"
        activation="out" />
      <element-reference element="i6" type="state-transition-element"
        activation="out" />
      <element-reference element="i7" type="state-transition-element"
        activation="out" />
      <element-reference element="i8" type="state-transition-element"
        activation="out" />
    </port-definition>
  - <state-transition-element id="i0" symbol-set="*" start="start-of-data">
      <activate-on-match element="i1" />
      <activate-on-match macro="pc00" element="p1" />
      <activate-on-match macro="pc01" element="p1" />
      <activate-on-match macro="pc02" element="p1" />
      <activate-on-match macro="pc03" element="p1" />
      <activate-on-match macro="pc04" element="p1" />
      <activate-on-match macro="pc05" element="p1" />
      <activate-on-match macro="pc06" element="p1" />
      <activate-on-match macro="pc07" element="p1" />
    </state-transition-element>
  - <state-transition-element id="i1" symbol-set="*">
      <activate-on-match element="i2" />
      <activate-on-match macro="pc00" element="p2" />
      <activate-on-match macro="pc01" element="p2" />
      <activate-on-match macro="pc02" element="p2" />
      <activate-on-match macro="pc03" element="p2" />
      <activate-on-match macro="pc04" element="p2" />
      <activate-on-match macro="pc05" element="p2" />
      <activate-on-match macro="pc06" element="p2" />
      <activate-on-match macro="pc07" element="p2" />
    </state-transition-element>
  - <state-transition-element id="i2" symbol-set="*">
      <activate-on-match element="i3" />
```

```
    <activate-on-match macro="pc00" element="p3" />
    <activate-on-match macro="pc01" element="p3" />
    <activate-on-match macro="pc02" element="p3" />
    <activate-on-match macro="pc03" element="p3" />
    <activate-on-match macro="pc04" element="p3" />
    <activate-on-match macro="pc05" element="p3" />
    <activate-on-match macro="pc06" element="p3" />
    <activate-on-match macro="pc07" element="p3" />
  </state-transition-element>
- <state-transition-element id="i3" symbol-set="*">
    <activate-on-match element="i4" />
    <activate-on-match macro="pc00" element="p4" />
    <activate-on-match macro="pc01" element="p4" />
    <activate-on-match macro="pc02" element="p4" />
    <activate-on-match macro="pc03" element="p4" />
    <activate-on-match macro="pc04" element="p4" />
    <activate-on-match macro="pc05" element="p4" />
    <activate-on-match macro="pc06" element="p4" />
    <activate-on-match macro="pc07" element="p4" />
  </state-transition-element>
- <state-transition-element id="i4" symbol-set="*">
    <activate-on-match element="i5" />
    <activate-on-match macro="pc00" element="p5" />
    <activate-on-match macro="pc01" element="p5" />
    <activate-on-match macro="pc02" element="p5" />
    <activate-on-match macro="pc03" element="p5" />
    <activate-on-match macro="pc04" element="p5" />
    <activate-on-match macro="pc05" element="p5" />
    <activate-on-match macro="pc06" element="p5" />
    <activate-on-match macro="pc07" element="p5" />
  </state-transition-element>
- <state-transition-element id="i5" symbol-set="*">
    <activate-on-match element="i6" />
    <activate-on-match macro="pc00" element="p6" />
    <activate-on-match macro="pc01" element="p6" />
    <activate-on-match macro="pc02" element="p6" />
    <activate-on-match macro="pc03" element="p6" />
    <activate-on-match macro="pc04" element="p6" />
    <activate-on-match macro="pc05" element="p6" />
    <activate-on-match macro="pc06" element="p6" />
    <activate-on-match macro="pc07" element="p6" />
  </state-transition-element>
- <state-transition-element id="i6" symbol-set="*">
    <activate-on-match element="i7" />
    <activate-on-match macro="pc00" element="p7" />
    <activate-on-match macro="pc01" element="p7" />
    <activate-on-match macro="pc02" element="p7" />
    <activate-on-match macro="pc03" element="p7" />
    <activate-on-match macro="pc04" element="p7" />
    <activate-on-match macro="pc05" element="p7" />
    <activate-on-match macro="pc06" element="p7" />
    <activate-on-match macro="pc07" element="p7" />
```

```
    </state-transition-element>
  - <state-transition-element id="i7" symbol-set="*">
      <activate-on-match macro="pc00" element="p8" />
      <activate-on-match macro="pc01" element="p8" />
      <activate-on-match macro="pc02" element="p8" />
      <activate-on-match macro="pc03" element="p8" />
      <activate-on-match macro="pc04" element="p8" />
      <activate-on-match macro="pc05" element="p8" />
      <activate-on-match macro="pc06" element="p8" />
      <activate-on-match macro="pc07" element="p8" />
    </state-transition-element>
  - <state-transition-element id="i8" symbol-set="*">
      <activate-on-match macro="dt00" element="i0" />
    </state-transition-element>
  </macro>
- <macro id="pc00" name="pattern 8 count">
  - <port-definition>
      <element-reference element="p1" type="state-transition-element"
        activation="in" />
      <element-reference element="p2" type="state-transition-element"
        activation="in" />
      <element-reference element="p3" type="state-transition-element"
        activation="in" />
      <element-reference element="p4" type="state-transition-element"
        activation="in" />
      <element-reference element="p5" type="state-transition-element"
        activation="in" />
      <element-reference element="p6" type="state-transition-element"
        activation="in" />
      <element-reference element="p7" type="state-transition-element"
        activation="in" />
      <element-reference element="p8" type="state-transition-element"
        activation="in" />
      <element-reference element="c1" type="counter" activation="in"
        reporting="true" />
      <element-reference element="r1" type="counter" activation="in" />
    </port-definition>
  - <state-transition-element id="p1" symbol-set="w">
      <activate-on-match element="c1" />
    </state-transition-element>
  - <state-transition-element id="p2" symbol-set="r">
      <activate-on-match element="c1" />
    </state-transition-element>
  - <state-transition-element id="p3" symbol-set="a">
      <activate-on-match element="c1" />
    </state-transition-element>
  - <state-transition-element id="p4" symbol-set="c">
      <activate-on-match element="c1" />
    </state-transition-element>
  - <state-transition-element id="p5" symbol-set="k">
      <activate-on-match element="c1" />
    </state-transition-element>
  - <state-transition-element id="p6" symbol-set="i">
```

```
      <activate-on-match element="c1" />
    </state-transition-element>
  - <state-transition-element id="p7" symbol-set="n">
      <activate-on-match element="c1" />
    </state-transition-element>
  - <state-transition-element id="p8" symbol-set="g">
      <activate-on-match element="c1" />
    </state-transition-element>
  - <counter countone="c1" reset="r1" target="8" at_target="roll">
      <report-on-target />
    </counter>
  </macro>
- <macro id="pc01" name="pattern 8 count">
  - <port-definition>
      <element-reference element="p1" type="state-transition-element"
        activation="in" />
      <element-reference element="p2" type="state-transition-element"
        activation="in" />
      <element-reference element="p3" type="state-transition-element"
        activation="in" />
      <element-reference element="p4" type="state-transition-element"
        activation="in" />
      <element-reference element="p5" type="state-transition-element"
        activation="in" />
      <element-reference element="p6" type="state-transition-element"
        activation="in" />
      <element-reference element="p7" type="state-transition-element"
        activation="in" />
      <element-reference element="p8" type="state-transition-element"
        activation="in" />
      <element-reference element="c1" type="counter" activation="in"
        reporting="true" />
      <element-reference element="r1" type="counter" activation="in" />
    </port-definition>
  - <state-transition-element id="p1" symbol-set="g">
      <activate-on-match element="c1" />
    </state-transition-element>
  - <state-transition-element id="p2" symbol-set="o">
      <activate-on-match element="c1" />
    </state-transition-element>
  - <state-transition-element id="p3" symbol-set="r">
      <activate-on-match element="c1" />
    </state-transition-element>
  - <state-transition-element id="p4" symbol-set="i">
      <activate-on-match element="c1" />
    </state-transition-element>
  - <state-transition-element id="p5" symbol-set="l">
      <activate-on-match element="c1" />
    </state-transition-element>
  - <state-transition-element id="p6" symbol-set="l">
      <activate-on-match element="c1" />
    </state-transition-element>
  - <state-transition-element id="p7" symbol-set="a">
```

```
      <activate-on-match element="c1" />
    </state-transition-element>
  - <state-transition-element id="p8" symbol-set="s">
      <activate-on-match element="c1" />
    </state-transition-element>
  - <counter countone="c1" reset="r1" target="8" at_target="roll">
      <report-on-target />
    </counter>
  </macro>
- <macro id="pc02" name="pattern 8 count">
  - <port-definition>
      <element-reference element="p1" type="state-transition-element"
        activation="in" />
      <element-reference element="p2" type="state-transition-element"
        activation="in" />
      <element-reference element="p3" type="state-transition-element"
        activation="in" />
      <element-reference element="p4" type="state-transition-element"
        activation="in" />
      <element-reference element="p5" type="state-transition-element"
        activation="in" />
      <element-reference element="p6" type="state-transition-element"
        activation="in" />
      <element-reference element="p7" type="state-transition-element"
        activation="in" />
      <element-reference element="p8" type="state-transition-element"
        activation="in" />
      <element-reference element="c1" type="counter" activation="in"
        reporting="true" />
      <element-reference element="r1" type="counter" activation="in" />
    </port-definition>
  - <state-transition-element id="p1" symbol-set="a">
      <activate-on-match element="c1" />
    </state-transition-element>
  - <state-transition-element id="p2" symbol-set="p">
      <activate-on-match element="c1" />
    </state-transition-element>
  - <state-transition-element id="p3" symbol-set="r">
      <activate-on-match element="c1" />
    </state-transition-element>
  - <state-transition-element id="p4" symbol-set="a">
      <activate-on-match element="c1" />
    </state-transition-element>
  - <state-transition-element id="p5" symbol-set="x">
      <activate-on-match element="c1" />
    </state-transition-element>
  - <state-transition-element id="p6" symbol-set="i">
      <activate-on-match element="c1" />
    </state-transition-element>
  - <state-transition-element id="p7" symbol-set="a">
      <activate-on-match element="c1" />
    </state-transition-element>
  - <state-transition-element id="p8" symbol-set="s">
```

```
      <activate-on-match element="c1" />
    </state-transition-element>
  - <counter countone="c1" reset="r1" target="8" at_target="roll">
      <report-on-target />
    </counter>
  </macro>
- <macro id="pc03" name="pattern 8 count">
  - <port-definition>
      <element-reference element="p1" type="state-transition-element"
        activation="in" />
      <element-reference element="p2" type="state-transition-element"
        activation="in" />
      <element-reference element="p3" type="state-transition-element"
        activation="in" />
      <element-reference element="p4" type="state-transition-element"
        activation="in" />
      <element-reference element="p5" type="state-transition-element"
        activation="in" />
      <element-reference element="p6" type="state-transition-element"
        activation="in" />
      <element-reference element="p7" type="state-transition-element"
        activation="in" />
      <element-reference element="p8" type="state-transition-element"
        activation="in" />
      <element-reference element="c1" type="counter" activation="in"
        reporting="true" />
      <element-reference element="r1" type="counter" activation="in" />
    </port-definition>
  - <state-transition-element id="p1" symbol-set="p">
      <activate-on-match element="c1" />
    </state-transition-element>
  - <state-transition-element id="p2" symbol-set="r">
      <activate-on-match element="c1" />
    </state-transition-element>
  - <state-transition-element id="p3" symbol-set="a">
      <activate-on-match element="c1" />
    </state-transition-element>
  - <state-transition-element id="p4" symbol-set="c">
      <activate-on-match element="c1" />
    </state-transition-element>
  - <state-transition-element id="p5" symbol-set="t">
      <activate-on-match element="c1" />
    </state-transition-element>
  - <state-transition-element id="p6" symbol-set="i">
      <activate-on-match element="c1" />
    </state-transition-element>
  - <state-transition-element id="p7" symbol-set="c">
      <activate-on-match element="c1" />
    </state-transition-element>
  - <state-transition-element id="p8" symbol-set="e">
      <activate-on-match element="c1" />
    </state-transition-element>
  - <counter countone="c1" reset="r1" target="8" at_target="roll">
```

```
      <report-on-target />
    </counter>
  </macro>
- <macro id="pc04" name="pattern 8 count">
  - <port-definition>
      <element-reference element="p1" type="state-transition-element"
        activation="in" />
      <element-reference element="p2" type="state-transition-element"
        activation="in" />
      <element-reference element="p3" type="state-transition-element"
        activation="in" />
      <element-reference element="p4" type="state-transition-element"
        activation="in" />
      <element-reference element="p5" type="state-transition-element"
        activation="in" />
      <element-reference element="p6" type="state-transition-element"
        activation="in" />
      <element-reference element="p7" type="state-transition-element"
        activation="in" />
      <element-reference element="p8" type="state-transition-element"
        activation="in" />
      <element-reference element="c1" type="counter" activation="in"
        reporting="true" />
      <element-reference element="r1" type="counter" activation="in" />
    </port-definition>
  - <state-transition-element id="p1" symbol-set="r">
      <activate-on-match element="c1" />
    </state-transition-element>
  - <state-transition-element id="p2" symbol-set="a">
      <activate-on-match element="c1" />
    </state-transition-element>
  - <state-transition-element id="p3" symbol-set="d">
      <activate-on-match element="c1" />
    </state-transition-element>
  - <state-transition-element id="p4" symbol-set="i">
      <activate-on-match element="c1" />
    </state-transition-element>
  - <state-transition-element id="p5" symbol-set="c">
      <activate-on-match element="c1" />
    </state-transition-element>
  - <state-transition-element id="p6" symbol-set="a">
      <activate-on-match element="c1" />
    </state-transition-element>
  - <state-transition-element id="p7" symbol-set="l">
      <activate-on-match element="c1" />
    </state-transition-element>
  - <state-transition-element id="p8" symbol-set="s">
      <activate-on-match element="c1" />
    </state-transition-element>
  - <counter countone="c1" reset="r1" target="8" at_target="roll">
      <report-on-target />
    </counter>
  </macro>
```

```
- <macro id="pc05" name="pattern 8 count">
  - <port-definition>
      <element-reference element="p1" type="state-transition-element"
        activation="in" />
      <element-reference element="p2" type="state-transition-element"
        activation="in" />
      <element-reference element="p3" type="state-transition-element"
        activation="in" />
      <element-reference element="p4" type="state-transition-element"
        activation="in" />
      <element-reference element="p5" type="state-transition-element"
        activation="in" />
      <element-reference element="p6" type="state-transition-element"
        activation="in" />
      <element-reference element="p7" type="state-transition-element"
        activation="in" />
      <element-reference element="p8" type="state-transition-element"
        activation="in" />
      <element-reference element="c1" type="counter" activation="in"
        reporting="true" />
      <element-reference element="r1" type="counter" activation="in" />
    </port-definition>
  - <state-transition-element id="p1" symbol-set="a">
      <activate-on-match element="c1" />
    </state-transition-element>
  - <state-transition-element id="p2" symbol-set="r">
      <activate-on-match element="c1" />
    </state-transition-element>
  - <state-transition-element id="p3" symbol-set="c">
      <activate-on-match element="c1" />
    </state-transition-element>
  - <state-transition-element id="p4" symbol-set="h">
      <activate-on-match element="c1" />
    </state-transition-element>
  - <state-transition-element id="p5" symbol-set="d">
      <activate-on-match element="c1" />
    </state-transition-element>
  - <state-transition-element id="p6" symbol-set="u">
      <activate-on-match element="c1" />
    </state-transition-element>
  - <state-transition-element id="p7" symbol-set="k">
      <activate-on-match element="c1" />
    </state-transition-element>
  - <state-transition-element id="p8" symbol-set="e">
      <activate-on-match element="c1" />
    </state-transition-element>
  - <counter countone="c1" reset="r1" target="8" at_target="roll">
      <report-on-target />
    </counter>
  </macro>
- <macro id="pc06" name="pattern 8 count">
  - <port-definition>
      <element-reference element="p1" type="state-transition-element"
```

```
    activation="in" />
      <element-reference element="p2" type="state-transition-element"
        activation="in" />
      <element-reference element="p3" type="state-transition-element"
        activation="in" />
      <element-reference element="p4" type="state-transition-element"
        activation="in" />
      <element-reference element="p5" type="state-transition-element"
        activation="in" />
      <element-reference element="p6" type="state-transition-element"
        activation="in" />
      <element-reference element="p7" type="state-transition-element"
        activation="in" />
      <element-reference element="p8" type="state-transition-element"
        activation="in" />
      <element-reference element="c1" type="counter" activation="in"
        reporting="true" />
      <element-reference element="r1" type="counter" activation="in" />
    </port-definition>
  - <state-transition-element id="p1" symbol-set="a">
      <activate-on-match element="c1" />
    </state-transition-element>
  - <state-transition-element id="p2" symbol-set="g">
      <activate-on-match element="c1" />
    </state-transition-element>
  - <state-transition-element id="p3" symbol-set="n">
      <activate-on-match element="c1" />
    </state-transition-element>
  - <state-transition-element id="p4" symbol-set="o">
      <activate-on-match element="c1" />
    </state-transition-element>
  - <state-transition-element id="p5" symbol-set="s">
      <activate-on-match element="c1" />
    </state-transition-element>
  - <state-transition-element id="p6" symbol-set="t">
      <activate-on-match element="c1" />
    </state-transition-element>
  - <state-transition-element id="p7" symbol-set="i">
      <activate-on-match element="c1" />
    </state-transition-element>
  - <state-transition-element id="p8" symbol-set="c">
      <activate-on-match element="c1" />
    </state-transition-element>
  - <counter countone="c1" reset="r1" target="8" at_target="roll">
      <report-on-target />
    </counter>
  </macro>
- <macro id="pc07" name="pattern 8 count">
  - <port-definition>
      <element-reference element="p1" type="state-transition-element"
        activation="in" />
      <element-reference element="p2" type="state-transition-element"
        activation="in" />
```

```
      <element-reference element="p3" type="state-transition-element"
        activation="in" />
      <element-reference element="p4" type="state-transition-element"
        activation="in" />
      <element-reference element="p5" type="state-transition-element"
        activation="in" />
      <element-reference element="p6" type="state-transition-element"
        activation="in" />
      <element-reference element="p7" type="state-transition-element"
        activation="in" />
      <element-reference element="p8" type="state-transition-element"
        activation="in" />
      <element-reference element="c1" type="counter" activation="in"
        reporting="true" />
      <element-reference element="r1" type="counter" activation="in" />
    </port-definition>
  - <state-transition-element id="p1" symbol-set="g">
      <activate-on-match element="c1" />
    </state-transition-element>
  - <state-transition-element id="p2" symbol-set="u">
      <activate-on-match element="c1" />
    </state-transition-element>
  - <state-transition-element id="p3" symbol-set="e">
      <activate-on-match element="c1" />
    </state-transition-element>
  - <state-transition-element id="p4" symbol-set="r">
      <activate-on-match element="c1" />
    </state-transition-element>
  - <state-transition-element id="p5" symbol-set="i">
      <activate-on-match element="c1" />
    </state-transition-element>
  - <state-transition-element id="p6" symbol-set="l">
      <activate-on-match element="c1" />
    </state-transition-element>
  - <state-transition-element id="p7" symbol-set="l">
      <activate-on-match element="c1" />
    </state-transition-element>
  - <state-transition-element id="p8" symbol-set="a">
      <activate-on-match element="c1" />
    </state-transition-element>
  - <counter countone="c1" reset="r1" target="8" at_target="roll">
      <report-on-target />
    </counter>
  </macro>
</automata-network>
```

Fuzzy Matching Extended to 1K 256 wide Symbol Patterns

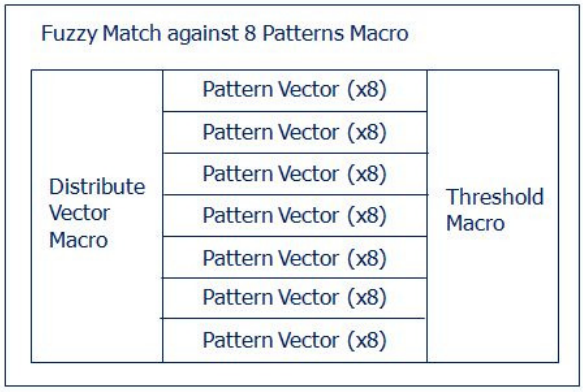
Fuzzy Matching Extended to 1K 256 wide Symbol Patterns

This automata uses the same design concepts as shown in the previous article on fuzzy matching on 8 wide symbol patterns but modifies and extends it to handle 1024 symbol patterns each of which is 256 symbols wide. The 8 symbol pattern macro is still used for testing against the input symbol stream so this design must change the pattern in the pattern macro 32 times to implement a 256 symbol wide pattern. While this may be an impractical machine from a performance perspective there may be situations where it will be effective to retain a fixed topology and to change the patterns encoding in state-transition-elements. This machine shows one way to do that.

The runtime environment may support *poke* operations which will allow one or more symbol-sets in state-transition-elements to be changed. This design uses *poke* to change **all** symbol-sets in the entire automata every eight symbol cycles. The input symbol vector size is 256 and the pattern vector size is also is 256 so after every eight symbols are processed from the input the pattern vector is changed, thereby running the entire 256 symbol input vector against a 256 symbol-set pattern. The diagram below represents the concept of breaking the processing of the symbol input vector into 8 symbol chunks followed by a poke.

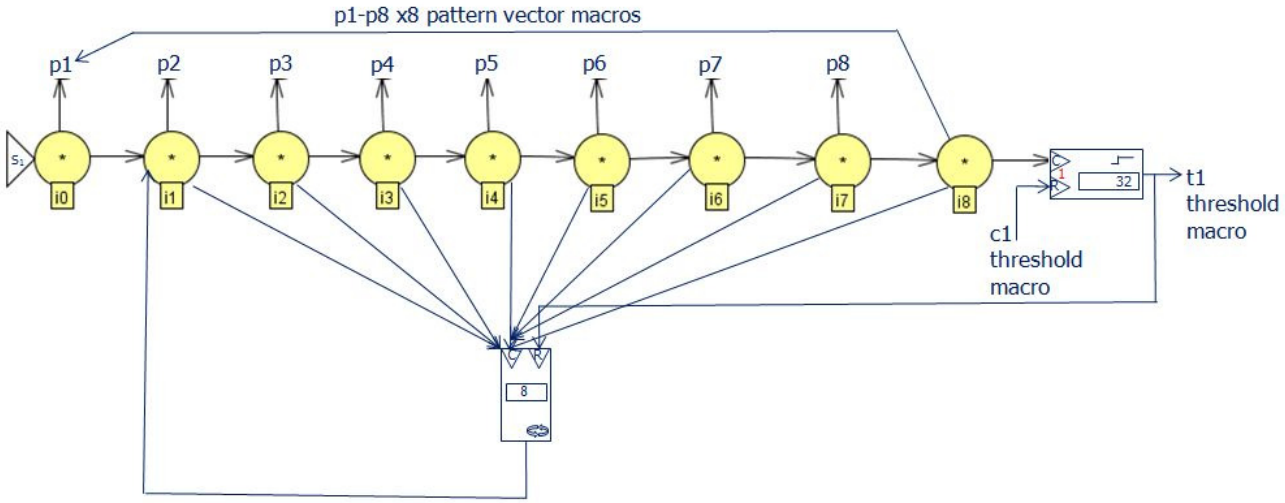
There are 1024 pattern vectors and since each pattern vector has 8 symbol-sets 8192 symbol sets are changed on every 8 symbol cycles.

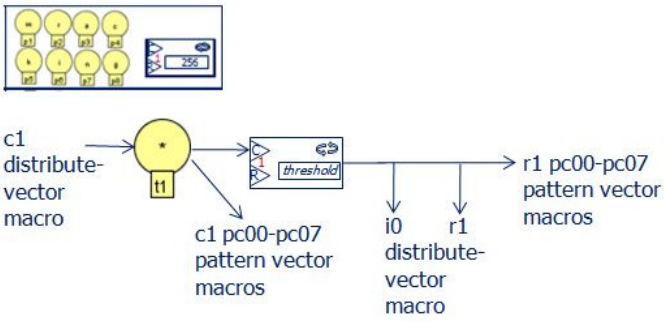
Pattern vectors are assembled into groups of 8, with each group sharing pre- and post-processing macros for distributing the input symbols and calculating the fuzzy match scores at or above a threshold. Here is a logical view of this structure:



There are 128 pattern groups needed to implement an automata with 1024 pattern vectors.

The macros used in the pattern groups are similar to those used in the previous article describing an 8 pattern vector machine but a number of modifications were made to handle the 256 symbol pattern size and the cycling of symbol patterns in 8 symbol increments.




c1
distribute-
vector
macro
t1
c1 pc00-pc07
pattern vector
macros
threshold
i0
r1
distribute-
vector
macro
r1 pc00-pc07
pattern vector
macros

Comparator, 3 bit Values

Comparator, 3 bit Values

This automaton takes a stream of 3-bit value pairs and reports whether the first (n_1) is numerically greater or lesser than the second (n_2). For example, for the stream of values

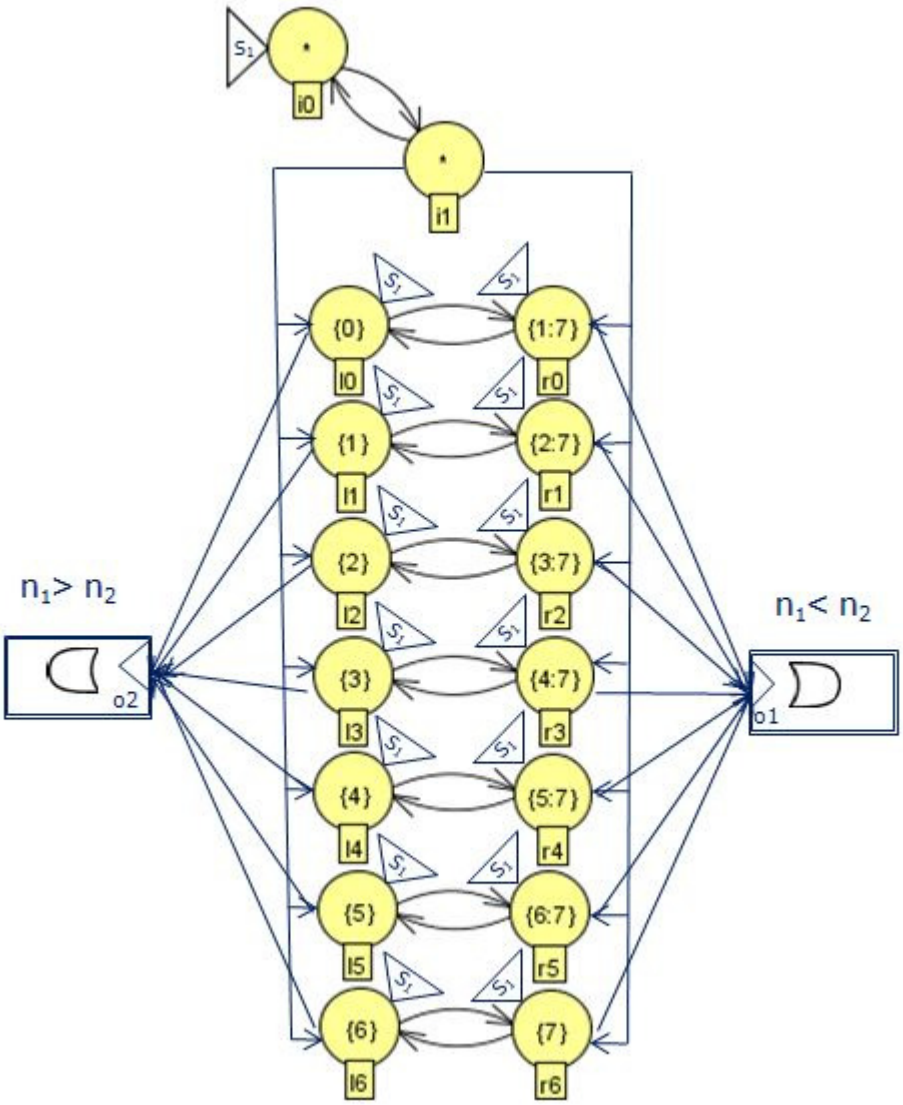
7 4 3 3 0 5 5 0 2 7

at 2nd symbol cycle report $n_1 > n_2$
at 4th symbol cycle report that neither $n_1 > n_2$ nor $n_1 < n_2$ are true
at 6th symbol cycle report $n_1 < n_2$
at 8th symbol cycle report $n_1 > n_2$
at 10th symbol cycle report $n_1 < n_2$

The automaton presented below economizes usage of state-transition-elements, reporting matches from both sides of the two-input state-transition-elements pairs and will therefore report matches on the odd symbol cycles. These matches (one or two of the output elements may report) are ignored, the correct result only appears on the even cycles when comparing pair of values. Also, to further conserve state-transition-elements no report is generated when the two values are equal. Equality would be inferred by the absence of an output report on an even symbol cycle.

The ANML description uses the bits-enabled notation for the symbol-set.

ANML-G Macro



ANML Template

This automaton tests for every possible value of in the symbol-set (decimal 0-7 for 3 bits) on the right hand side and for all values in the symbol set greater than that value on the left hand side. The converse test is made from left to right, simultaneously, the first value is tested for a range of values and the second value for the first value less than that range. There are three possible outcomes, occuring on the even symbol cycles, a state-transition-element matches on either the right or left side, activating either combinatorial element o2 or o1, indicating where value is greater, or there is no output which indicates that the values were equal. On odd cycles one or both of o1 and o2 may report output but this should be discarded by the application.

The two state-transition-elements at the top serve to activate the first value state-transition-elements every 2 cycles so that a continuous stream of value pairs can be read.

This automaton can easily be extended to cover 8 bit values by duplicating the containing macro 32 times and setting the symbol-set to reflect the larger range. It should also be possible to increase the size of macro to cover 4 bits, reducing the number of macros needed

to 16.

There are number of variations to this design that can alter the reporting behavior. If the macro is duplicated and the bi-directional transitions between the state-transition-elements and one of the or elements is removed, leaving one macro with right transitions and the other with left transitions and corresponding reporting of only one of greater-than or less-than no spurious reports will be generated on the the odd symbol cycles. But this approach doubles the number of state-transition-elements. Another possibility is to use the end-of-data signal in the or elements to only enable output on the even symbol cycles. This requires support for this in the API. Another variation is to not use the or elements at all, saving two OR elements per macro, and to make each state-transition-element output reporting. This will cause an even larger number of spurious matches on the odd symbol cycles as as many as all 7 of the right-hand state-transition-elements can generate output on a single cycle. This could be mitigated by using the or element on the right-hand side where multiple matches may occur but not on the left-hand side, where no more than one state-transition-element can report output on a symbol cycle. The two-macro, greater-than and less-than independant comparator design could go one step further, using no or elements since there is only a single matching state-transition element on each even symbol cycle.

All the variations discussed above are illustrated below.

1. 4-bit Comparator
2. 3-bit Comparator with GT LT EQ macros and no spurious matches
3. Comparator, 3 bit with end-of-data to eliminate spurious matches
4. 3-bit Comparator using STE implicit OR instead of OR element
5. 3-bit Comparator without OR elements
6. 3-bit Comparator with 1 OR element

```
<?xml version="1.0" ?>
- <macro name="Comparator, 3 bit" id="m01"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="file:///c:/ANML/cookbook/ANML_v4.xsd">
  - <state-transition-element id="i0" symbol-set="*" start="start-of-data">
      <activate-on-match element="i1" />
    </state-transition-element>
  - <state-transition-element id="i1" symbol-set="*">
      <activate-on-match element="i0" />
      <activate-on-match element="l0" />
      <activate-on-match element="l1" />
      <activate-on-match element="l2" />
      <activate-on-match element="l3" />
      <activate-on-match element="l4" />
      <activate-on-match element="l5" />
      <activate-on-match element="l6" />
      <activate-on-match element="r0" />
      <activate-on-match element="r1" />
      <activate-on-match element="r2" />
      <activate-on-match element="r3" />
      <activate-on-match element="r4" />
      <activate-on-match element="r5" />
      <activate-on-match element="r6" />
    </state-transition-element>
  - <state-transition-element id="l0" symbol-set="{0}" start="start-of-data">
      <activate-on-match element="r0" />
      <activate-on-match element="o2" />
    </state-transition-element>
  - <state-transition-element id="l1" symbol-set="{1}" start="start-of-data">
      <activate-on-match element="r1" />
      <activate-on-match element="o2" />
    </state-transition-element>
  - <state-transition-element id="l2" symbol-set="{2}" start="start-of-data">
      <activate-on-match element="r2" />
      <activate-on-match element="o2" />
    </state-transition-element>
  - <state-transition-element id="l3" symbol-set="{3}" start="start-of-data">
      <activate-on-match element="r3" />
      <activate-on-match element="o2" />
    </state-transition-element>
  - <state-transition-element id="l4" symbol-set="{4}" start="start-of-data">
      <activate-on-match element="r4" />
      <activate-on-match element="o2" />
    </state-transition-element>
  - <state-transition-element id="l5" symbol-set="{5}" start="start-of-data">
      <activate-on-match element="r5" />
      <activate-on-match element="o2" />
    </state-transition-element>
  - <state-transition-element id="l6" symbol-set="{6}" start="start-of-data">
      <activate-on-match element="r6" />
      <activate-on-match element="o2" />
    </state-transition-element>
  - <state-transition-element id="r0" symbol-set="{1:7}" start="start-of-data">
      <activate-on-match element="l0" />
      <activate-on-match element="o1" />
    </state-transition-element>
```

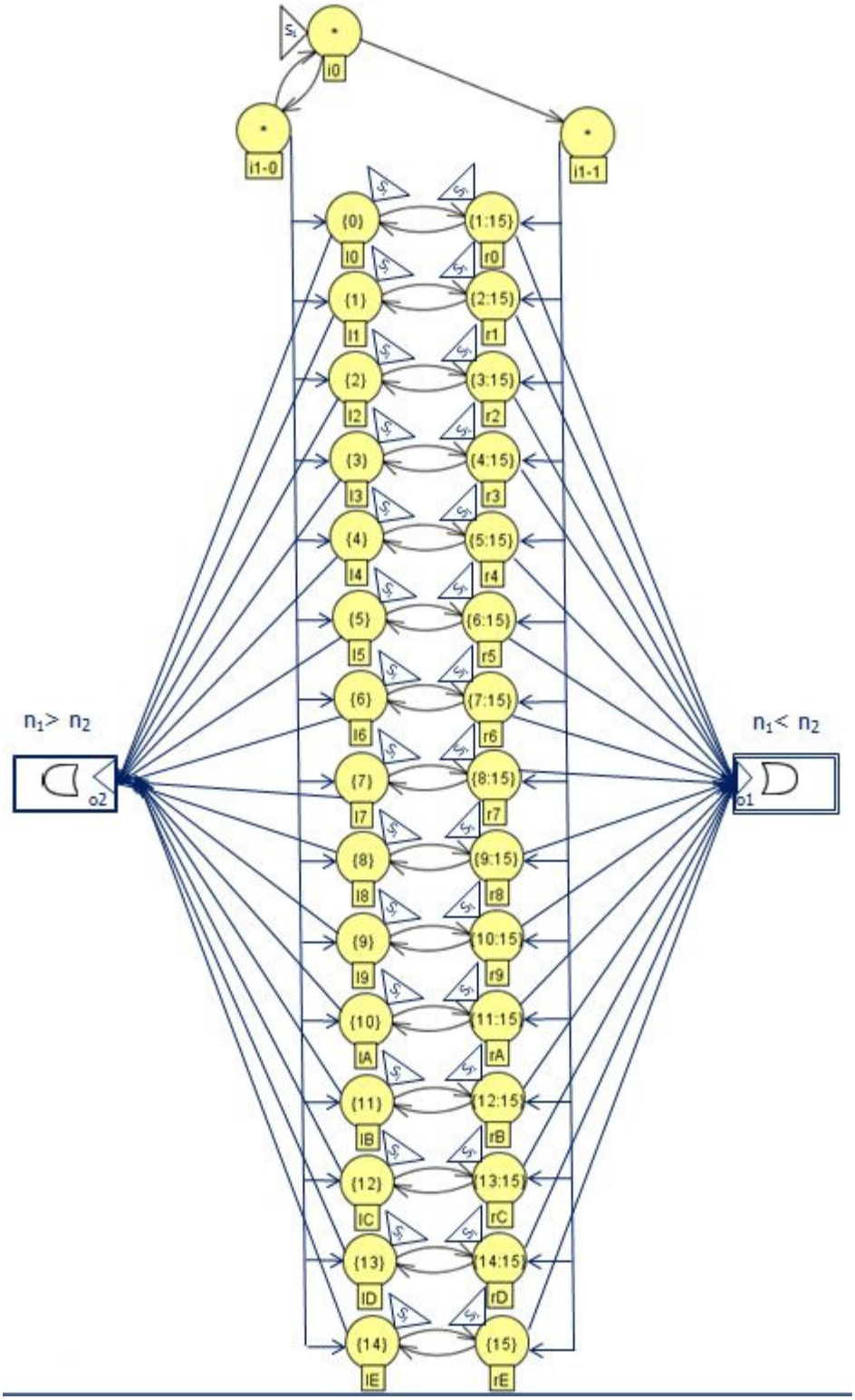
```
- <state-transition-element id="r1" symbol-set="{2:7}" start="start-of-data">
    <activate-on-match element="l1" />
    <activate-on-match element="o1" />
  </state-transition-element>
- <state-transition-element id="r2" symbol-set="{3:7}" start="start-of-data">
    <activate-on-match element="l2" />
    <activate-on-match element="o1" />
  </state-transition-element>
- <state-transition-element id="r3" symbol-set="{4:7}" start="start-of-data">
    <activate-on-match element="l3" />
    <activate-on-match element="o1" />
  </state-transition-element>
- <state-transition-element id="r4" symbol-set="{5:7}" start="start-of-data">
    <activate-on-match element="l4" />
    <activate-on-match element="o1" />
  </state-transition-element>
- <state-transition-element id="r5" symbol-set="{6:7}" start="start-of-data">
    <activate-on-match element="l5" />
    <activate-on-match element="o1" />
  </state-transition-element>
- <state-transition-element id="r6" symbol-set="{7}" start="start-of-data">
    <activate-on-match element="l6" />
    <activate-on-match element="o1" />
  </state-transition-element>
- <or id="o2">
    <!-- first number is greater than second number -->
    <report-on-high />
  </or>
- <or id="o1">
    <!-- first number is less than second number -->
    <report-on-high />
  </or>
</macro>
```

Comparator, 4 bit Values

Comparator, 4 bit Values

This automaton expands the design of the previous 3 bit comparator to compare pairs of values between decimal values 0 and 15. Output is generated on every symbol cycle but only the even symbol cycles reports the result of the comparison.

ANML-G Macro



ANML Template

Two state-transition-elements are used to activate the right and left-hand value-recognizing state-transition-elements on even symbol cycles instead of one in the three-bit comparator to half a fan-out likely to exceed implemention parameters.

```
<?xml version="1.0" ?>
- <macro name="Comparator, 4 bit" id="m01"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="file:///c:/ANML/cookbook/ANML_v4.xsd">
  - <state-transition-element id="i0" symbol-set="*" start="start-of-data">
      <activate-on-match element="i1-0" />
      <activate-on-match element="i1-1" />
    </state-transition-element>
  - <state-transition-element id="i1-0" symbol-set="*">
      <activate-on-match element="i0" />
      <activate-on-match element="l0" />
      <activate-on-match element="l1" />
      <activate-on-match element="l2" />
      <activate-on-match element="l3" />
      <activate-on-match element="l4" />
      <activate-on-match element="l5" />
      <activate-on-match element="l6" />
      <activate-on-match element="l7" />
      <activate-on-match element="l8" />
      <activate-on-match element="l9" />
      <activate-on-match element="lA" />
      <activate-on-match element="lB" />
      <activate-on-match element="lC" />
      <activate-on-match element="lD" />
      <activate-on-match element="lE" />
    </state-transition-element>
  - <state-transition-element id="i1-1" symbol-set="*">
      <activate-on-match element="l0" />
      <activate-on-match element="l1" />
      <activate-on-match element="l2" />
      <activate-on-match element="l3" />
      <activate-on-match element="l4" />
      <activate-on-match element="l5" />
      <activate-on-match element="l6" />
      <activate-on-match element="r0" />
      <activate-on-match element="r1" />
      <activate-on-match element="r2" />
      <activate-on-match element="r3" />
      <activate-on-match element="r4" />
      <activate-on-match element="r5" />
      <activate-on-match element="r6" />
    </state-transition-element>
  + <state-transition-element id="l0" symbol-set="{0}" start="start-of-data">
  - <state-transition-element id="l1" symbol-set="{1}" start="start-of-data">
      <activate-on-match element="r1" />
      <activate-on-match element="o2" />
    </state-transition-element>
  - <state-transition-element id="l2" symbol-set="{2}" start="start-of-data">
      <activate-on-match element="r2" />
      <activate-on-match element="o2" />
    </state-transition-element>
  - <state-transition-element id="l3" symbol-set="{3}" start="start-of-data">
      <activate-on-match element="r3" />
      <activate-on-match element="o2" />
    </state-transition-element>
  - <state-transition-element id="l4" symbol-set="{4}" start="start-of-data">
```

```
    <activate-on-match element="r4" />
    <activate-on-match element="o2" />
  </state-transition-element>
- <state-transition-element id="l5" symbol-set="{5}" start="start-of-data">
    <activate-on-match element="r5" />
    <activate-on-match element="o2" />
  </state-transition-element>
- <state-transition-element id="l6" symbol-set="{6}" start="start-of-data">
    <activate-on-match element="r6" />
    <activate-on-match element="o2" />
  </state-transition-element>
- <state-transition-element id="r0" symbol-set="{1:7}" start="start-of-data">
    <activate-on-match element="l0" />
    <activate-on-match element="o1" />
  </state-transition-element>
- <state-transition-element id="r1" symbol-set="{2:7}" start="start-of-data">
    <activate-on-match element="l1" />
    <activate-on-match element="o1" />
  </state-transition-element>
- <state-transition-element id="r2" symbol-set="{3:7}" start="start-of-data">
    <activate-on-match element="l2" />
    <activate-on-match element="o1" />
  </state-transition-element>
- <state-transition-element id="r3" symbol-set="{4:7}" start="start-of-data">
    <activate-on-match element="l3" />
    <activate-on-match element="o1" />
  </state-transition-element>
- <state-transition-element id="r4" symbol-set="{5:7}" start="start-of-data">
    <activate-on-match element="l4" />
    <activate-on-match element="o1" />
  </state-transition-element>
- <state-transition-element id="r5" symbol-set="{6:7}" start="start-of-data">
    <activate-on-match element="l5" />
    <activate-on-match element="o1" />
  </state-transition-element>
- <state-transition-element id="r6" symbol-set="{7}" start="start-of-data">
    <activate-on-match element="l6" />
    <activate-on-match element="o1" />
  </state-transition-element>
- <state-transition-element id="l0" symbol-set="{0}" start="start-of-data">
    <activate-on-match element="r0" />
    <activate-on-match element="o2" />
  </state-transition-element>
- <state-transition-element id="l1" symbol-set="{1}" start="start-of-data">
    <activate-on-match element="r1" />
    <activate-on-match element="o2" />
  </state-transition-element>
- <state-transition-element id="l2" symbol-set="{2}" start="start-of-data">
    <activate-on-match element="r2" />
    <activate-on-match element="o2" />
  </state-transition-element>
- <state-transition-element id="l3" symbol-set="{3}" start="start-of-data">
    <activate-on-match element="r3" />
    <activate-on-match element="o2" />
  </state-transition-element>
- <state-transition-element id="l4" symbol-set="{4}" start="start-of-data">
```

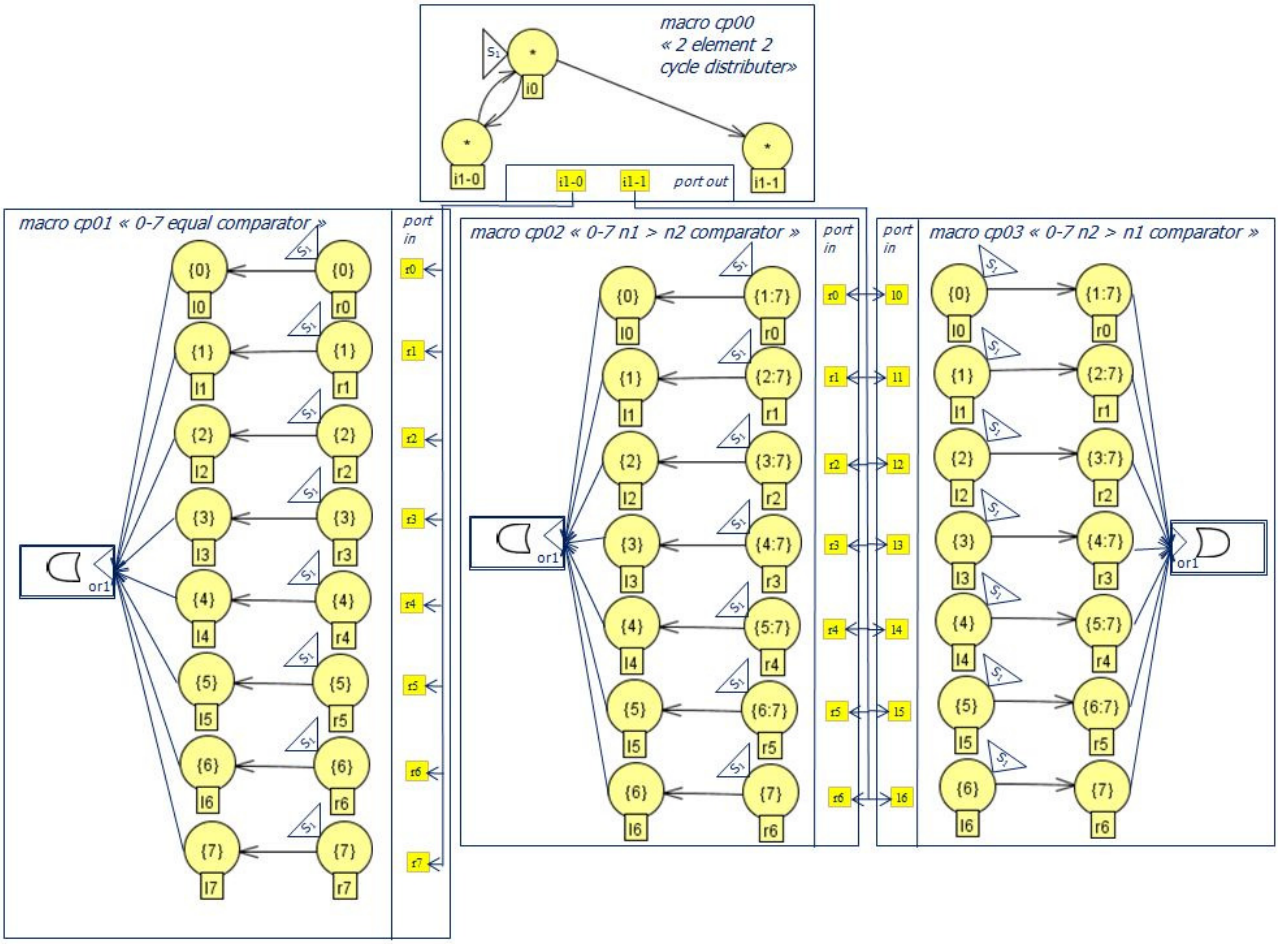
```
    <activate-on-match element="r4" />
    <activate-on-match element="o2" />
  </state-transition-element>
- <state-transition-element id="l5" symbol-set="{5}" start="start-of-data">
    <activate-on-match element="r5" />
    <activate-on-match element="o2" />
  </state-transition-element>
- <state-transition-element id="l6" symbol-set="{6}" start="start-of-data">
    <activate-on-match element="r6" />
    <activate-on-match element="o2" />
  </state-transition-element>
- <state-transition-element id="r0" symbol-set="{1:7}" start="start-of-data">
    <activate-on-match element="l0" />
    <activate-on-match element="o1" />
  </state-transition-element>
- <state-transition-element id="r1" symbol-set="{2:7}" start="start-of-data">
    <activate-on-match element="l1" />
    <activate-on-match element="o1" />
  </state-transition-element>
- <state-transition-element id="r2" symbol-set="{3:7}" start="start-of-data">
    <activate-on-match element="l2" />
    <activate-on-match element="o1" />
  </state-transition-element>
- <state-transition-element id="r3" symbol-set="{4:7}" start="start-of-data">
    <activate-on-match element="l3" />
    <activate-on-match element="o1" />
  </state-transition-element>
- <state-transition-element id="r4" symbol-set="{5:7}" start="start-of-data">
    <activate-on-match element="l4" />
    <activate-on-match element="o1" />
  </state-transition-element>
- <state-transition-element id="r5" symbol-set="{6:7}" start="start-of-data">
    <activate-on-match element="l5" />
    <activate-on-match element="o1" />
  </state-transition-element>
- <state-transition-element id="r6" symbol-set="{7}" start="start-of-data">
    <activate-on-match element="l6" />
    <activate-on-match element="o1" />
  </state-transition-element>
- <or id="o2">
    <!--  first number is greater than second number   -->
    <report-on-high />
  </or>
- <or id="o1">
    <!--  first number is less than second number   -->
    <report-on-high />
  </or>
</macro>
```

Comparator, 3 bit with separate gt and lt automatons to eliminatespurious matches

Comparator, 3 bit with separate GT LT and EQ to eliminate spurious matches

This automaton is a variation of the 3-bit comparator which does not generate "spurious" matches. Spurious matches are those that occur on off-cycles that must be ignored by the application. This automaton only produces a match result on every other cycle when two numbers have been compared. It also produces a distinct output for all three possible conditions: equality, first number less than second, and first number greater than second.

ANML-G Macro



ANML Template

The automaton uses many more state-transition-elements than the comparator which produces spurious matches. It also has many more connections between state-transition-elements so it uses the same input distributor as the 4-bit comparator to reduce problems with excessive fan-out. The number of state-transition-elements could be reduced in this design by remove one of the macros for equality, greater-than or less-than conditions, as the existence of this condition could be inferred by the lack of the other two on an output-producing symbol cycle.

```
<?xml version="1.0" ?>
- <automata-network name="an01" id="3 bit comparator with GT LT EQ no spurious matches"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="file:///c:/ANML/cookbook/ANML_v4.xsd">
  - <macro name="2 element 2 cycle distributer" id="cp00">
    - <port-definition>
        <element-reference element="i1-0" type="out" />
        <element-reference element="i1-1" type="out" />
      </port-definition>
    - <state-transition-element id="i0" symbol-set="*" start="start-of-data">
        <activate-on-match element="i1-0" />
        <activate-on-match element="i1-1" />
      </state-transition-element>
    - <state-transition-element id="i1-0" symbol-set="*">
        <activate-on-match element="i0" />
        <activate-on-match macro="cp01" element="r0" />
        <activate-on-match macro="cp01" element="r1" />
        <activate-on-match macro="cp01" element="r2" />
        <activate-on-match macro="cp01" element="r3" />
        <activate-on-match macro="cp01" element="r4" />
        <activate-on-match macro="cp01" element="r5" />
        <activate-on-match macro="cp01" element="r6" />
        <activate-on-match macro="cp01" element="r7" />
      </state-transition-element>
    - <state-transition-element id="i1-1" symbol-set="*">
        <activate-on-match macro="cp02" element="r0" />
        <activate-on-match macro="cp02" element="r1" />
        <activate-on-match macro="cp02" element="r2" />
        <activate-on-match macro="cp02" element="r3" />
        <activate-on-match macro="cp02" element="r4" />
        <activate-on-match macro="cp02" element="r5" />
        <activate-on-match macro="cp02" element="r6" />
        <activate-on-match macro="cp03" element="l0" />
        <activate-on-match macro="cp03" element="l1" />
        <activate-on-match macro="cp03" element="l2" />
        <activate-on-match macro="cp03" element="l3" />
        <activate-on-match macro="cp03" element="l4" />
        <activate-on-match macro="cp03" element="l5" />
        <activate-on-match macro="cp03" element="l6" />
      </state-transition-element>
    </macro>
  - <macro name="0-7 equal comparator" id="cp01">
    - <port-definition>
        <element-reference element="r0" type="in" />
        <element-reference element="r1" type="in" />
        <element-reference element="r2" type="in" />
        <element-reference element="r3" type="in" />
        <element-reference element="r4" type="in" />
        <element-reference element="r5" type="in" />
        <element-reference element="r6" type="in" />
        <element-reference element="r7" type="in" />
      </port-definition>
    - <state-transition-element id="r0" symbol-set="{0}" start="start-of-data">
        <activate-on-match element="l0" />
      </state-transition-element>
    - <state-transition-element id="r1" symbol-set="{1}" start="start-of-data">
```

```
    <activate-on-match element="I1" />
  </state-transition-element>
- <state-transition-element id="r2" symbol-set="{2}" start="start-of-data">
    <activate-on-match element="I2" />
  </state-transition-element>
- <state-transition-element id="r3" symbol-set="{3}" start="start-of-data">
    <activate-on-match element="I3" />
  </state-transition-element>
- <state-transition-element id="r4" symbol-set="{4}" start="start-of-data">
    <activate-on-match element="I4" />
  </state-transition-element>
- <state-transition-element id="r5" symbol-set="{5}" start="start-of-data">
    <activate-on-match element="I5" />
  </state-transition-element>
- <state-transition-element id="r6" symbol-set="{6}" start="start-of-data">
    <activate-on-match element="I6" />
  </state-transition-element>
- <state-transition-element id="r7" symbol-set="{7}" start="start-of-data">
    <activate-on-match element="I7" />
  </state-transition-element>
- <state-transition-element id="I0" symbol-set="{0}">
    <activate-on-match element="or1" />
  </state-transition-element>
- <state-transition-element id="I0" symbol-set="{1}">
    <activate-on-match element="or1" />
  </state-transition-element>
- <state-transition-element id="I0" symbol-set="{2}">
    <activate-on-match element="or1" />
  </state-transition-element>
- <state-transition-element id="I0" symbol-set="{3}">
    <activate-on-match element="or1" />
  </state-transition-element>
- <state-transition-element id="I0" symbol-set="{4}">
    <activate-on-match element="or1" />
  </state-transition-element>
- <state-transition-element id="I0" symbol-set="{5}">
    <activate-on-match element="or1" />
  </state-transition-element>
- <state-transition-element id="I0" symbol-set="{6}">
    <activate-on-match element="or1" />
  </state-transition-element>
- <state-transition-element id="I0" symbol-set="{7}">
    <activate-on-match element="or1" />
  </state-transition-element>
- <or id="or1">
    <!--  first number is equal to second number  -->
    <report-on-high />
  </or>
</macro>
- <macro name="0-7 n1 > n2 comparator" id="cp02">
- <port-definition>
    <element-reference element="r0" type="in" />
    <element-reference element="r1" type="in" />
    <element-reference element="r2" type="in" />
    <element-reference element="r3" type="in" />
    <element-reference element="r4" type="in" />
```

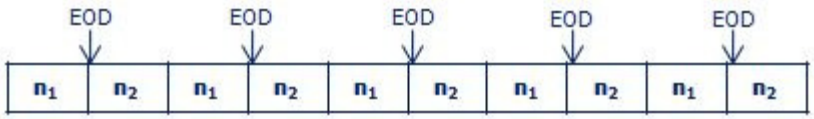
```
      <element-reference element="r5" type="in" />
      <element-reference element="r6" type="in" />
    </port-definition>
  - <state-transition-element id="r0" symbol-set="{1:7}" start="start-of-data">
      <activate-on-match element="l0" />
    </state-transition-element>
  - <state-transition-element id="r1" symbol-set="{2:7}" start="start-of-data">
      <activate-on-match element="l1" />
    </state-transition-element>
  - <state-transition-element id="r2" symbol-set="{3:7}" start="start-of-data">
      <activate-on-match element="l2" />
    </state-transition-element>
  - <state-transition-element id="r3" symbol-set="{4:7}" start="start-of-data">
      <activate-on-match element="l3" />
    </state-transition-element>
  - <state-transition-element id="r4" symbol-set="{5:7}" start="start-of-data">
      <activate-on-match element="l4" />
    </state-transition-element>
  - <state-transition-element id="r5" symbol-set="{6:7}" start="start-of-data">
      <activate-on-match element="l5" />
    </state-transition-element>
  - <state-transition-element id="r6" symbol-set="{7}" start="start-of-data">
      <activate-on-match element="l6" />
    </state-transition-element>
  - <state-transition-element id="l0" symbol-set="{0}">
      <activate-on-match element="or1" />
    </state-transition-element>
  - <state-transition-element id="l1" symbol-set="{1}">
      <activate-on-match element="or1" />
    </state-transition-element>
  - <state-transition-element id="l2" symbol-set="{2}">
      <activate-on-match element="or1" />
    </state-transition-element>
  - <state-transition-element id="l3" symbol-set="{3}">
      <activate-on-match element="or1" />
    </state-transition-element>
  - <state-transition-element id="l4" symbol-set="{4}">
      <activate-on-match element="or1" />
    </state-transition-element>
  - <state-transition-element id="l5" symbol-set="{5}">
      <activate-on-match element="or1" />
    </state-transition-element>
  - <state-transition-element id="l6" symbol-set="{6}">
      <activate-on-match element="or1" />
    </state-transition-element>
  - <or id="or1">
      <!--  first number is greater than second number   -->
      <report-on-high />
    </or>
  </macro>
- <macro name="0-7 n2 > n1 comparator" id="cp03">
  - <port-definition>
      <element-reference element="l0" type="in" />
      <element-reference element="l1" type="in" />
      <element-reference element="l2" type="in" />
      <element-reference element="l3" type="in" />
```

```
      <element-reference element="l4" type="in" />
      <element-reference element="l5" type="in" />
      <element-reference element="l6" type="in" />
    </port-definition>
  - <state-transition-element id="l0" symbol-set="{0}" start="start-of-data">
      <activate-on-match element="r0" />
    </state-transition-element>
  - <state-transition-element id="l1" symbol-set="{1}" start="start-of-data">
      <activate-on-match element="r1" />
    </state-transition-element>
  - <state-transition-element id="l2" symbol-set="{2}" start="start-of-data">
      <activate-on-match element="r2" />
    </state-transition-element>
  - <state-transition-element id="l3" symbol-set="{3}" start="start-of-data">
      <activate-on-match element="r3" />
    </state-transition-element>
  - <state-transition-element id="l4" symbol-set="{4}" start="start-of-data">
      <activate-on-match element="r4" />
    </state-transition-element>
  - <state-transition-element id="l5" symbol-set="{5}" start="start-of-data">
      <activate-on-match element="r5" />
    </state-transition-element>
  - <state-transition-element id="l6" symbol-set="{6}" start="start-of-data">
      <activate-on-match element="r6" />
    </state-transition-element>
  - <state-transition-element id="r0" symbol-set="{1:7}">
      <activate-on-match element="or1" />
    </state-transition-element>
  - <state-transition-element id="r1" symbol-set="{2:7}">
      <activate-on-match element="or1" />
    </state-transition-element>
  - <state-transition-element id="r2" symbol-set="{3:7}">
      <activate-on-match element="or1" />
    </state-transition-element>
  - <state-transition-element id="r3" symbol-set="{4:7}">
      <activate-on-match element="or1" />
    </state-transition-element>
  - <state-transition-element id="r4" symbol-set="{5:7}">
      <activate-on-match element="or1" />
    </state-transition-element>
  - <state-transition-element id="r5" symbol-set="{6:7}">
      <activate-on-match element="or1" />
    </state-transition-element>
  - <state-transition-element id="r6" symbol-set="{7}">
      <activate-on-match element="or1" />
    </state-transition-element>
  - <or id="or1">
      <!-- first number is less than second number  -->
      <report-on-high />
    </or>
  </macro>
</automata-network>
```

Comparator, 3 bit with end-of-data to eliminate spurious matches

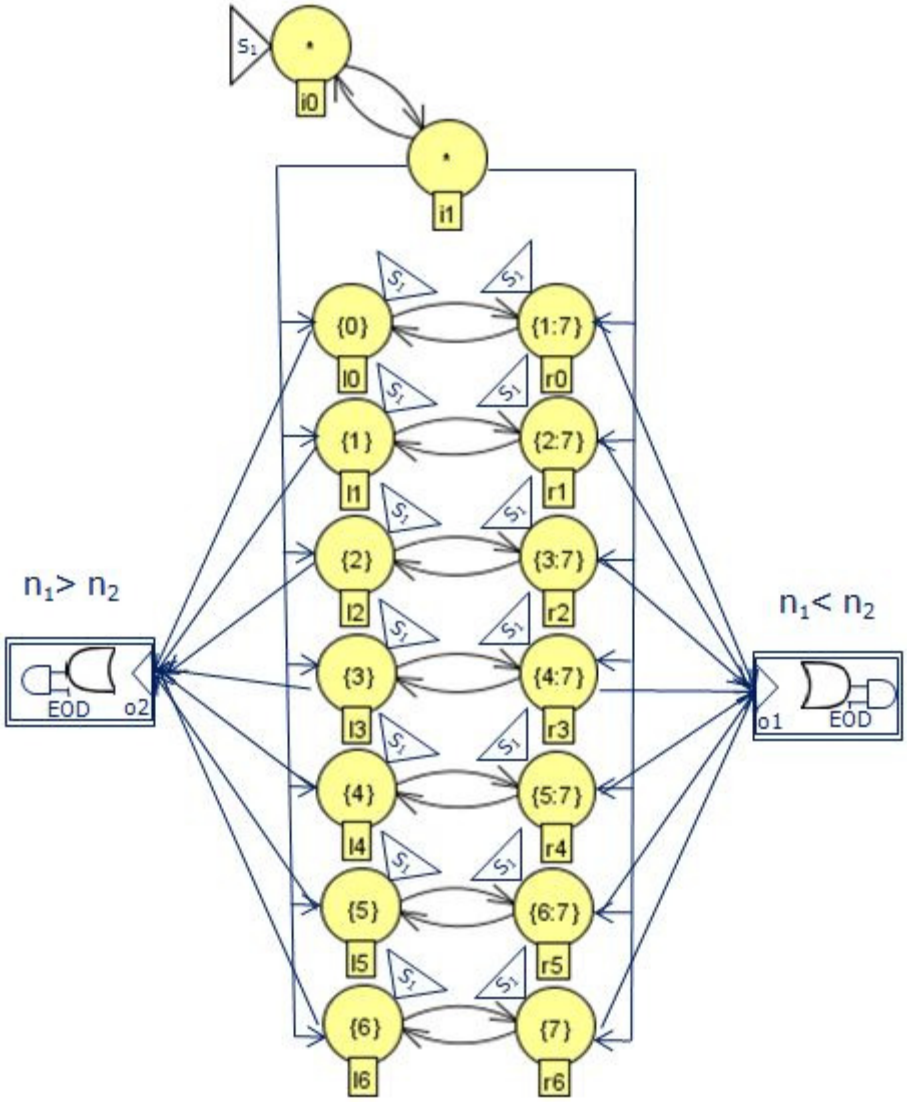
Comparator, 3 bit with EOD to eliminate spurious matches

This automaton is almost identical to the first 3-bit comparator except that it requires an end-of-data signal to be asserted on each even symbol cycle to function correctly. This operation may or may not be supported by the API for a particular implementation. The numbers to be compared are presented in pairs, as before, but with the addition that EOD must be asserted with the second number, as shown below.



The advantage of this design is that, like the preceding one with macros for greater-than, equal, and less-than conditions, it does not generate any spurious matches on the odd symbol cycle.

ANML-G Macro



ANML Template

This automaton is identical to the first 3-bit comparator except for the two OR elements. These have high-only-on-eod enabled.

```
<?xml version="1.0" ?>
- <macro name="Comparator, 3 bit" id="m01"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="file:///c:/ANML/cookbook/ANML_v4.xsd">
  - <state-transition-element id="i0" symbol-set="*" start="start-of-data">
      <activate-on-match element="i1" />
    </state-transition-element>
  - <state-transition-element id="i1" symbol-set="*">
      <activate-on-match element="i0" />
      <activate-on-match element="l0" />
      <activate-on-match element="l1" />
      <activate-on-match element="l2" />
      <activate-on-match element="l3" />
      <activate-on-match element="l4" />
      <activate-on-match element="l5" />
      <activate-on-match element="l6" />
      <activate-on-match element="r0" />
      <activate-on-match element="r1" />
      <activate-on-match element="r2" />
      <activate-on-match element="r3" />
      <activate-on-match element="r4" />
      <activate-on-match element="r5" />
      <activate-on-match element="r6" />
    </state-transition-element>
  - <state-transition-element id="l0" symbol-set="{0}" start="start-of-data">
      <activate-on-match element="r0" />
      <activate-on-match element="o2" />
    </state-transition-element>
  - <state-transition-element id="l1" symbol-set="{1}" start="start-of-data">
      <activate-on-match element="r1" />
      <activate-on-match element="o2" />
    </state-transition-element>
  - <state-transition-element id="l2" symbol-set="{2}" start="start-of-data">
      <activate-on-match element="r2" />
      <activate-on-match element="o2" />
    </state-transition-element>
  - <state-transition-element id="l3" symbol-set="{3}" start="start-of-data">
      <activate-on-match element="r3" />
      <activate-on-match element="o2" />
    </state-transition-element>
  - <state-transition-element id="l4" symbol-set="{4}" start="start-of-data">
      <activate-on-match element="r4" />
      <activate-on-match element="o2" />
    </state-transition-element>
  - <state-transition-element id="l5" symbol-set="{5}" start="start-of-data">
      <activate-on-match element="r5" />
      <activate-on-match element="o2" />
    </state-transition-element>
  - <state-transition-element id="l6" symbol-set="{6}" start="start-of-data">
      <activate-on-match element="r6" />
      <activate-on-match element="o2" />
    </state-transition-element>
  - <state-transition-element id="r0" symbol-set="{1:7}" start="start-of-data">
      <activate-on-match element="l0" />
      <activate-on-match element="o1" />
    </state-transition-element>
```

```
- <state-transition-element id="r1" symbol-set="{2:7}" start="start-of-data">
    <activate-on-match element="l1" />
    <activate-on-match element="o1" />
  </state-transition-element>
- <state-transition-element id="r2" symbol-set="{3:7}" start="start-of-data">
    <activate-on-match element="l2" />
    <activate-on-match element="o1" />
  </state-transition-element>
- <state-transition-element id="r3" symbol-set="{4:7}" start="start-of-data">
    <activate-on-match element="l3" />
    <activate-on-match element="o1" />
  </state-transition-element>
- <state-transition-element id="r4" symbol-set="{5:7}" start="start-of-data">
    <activate-on-match element="l4" />
    <activate-on-match element="o1" />
  </state-transition-element>
- <state-transition-element id="r5" symbol-set="{6:7}" start="start-of-data">
    <activate-on-match element="l5" />
    <activate-on-match element="o1" />
  </state-transition-element>
- <state-transition-element id="r6" symbol-set="{7}" start="start-of-data">
    <activate-on-match element="l6" />
    <activate-on-match element="o1" />
  </state-transition-element>
- <or id="o2" high-only-on-eod="true">
    <!-- first number is greater than second number -->
    <report-on-high />
  </or>
- <or id="o1" high-only-on-eod="true">
    <!-- first number is less than second number -->
    <report-on-high />
  </or>
</macro>
```

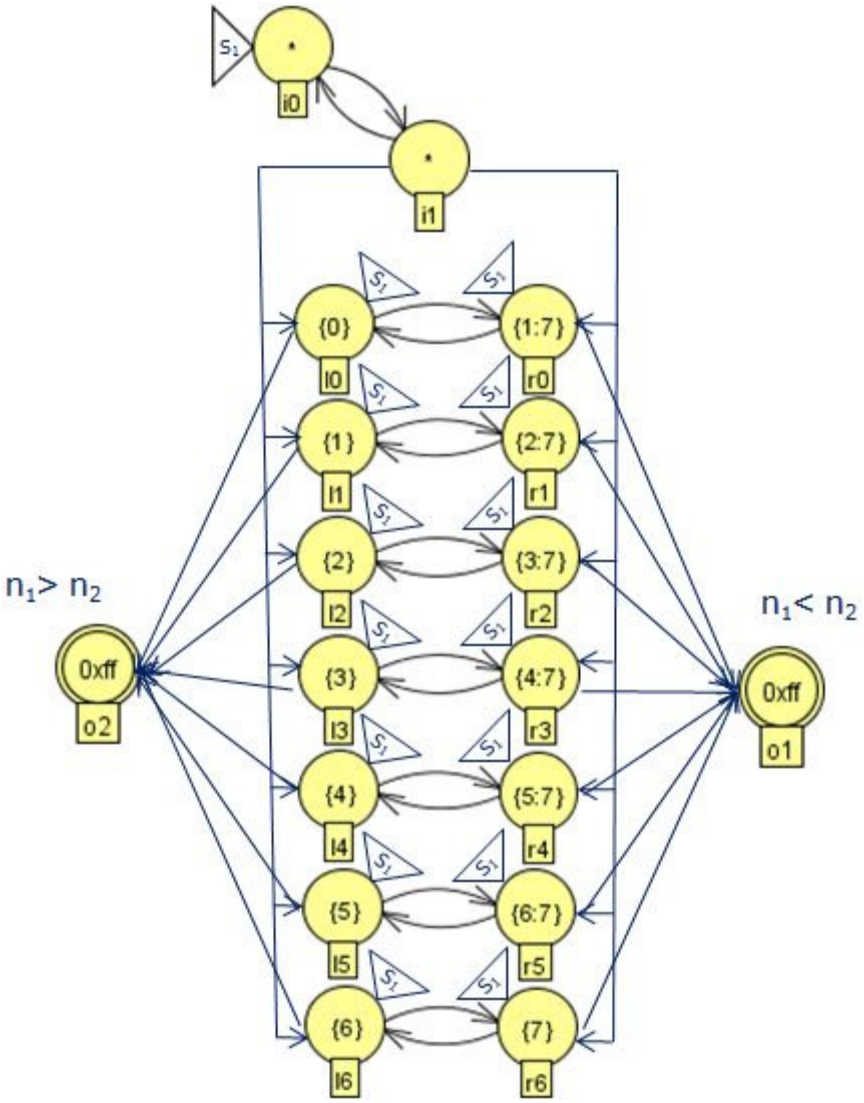
Comparator, 3 bit using STE implicit OR instead of OR element

Comparator, 3 bit using State-Transition-Element implicit OR

This automaton is similiar to the previous comparator that used EOD to eliminate spurious matches except that instead of an EOD signal it expects a terminator symbol to be placed between number pairs as shown below.

n_1	n_2	0xff	n_1	n_2	0xff	n_1	n_2	0xff	n_1

ANML-G Macro



ANML Template

The automaton uses the fact that there is an implicit OR in the state-transition-element, enabling it be activated by multiple connected state-transition-elements in a single symbol cycle, allowing the state-transition-element to replace an OR element. This may be helpful if OR elements are limited and state-transition-elements less so. The disadvantage of this design is that the state-transition-element must consume an input symbol so the input buffer must be arranged so that there is a "throw-away" symbol to trigger the operation of the state-transition-element.

```
<?xml version="1.0" ?>
- <macro name="Comparator, 3 bit" id="m01"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="file:///c:/ANML/cookbook/ANML_v4.xsd">
  - <state-transition-element id="i0" symbol-set="*" start="start-of-data">
      <activate-on-match element="i1" />
    </state-transition-element>
  - <state-transition-element id="i1" symbol-set="*">
      <activate-on-match element="i0" />
      <activate-on-match element="l0" />
      <activate-on-match element="l1" />
      <activate-on-match element="l2" />
      <activate-on-match element="l3" />
      <activate-on-match element="l4" />
      <activate-on-match element="l5" />
      <activate-on-match element="l6" />
      <activate-on-match element="r0" />
      <activate-on-match element="r1" />
      <activate-on-match element="r2" />
      <activate-on-match element="r3" />
      <activate-on-match element="r4" />
      <activate-on-match element="r5" />
      <activate-on-match element="r6" />
    </state-transition-element>
  - <state-transition-element id="l0" symbol-set="{0}" start="start-of-data">
      <activate-on-match element="r0" />
      <activate-on-match element="o2" />
    </state-transition-element>
  - <state-transition-element id="l1" symbol-set="{1}" start="start-of-data">
      <activate-on-match element="r1" />
      <activate-on-match element="o2" />
    </state-transition-element>
  - <state-transition-element id="l2" symbol-set="{2}" start="start-of-data">
      <activate-on-match element="r2" />
      <activate-on-match element="o2" />
    </state-transition-element>
  - <state-transition-element id="l3" symbol-set="{3}" start="start-of-data">
      <activate-on-match element="r3" />
      <activate-on-match element="o2" />
    </state-transition-element>
  - <state-transition-element id="l4" symbol-set="{4}" start="start-of-data">
      <activate-on-match element="r4" />
      <activate-on-match element="o2" />
    </state-transition-element>
  - <state-transition-element id="l5" symbol-set="{5}" start="start-of-data">
      <activate-on-match element="r5" />
      <activate-on-match element="o2" />
    </state-transition-element>
  - <state-transition-element id="l6" symbol-set="{6}" start="start-of-data">
      <activate-on-match element="r6" />
      <activate-on-match element="o2" />
    </state-transition-element>
  - <state-transition-element id="r0" symbol-set="{1:7}" start="start-of-data">
      <activate-on-match element="l0" />
      <activate-on-match element="o1" />
    </state-transition-element>
```

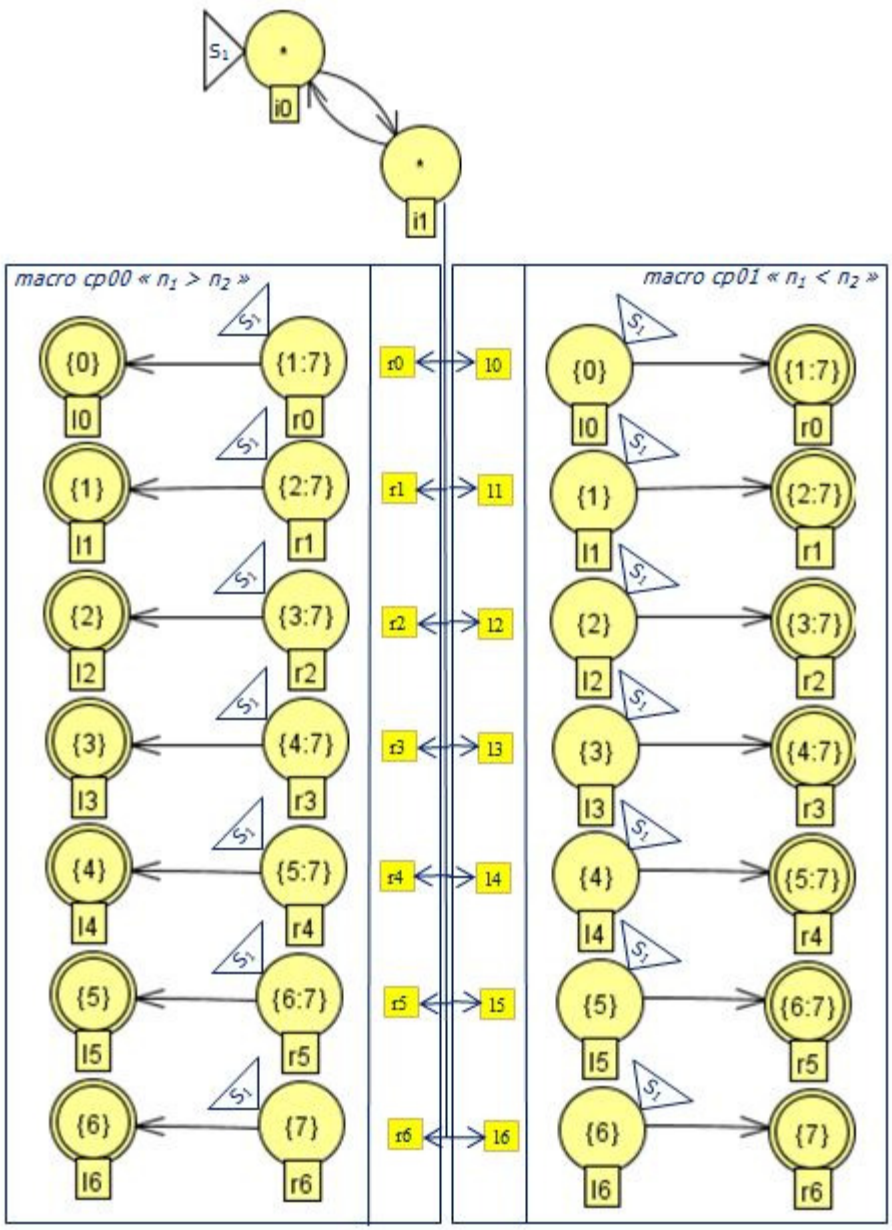
```
- <state-transition-element id="r1" symbol-set="{2:7}" start="start-of-data">
    <activate-on-match element="l1" />
    <activate-on-match element="o1" />
  </state-transition-element>
- <state-transition-element id="r2" symbol-set="{3:7}" start="start-of-data">
    <activate-on-match element="l2" />
    <activate-on-match element="o1" />
  </state-transition-element>
- <state-transition-element id="r3" symbol-set="{4:7}" start="start-of-data">
    <activate-on-match element="l3" />
    <activate-on-match element="o1" />
  </state-transition-element>
- <state-transition-element id="r4" symbol-set="{5:7}" start="start-of-data">
    <activate-on-match element="l4" />
    <activate-on-match element="o1" />
  </state-transition-element>
- <state-transition-element id="r5" symbol-set="{6:7}" start="start-of-data">
    <activate-on-match element="l5" />
    <activate-on-match element="o1" />
  </state-transition-element>
- <state-transition-element id="r6" symbol-set="{7}" start="start-of-data">
    <activate-on-match element="l6" />
    <activate-on-match element="o1" />
  </state-transition-element>
- <state-transition-element id="o2" symbol-set="0xff">
    <!-- first number is greater than second number -->
    <report-on-match />
  </state-transition-element>
- <state-transition-element id="o1" symbol-set="0xff">
    <!-- first number is less than second number -->
    <report-on-match />
  </state-transition-element>
</macro>
```

Comparator, 3 bit without OR elements

Comparator, 3-bit without OR Elements

This automaton does not use OR elements to report comparison output. Output may come from one (and only one in this design) of 14 state-transition-elements. While this automaton requires fewer elements than the comparable version with ORs, it has the disadvantage that the application must correlate the output from a larger number of elements with the results obtained from the comparison.

ANML-G Macro



ANML Template

This automaton provides an alternative to equivalent designs using OR elements and may be useful when it is important to conserve ORs. It may also be less routing resource intensive, although this depends on balance between available activation connections and output connections (it uses less activations and more output). This design is a variant of the 3-bit comparator using greater-than, less-than and equal macros producing no spurious output, although in this automaton there is no equal macro so the equal condition is inferred when there is no other output on an even symbol cycle.

```
<?xml version="1.0" ?>
- <automata-network name="an01" id="3 bit comparator without OR elements"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="file:///c:/ANML/cookbook/ANML_v4.xsd">
  - <state-transition-element id="i0" symbol-set="*" start="start-of-data">
      <activate-on-match element="i1" />
    </state-transition-element>
  - <state-transition-element id="i1" symbol-set="*">
      <activate-on-match element="i0" />
      <activate-on-match macro="cp00" element="r0" />
      <activate-on-match macro="cp00" element="r1" />
      <activate-on-match macro="cp00" element="r2" />
      <activate-on-match macro="cp00" element="r3" />
      <activate-on-match macro="cp00" element="r4" />
      <activate-on-match macro="cp00" element="r5" />
      <activate-on-match macro="cp00" element="r6" />
      <activate-on-match macro="cp01" element="l0" />
      <activate-on-match macro="cp01" element="l1" />
      <activate-on-match macro="cp01" element="l2" />
      <activate-on-match macro="cp01" element="l3" />
      <activate-on-match macro="cp01" element="l4" />
      <activate-on-match macro="cp01" element="l5" />
      <activate-on-match macro="cp01" element="l6" />
    </state-transition-element>
  - <macro name="n1 > n2" id="cp00">
    - <port-definition>
        <element-reference element="r0" type="in" />
        <element-reference element="r1" type="in" />
        <element-reference element="r2" type="in" />
        <element-reference element="r3" type="in" />
        <element-reference element="r4" type="in" />
        <element-reference element="r5" type="in" />
        <element-reference element="r6" type="in" />
      </port-definition>
    - <state-transition-element id="r0" symbol-set="{1:7}" start="start-of-data">
        <activate-on-match element="l0" />
      </state-transition-element>
    - <state-transition-element id="r1" symbol-set="{2:7}" start="start-of-data">
        <activate-on-match element="l1" />
      </state-transition-element>
    - <state-transition-element id="r2" symbol-set="{3:7}" start="start-of-data">
        <activate-on-match element="l2" />
      </state-transition-element>
    - <state-transition-element id="r3" symbol-set="{4:7}" start="start-of-data">
        <activate-on-match element="l3" />
      </state-transition-element>
    - <state-transition-element id="r4" symbol-set="{5:7}" start="start-of-data">
        <activate-on-match element="l4" />
      </state-transition-element>
    - <state-transition-element id="r5" symbol-set="{6:7}" start="start-of-data">
        <activate-on-match element="l5" />
      </state-transition-element>
    - <state-transition-element id="r6" symbol-set="{7}" start="start-of-data">
        <activate-on-match element="l6" />
      </state-transition-element>
    - <state-transition-element id="l0" symbol-set="{0}">
```

```
      <report-on-match />
    </state-transition-element>
  - <state-transition-element id="I1" symbol-set="{1}">
      <report-on-match />
    </state-transition-element>
  - <state-transition-element id="I2" symbol-set="{2}">
      <report-on-match />
    </state-transition-element>
  - <state-transition-element id="I3" symbol-set="{3}">
      <report-on-match />
    </state-transition-element>
  - <state-transition-element id="I4" symbol-set="{4}">
      <report-on-match />
    </state-transition-element>
  - <state-transition-element id="I5" symbol-set="{5}">
      <report-on-match />
    </state-transition-element>
  - <state-transition-element id="I6" symbol-set="{6}">
      <report-on-match />
    </state-transition-element>
  </macro>
- <macro name="n1 < n2 comparator" id="cp01">
  - <port-definition>
      <element-reference element="I0" type="in" />
      <element-reference element="I1" type="in" />
      <element-reference element="I2" type="in" />
      <element-reference element="I3" type="in" />
      <element-reference element="I4" type="in" />
      <element-reference element="I5" type="in" />
      <element-reference element="I6" type="in" />
    </port-definition>
  - <state-transition-element id="I0" symbol-set="{0}" start="start-of-data">
      <activate-on-match element="r0" />
    </state-transition-element>
  - <state-transition-element id="I1" symbol-set="{1}" start="start-of-data">
      <activate-on-match element="r1" />
    </state-transition-element>
  - <state-transition-element id="I2" symbol-set="{2}" start="start-of-data">
      <activate-on-match element="r2" />
    </state-transition-element>
  - <state-transition-element id="I3" symbol-set="{3}" start="start-of-data">
      <activate-on-match element="r3" />
    </state-transition-element>
  - <state-transition-element id="I4" symbol-set="{4}" start="start-of-data">
      <activate-on-match element="r4" />
    </state-transition-element>
  - <state-transition-element id="I5" symbol-set="{5}" start="start-of-data">
      <activate-on-match element="r5" />
    </state-transition-element>
  - <state-transition-element id="I6" symbol-set="{6}" start="start-of-data">
      <activate-on-match element="r6" />
    </state-transition-element>
  - <state-transition-element id="r0" symbol-set="{1:7}">
      <report-on-match />
    </state-transition-element>
  - <state-transition-element id="r1" symbol-set="{2:7}">
```

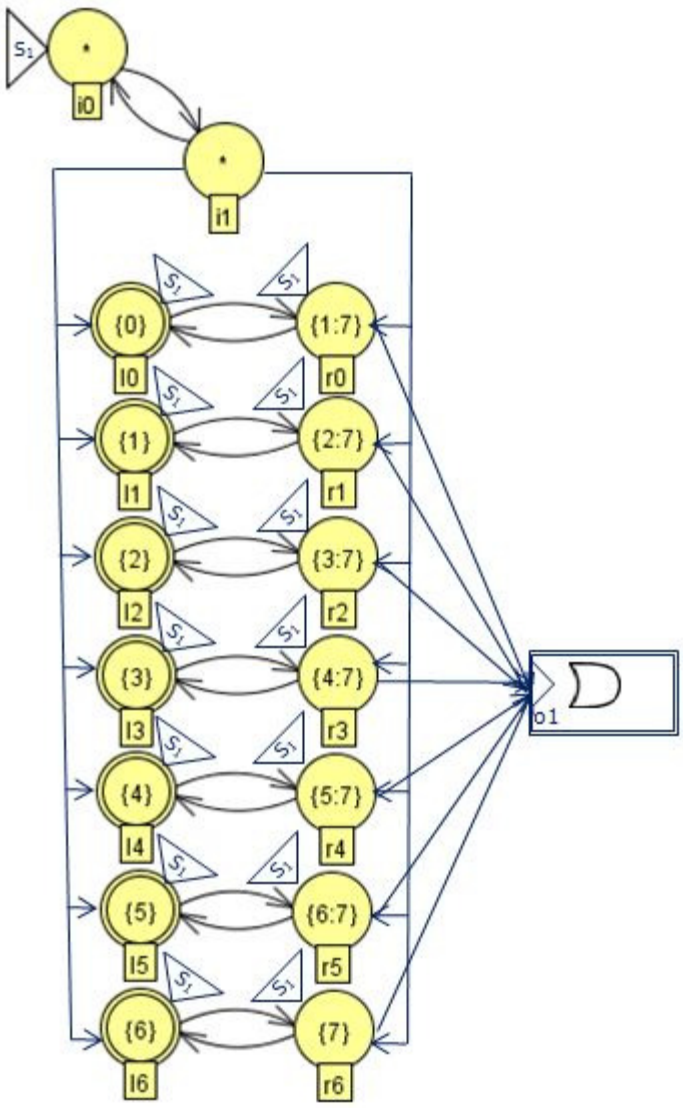
```
      <report-on-match />
    </state-transition-element>
  – <state-transition-element id="r2" symbol-set="{3:7}">
      <report-on-match />
    </state-transition-element>
  – <state-transition-element id="r3" symbol-set="{4:7}">
      <report-on-match />
    </state-transition-element>
  – <state-transition-element id="r4" symbol-set="{5:7}">
      <report-on-match />
    </state-transition-element>
  – <state-transition-element id="r5" symbol-set="{6:7}">
      <report-on-match />
    </state-transition-element>
  – <state-transition-element id="r6" symbol-set="{7}">
      <report-on-match />
    </state-transition-element>
  </macro>
</automata-network>
```

Comparator, 3 bit with only one OR element to reduce spurious matches

Comparator, 3-bit with only 1 OR element to limit but not eliminate spurious matches

This automaton is based on the first 3-bit comparator which uses 2 OR elements and can have up to 2 spurious matches on odd symbol cycles but eliminates one of the OR elements while still limiting the number of possible spurious matches per odd symbol cycle to 2. The n1 > n2 comparison reports results from one and only one output generating state-transition-element instead of the OR element on the left-hand side. No more than one of the left-hand side state-transition-elements can be true on the odd symbol cycles so the left-hand side can only report one result as with the two OR element solution. The application must be able to derive the comparison result from a larger set of possible reporting elements than with the two OR element solution.

ANML-G Macro



ANML Template

This automaton provides yet another design alternative which may be useful under specific implementation constraints.

```
<?xml version="1.0" ?>
- <macro name="Comparator, 3 bit with 1 OR" id="m01"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="file:///c:/ANML/cookbook/ANML_v4.xsd">
  - <state-transition-element id="i0" symbol-set="*" start="start-of-data">
      <activate-on-match element="i1" />
    </state-transition-element>
  - <state-transition-element id="i1" symbol-set="*">
      <activate-on-match element="i0" />
      <activate-on-match element="l0" />
      <activate-on-match element="l1" />
      <activate-on-match element="l2" />
      <activate-on-match element="l3" />
      <activate-on-match element="l4" />
      <activate-on-match element="l5" />
      <activate-on-match element="l6" />
      <activate-on-match element="r0" />
      <activate-on-match element="r1" />
      <activate-on-match element="r2" />
      <activate-on-match element="r3" />
      <activate-on-match element="r4" />
      <activate-on-match element="r5" />
      <activate-on-match element="r6" />
    </state-transition-element>
  - <state-transition-element id="l0" symbol-set="{0}" start="start-of-data">
      <report-on-match />
      <activate-on-match element="r0" />
    </state-transition-element>
  - <state-transition-element id="l1" symbol-set="{1}" start="start-of-data">
      <report-on-match />
      <activate-on-match element="r1" />
    </state-transition-element>
  - <state-transition-element id="l2" symbol-set="{2}" start="start-of-data">
      <report-on-match />
      <activate-on-match element="r2" />
    </state-transition-element>
  - <state-transition-element id="l3" symbol-set="{3}" start="start-of-data">
      <report-on-match />
      <activate-on-match element="r3" />
    </state-transition-element>
  - <state-transition-element id="l4" symbol-set="{4}" start="start-of-data">
      <report-on-match />
      <activate-on-match element="r4" />
    </state-transition-element>
  - <state-transition-element id="l5" symbol-set="{5}" start="start-of-data">
      <report-on-match />
      <activate-on-match element="r5" />
    </state-transition-element>
  - <state-transition-element id="l6" symbol-set="{6}" start="start-of-data">
      <report-on-match />
      <activate-on-match element="r6" />
    </state-transition-element>
  - <state-transition-element id="r0" symbol-set="{1:7}" start="start-of-data">
      <activate-on-match element="l0" />
      <activate-on-match element="o1" />
    </state-transition-element>
```

```
- <state-transition-element id="r1" symbol-set="{2:7}" start="start-of-data">
    <activate-on-match element="l1" />
    <activate-on-match element="o1" />
  </state-transition-element>
- <state-transition-element id="r2" symbol-set="{3:7}" start="start-of-data">
    <activate-on-match element="l2" />
    <activate-on-match element="o1" />
  </state-transition-element>
- <state-transition-element id="r3" symbol-set="{4:7}" start="start-of-data">
    <activate-on-match element="l3" />
    <activate-on-match element="o1" />
  </state-transition-element>
- <state-transition-element id="r4" symbol-set="{5:7}" start="start-of-data">
    <activate-on-match element="l4" />
    <activate-on-match element="o1" />
  </state-transition-element>
- <state-transition-element id="r5" symbol-set="{6:7}" start="start-of-data">
    <activate-on-match element="l5" />
    <activate-on-match element="o1" />
  </state-transition-element>
- <state-transition-element id="r6" symbol-set="{7}" start="start-of-data">
    <activate-on-match element="l6" />
    <activate-on-match element="o1" />
  </state-transition-element>
- <or id="o1">
    <!--  first number is less than second number   -->
    <report-on-high />
  </or>
</macro>
```

Characterize ASCII stream, Output on Stream Terminator

Characterize ASCII Stream, Output on Stream Terminator

This automaton examines an ASCII stream, reporting on the characteristics it identifies only once it sees the stream terminator. The automaton shows one way to report matches only once even though the match condition might have been met many times as the input is examined and also how to report matches all on the same cycle.

The input to this automaton is an ASCII stream with values from 0x00 to 0xFE. The last ASCII value, 0xFF, is reserved in this example for signaling stream termination.

The automaton can detect 9 different characteristics:

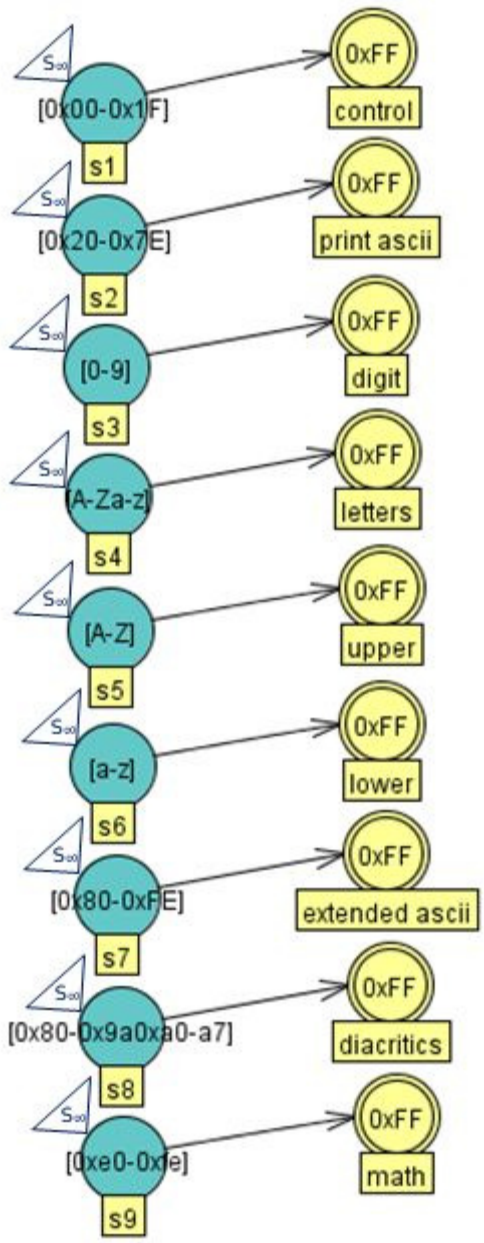
1. The stream contains control characters
2. The stream contains printable ASCII characters
3. The stream contains digits
4. The stream contains English letters
5. The stream contains upper case English letters
6. The stream contains lower case English letters
7. The stream contains extended ASCII symbols
8. The stream contains letters with diacritics
9. The stream contains math symbols

For example, if presented with the stream

münchen12230xff

(where 0xff is a single symbol with the decimal value of 255) the automaton will report the stream contains printable ASCII characters, digits, English letter, lower case English letters, extended ASCII symbols, and letters with diacritics once it sees the final 0xff.

ANML-G Macro



ANML Template

The state-transition-elements that detect the characteristics of the stream each examine every input symbol. They are all latched state-transition-elements so that if they do detect a characteristic once they will continue to assert activate-on-match on every symbol cycle. Each characteristic-detecting state-transition-element has an activate-on-match to a report generating state-transition-element that recognizes the end of stream marker. Since all end-of-stream detecting state-transition-elements will be continuously activated once their corresponding characteristic has been detected, all detected characteristics will be output on the same, final symbol cycle.

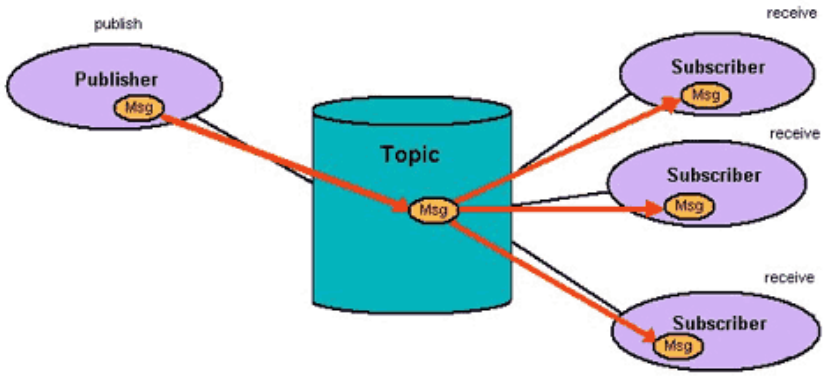
```
<?xml version="1.0" ?>
- <macro name="Characterize ASCII string, report on end-of-stream" id="mx00"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="file:///c:/ANML/cookbook/ANML_v4.xsd">
  - <state-transition-element id="s1" symbol-set="[0x00-0x1f]" start="all-input" latch="true">
      <activate-on-match element="control" />
    </state-transition-element>
  - <state-transition-element id="control" symbol-set="[0xff]">
      <report-on-match />
    </state-transition-element>
  - <state-transition-element id="s2" symbol-set="[0x20-0x7e]" start="all-input" latch="true">
      <activate-on-match element="print ascii" />
    </state-transition-element>
  - <state-transition-element id="print ascii" symbol-set="[0xff]">
      <report-on-match />
    </state-transition-element>
  - <state-transition-element id="s3" symbol-set="[0-9]" start="all-input" latch="true">
      <activate-on-match element="digit" />
    </state-transition-element>
  - <state-transition-element id="digit" symbol-set="[0xff]">
      <report-on-match />
    </state-transition-element>
  - <state-transition-element id="s4" symbol-set="[A-Za-z]" start="all-input" latch="true">
      <activate-on-match element="letters" />
    </state-transition-element>
  - <state-transition-element id="letters" symbol-set="[0xff]">
      <report-on-match />
    </state-transition-element>
  - <state-transition-element id="s5" symbol-set="[A-Z]" start="all-input" latch="true">
      <activate-on-match element="upper" />
    </state-transition-element>
  - <state-transition-element id="upper" symbol-set="[0xff]">
      <report-on-match />
    </state-transition-element>
  - <state-transition-element id="s6" symbol-set="[a-z]" start="all-input" latch="true">
      <activate-on-match element="lower" />
    </state-transition-element>
  - <state-transition-element id="lower" symbol-set="[0xff]">
      <report-on-match />
    </state-transition-element>
  - <state-transition-element id="s7" symbol-set="[0x80-0xfe]" start="all-input" latch="true">
      <activate-on-match element="extended ascii" />
    </state-transition-element>
  - <state-transition-element id="extended ascii" symbol-set="[0xff]">
      <report-on-match />
    </state-transition-element>
  - <state-transition-element id="s8" symbol-set="[0x80-0x9a0xa0-0xa7]" start="all-input"
      latch="true">
      <activate-on-match element="diacritics" />
    </state-transition-element>
  - <state-transition-element id="diacritics" symbol-set="[0xff]">
      <report-on-match />
    </state-transition-element>
  - <state-transition-element id="s9" symbol-set="[0xe0-0xfe]" start="all-input" latch="true">
      <activate-on-match element="math" />
    </state-transition-element>
  - <state-transition-element id="math" symbol-set="[0xff]">
      <report-on-match />
    </state-transition-element>
</macro>
```

PubSub

PubSub

PubSub, Publish-Subscribe (Wikipedia article here), is a widely used communication paradigm in computing systems. PubSub is a highly decoupled architecture where the information producer (publisher) does not specify the destination for the information and information consumers (subscribers) receive information based on examination of the information content, determining, through a mediating process (PubSub engine) if it meets criteria for an object of interest. The publisher sends the information to the PubSub engine and the PubSub engine resends the information to the subscribers. The pubsub engine also receives the subscribers subscription criteria which it uses when deciding whether or not to send an information object to the subscriber.

The image below shows a high-level view of a PubSub system. Msg is sent by the Publisher to Topic, the PubSub engine and Topic distributes the Msg to 3 Subscribers whose subscription criteria is satisified by the content of Msg.



PubSub is at the core at RSS and Atom (web feeds), Jabber and XMPP, JMS (message-oriented middleware). Moreover, pubsub concepts are used in an enormous number of event-driven computing systems and is a very general architectural pattern in computing. The importance of pubsub is growing as the computing landscape supports massive numbers of decoupled heterogenous devices and services grow increasingly personalized.

The ANML automaton implements a PubSub engine. The information input to the engine is simply a symbol stream terminated by the value 0xFF and followed by two additional characters. The symbol stream may be an article, a web page, a document, a stock feed, in short, whatever an information object is defined to be in the application. The subscriber criteria is expressed in a simple query language. Here is an example of a query:

```
OR(  AND(long-term effects,lack of sunlight),  AND(Micron,stock price),   Dilbert)
```

Subscribers using this query want to receive articles that contain either both the terms "long-term effects" and "lack of sunlight" or both the terms "Micron" and "stock price" or the term Dilbert.

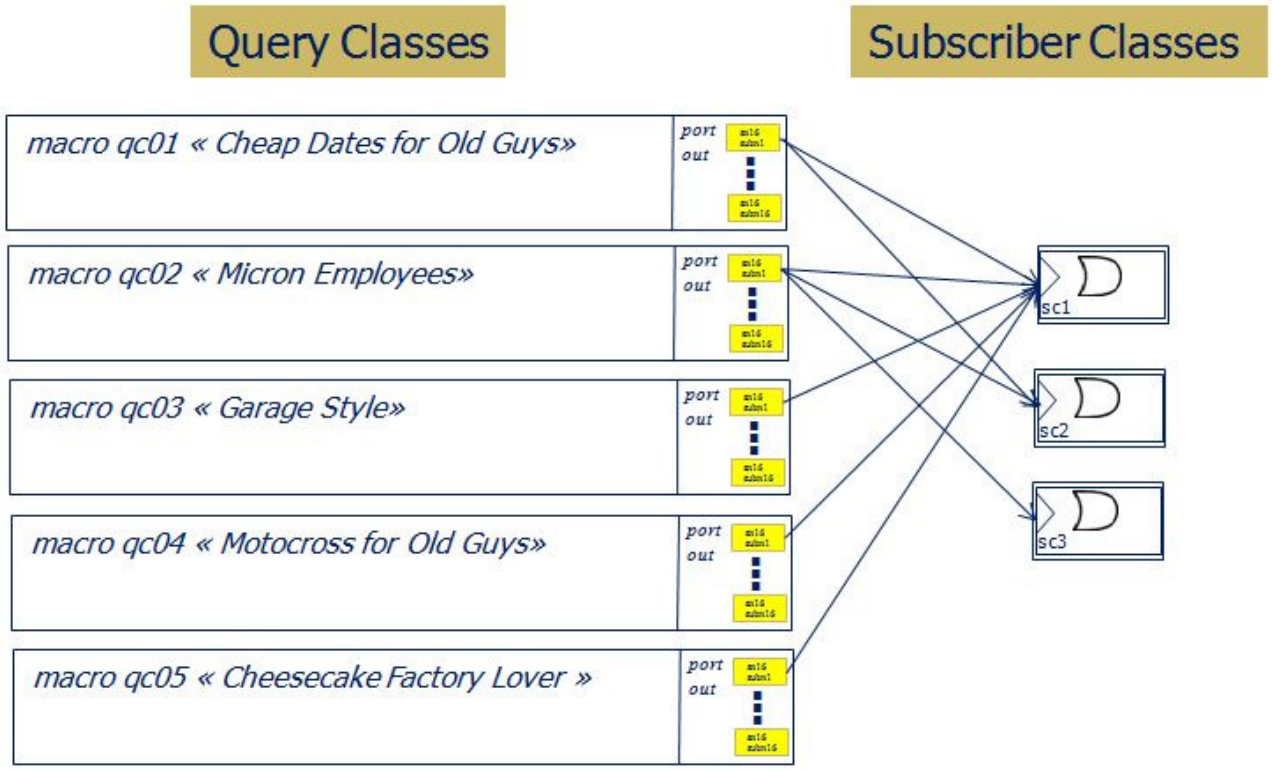
In BNF the definition of this language is:

<query-class> ::= <boolean-expression> | <term>
<term> ::= <term> "," <boolean-expression> | <boolean-expression> "," <term> | <term> "," <term> | <text>
<boolean-expression> ::= <operation> "(" <term> ")"
<operation> ::= "AND" | "OR" | "NOR" | "NAND"
<text> ::= per definition of symbol-set attribute multiple

In typical usage in a PubSub system there may be a fixed set of query terms or there will be a great deal of commonality in the terms that subscribers choose. Likewise, there may be a fixed set of possible queries or a great deal of commonality in the queries that subscribers form. In the PubSub automaton terms are independant of the queries and queries are placed in classes that subscribers share. In a sense, the query classes subscribe to the terms and subscribers subscribe to the query classes. In our example, when a subscriber subscribes to multiple query classes it is treated as an implicit OR - if the information object is matched by any of the query classes the information object will be sent to the subscriber.

ANML-G Macro

The image below shows a higher level view of the example. Links are provided below the image to see the detail view of each query class.



Query Classes

- qc01 Cheap Dates for Old Guys
- qc02 Micron Employees
- qc03 Garage Style
- qc04 Motocross for Old Guys
- qc05 Cheesecake Factory Lover

ANML Template

The criteria used to determine which subscribers receive an information object is programmed into the automaton in a modular fashion. There are three types of modules: terms, query classes, and subscriber

classes. In this example the terms are specific to each query class and are nested inside the query class macro. A more general implementation would keep the terms exterior to the query classes so that terms could be more easily shared among query classes. Subscribers are outside the query class macros, in the automata network. Subscribers may also be treated as classes, as more than one subscriber may have identical subscriptions. The application would therefore associated the output event in the subscriber class with a list of subscribers. Each subscriber class selects whatever query classes it wishes to subscribe to, implemented as an OR element since matches any of the subscriber's query classes should cause the information to be delivered the subscribers in the subscriber class.

Each query class has a "subscriber network fanout 16" macro which enables subscription to the query class by a large number of subscriber classes. In effect, the fanout macro increases the potential number of subscribers to a query class by 16X. The same principle could be used, at the cost of one additional symbol cycle (an extra character added at the end of the article). For example, if the fanout of a state-transition-element is 16 the fanout macro increases the number of subscriber classes that may subscribe to a query class to 256. If each state-transition-element in the fanout macro were to feed another fanout macro the total number of possible subscriber classes would increase to 4096. The fan-in limitation of the subscriber class OR element will limit the number of query classes a subscriber class may subscribe to. This can be increased simply by add more OR elements associated with a given subscriber class.

The modular nature of the design of the PubSub automaton should enable it be scaled. Multiple chip implementations should be straightforward.

```
<?xml version="1.0" ?>
- <automata-network id="an01" name="PubSub Automata Network"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="ANML_v6.xsd">
  <!-- Subscriber Classes -->
  - <or id="sc1">
    <report-on-high />
  </or>
  - <or id="sc2">
    <report-on-high />
  </or>
  - <or id="sc3">
    <report-on-high />
  </or>
  <!-- Query Classes and Terms -->
  - <macro id="qc01" name="Cheap Dates for Old Guys">
    - <port-definition>
      <element-reference macro="sn16" element="subn1" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn2" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn3" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn4" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn5" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn6" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn7" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn8" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn9" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn10" type="state-
        transition-element" activation="out" />
      <element-reference macro="sn16" element="subn11" type="state-
        transition-element" activation="out" />
      <element-reference macro="sn16" element="subn12" type="state-
        transition-element" activation="out" />
      <element-reference macro="sn16" element="subn13" type="state-
        transition-element" activation="out" />
      <element-reference macro="sn16" element="subn14" type="state-
        transition-element" activation="out" />
      <element-reference macro="sn16" element="subn15" type="state-
        transition-element" activation="out" />
      <element-reference macro="sn16" element="subn16" type="state-
        transition-element" activation="out" />
    </port-definition>
    - <macro id="tx6-1" name="term 6 symbols">
      - <port-definition>
        <element-reference element="term" type="state-transition-element"
```

```
    start="start-of-data" />
      <element-reference element="eob" type="state-transition-element"
        activation="out" />
    </port-definition>
  - <state-transition-element id="term" symbol-set="dating" case-
      insensitive="true" start="start-of-data" latch="true">
      <activate-on-match element="eob" />
    </state-transition-element>
  - <state-transition-element id="eob" symbol-set="0xFF">
      <activate-on-match macro="q006" element="cycle1-1" />
    </state-transition-element>
  </macro>
- <macro id="tx4-1" name="term 4 symbols">
  - <port-definition>
      <element-reference element="term" type="state-transition-element"
        start="start-of-data" />
      <element-reference element="eob" type="state-transition-element"
        activation="out" />
    </port-definition>
  - <state-transition-element id="term" symbol-set="tips" case-insensitive="true"
      start="start-of-data" latch="true">
      <activate-on-match element="eob" />
    </state-transition-element>
  - <state-transition-element id="eob" symbol-set="0xFF">
      <activate-on-match macro="q006" element="q0-t1" />
    </state-transition-element>
  </macro>
- <macro id="tx9-1" name="term 9 symbols">
  - <port-definition>
      <element-reference element="term" type="state-transition-element"
        start="start-of-data" />
      <element-reference element="eob" type="state-transition-element"
        activation="out" />
    </port-definition>
  - <state-transition-element id="term" symbol-set="older men" case-
      insensitive="true" start="start-of-data" latch="true">
      <activate-on-match element="eob" />
    </state-transition-element>
  - <state-transition-element id="eob" symbol-set="0xFF">
      <activate-on-match macro="q006" element="q0-t1" />
    </state-transition-element>
  </macro>
- <macro id="tx4-2" name="term 4 symbols">
  - <port-definition>
      <element-reference element="term" type="state-transition-element"
        start="start-of-data" />
      <element-reference element="eob" type="state-transition-element"
        activation="out" />
    </port-definition>
  - <state-transition-element id="term" symbol-set="free" case-
      insensitive="true" start="start-of-data" latch="true">
      <activate-on-match element="eob" />
    </state-transition-element>
```

```
  - <state-transition-element id="eob" symbol-set="0xFF">
      <activate-on-match macro="q006" element="q0-t2" />
    </state-transition-element>
  </macro>
- <macro id="tx7-1" name="term 7 symbols">
  - <port-definition>
      <element-reference element="term" type="state-transition-element"
        start="start-of-data" />
      <element-reference element="eob" type="state-transition-element"
        activation="out" />
    </port-definition>
  - <state-transition-element id="term" symbol-set="on-line" case-
      insensitive="true" start="start-of-data" latch="true">
      <activate-on-match element="eob" />
    </state-transition-element>
  - <state-transition-element id="eob" symbol-set="0xFF">
      <activate-on-match macro="q006" element="q0-t2" />
    </state-transition-element>
  </macro>
- <macro id="tx10-1" name="term 10 symbols">
  - <port-definition>
      <element-reference element="term" type="state-transition-element"
        start="start-of-data" />
      <element-reference element="eob" type="state-transition-element"
        activation="out" />
    </port-definition>
  - <state-transition-element id="term" symbol-set="mail-order" case-
      insensitive="true" start="start-of-data" latch="true">
      <activate-on-match element="eob" />
    </state-transition-element>
  - <state-transition-element id="eob" symbol-set="0xFF">
      <activate-on-match macro="q006" element="q0-t4" />
    </state-transition-element>
  </macro>
- <macro id="tx6-2" name="term 6 symbols">
  - <port-definition>
      <element-reference element="term" type="state-transition-element"
        start="start-of-data" />
      <element-reference element="eob" type="state-transition-element"
        activation="out" />
    </port-definition>
  - <state-transition-element id="term" symbol-set="escort" start="start-of-
      data" case-insensitive="true" latch="true">
      <activate-on-match element="eob" />
    </state-transition-element>
  - <state-transition-element id="eob" symbol-set="0xFF">
      <activate-on-match macro="q006" element="q0-t4" />
    </state-transition-element>
  </macro>
- <macro id="q006" name="query type 6">
  - <port-definition>
      <element-reference element="cycle1-1" type="state-transition-
        element" activation="in" />
```

```
      <element-reference element="q0-t1" type="sum-of-products"
        activation="in" />
      <element-reference element="q0-t2" type="sum-of-products"
        activation="in" />
      <element-reference element="q0-t4" type="nor" activation="in" />
      <element-reference element="pos1" type="product-of-sums"
        activation="out" />
    </port-definition>
  - <sum-of-products id="sop1">
      <product-term id="q0-t1" />
      <product-term id="q0-t2" />
      <activate-on-high element="cycle1-2" />
    </sum-of-products>
  - <nor id="q0-t4">
      <activate-on-high element="cycle1-3" />
    </nor>
  - <state-transition-element id="cycle1-1" symbol-set="*">
      <activate-on-match element="q1-t1" />
    </state-transition-element>
  - <state-transition-element id="cycle1-2" symbol-set="*">
      <activate-on-match element="q1-t2" />
    </state-transition-element>
  - <state-transition-element id="cycle1-3" symbol-set="*">
      <activate-on-match element="q1-t3" />
    </state-transition-element>
  - <product-of-sums id="pos1">
      <sum-term id="q1-t1" />
      <sum-term id="q1-t2" />
      <activate-on-high macro="sn16" element="subn1" />
      <activate-on-high macro="sn16" element="subn2" />
      <activate-on-high macro="sn16" element="subn3" />
      <activate-on-high macro="sn16" element="subn4" />
      <activate-on-high macro="sn16" element="subn5" />
      <activate-on-high macro="sn16" element="subn6" />
      <activate-on-high macro="sn16" element="subn7" />
      <activate-on-high macro="sn16" element="subn8" />
      <activate-on-high macro="sn16" element="subn9" />
      <activate-on-high macro="sn16" element="subn10" />
      <activate-on-high macro="sn16" element="subn11" />
      <activate-on-high macro="sn16" element="subn12" />
      <activate-on-high macro="sn16" element="subn13" />
      <activate-on-high macro="sn16" element="subn14" />
      <activate-on-high macro="sn16" element="subn15" />
      <activate-on-high macro="sn16" element="subn16" />
    </product-of-sums>
  </macro>
- <macro name="sn16" id="subscriber network fanout 16">
  - <port-definition>
      <element-reference element="subn1" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn2" type="state-transition-element"
        activation="inout" />
```

```
        <element-reference element="subn3" type="state-transition-element"
          activation="inout" />
        <element-reference element="subn4" type="state-transition-element"
          activation="inout" />
        <element-reference element="subn5" type="state-transition-element"
          activation="inout" />
        <element-reference element="subn6" type="state-transition-element"
          activation="inout" />
        <element-reference element="subn7" type="state-transition-element"
          activation="inout" />
        <element-reference element="subn8" type="state-transition-element"
          activation="inout" />
        <element-reference element="subn9" type="state-transition-element"
          activation="inout" />
        <element-reference element="subn10" type="state-transition-element"
          activation="inout" />
        <element-reference element="subn11" type="state-transition-element"
          activation="inout" />
        <element-reference element="subn12" type="state-transition-element"
          activation="inout" />
        <element-reference element="subn13" type="state-transition-element"
          activation="inout" />
        <element-reference element="subn14" type="state-transition-element"
          activation="inout" />
        <element-reference element="subn15" type="state-transition-element"
          activation="inout" />
        <element-reference element="subn16" type="state-transition-element"
          activation="inout" />
      </port-definition>
    - <state-transition-element id="subn1" symbol-set="*">
        <activate-on-match element="sc1" />
        <activate-on-match element="sc2" />
      </state-transition-element>
      <state-transition-element id="subn2" symbol-set="*" />
      <state-transition-element id="subn3" symbol-set="*" />
      <state-transition-element id="subn4" symbol-set="*" />
      <state-transition-element id="subn5" symbol-set="*" />
      <state-transition-element id="subn6" symbol-set="*" />
      <state-transition-element id="subn7" symbol-set="*" />
      <state-transition-element id="subn8" symbol-set="*" />
      <state-transition-element id="subn9" symbol-set="*" />
      <state-transition-element id="subn10" symbol-set="*" />
      <state-transition-element id="subn11" symbol-set="*" />
      <state-transition-element id="subn12" symbol-set="*" />
      <state-transition-element id="subn13" symbol-set="*" />
      <state-transition-element id="subn14" symbol-set="*" />
      <state-transition-element id="subn15" symbol-set="*" />
      <state-transition-element id="subn16" symbol-set="*" />
    </macro>
  </macro>
- <macro id="qc02" name="Micron Employees">
  - <port-definition>
      <element-reference macro="sn16" element="subn1" type="state-transition-
```

```
element" activation="out" />
    <element-reference macro="sn16" element="subn2" type="state-transition-
      element" activation="out" />
    <element-reference macro="sn16" element="subn3" type="state-transition-
      element" activation="out" />
    <element-reference macro="sn16" element="subn4" type="state-transition-
      element" activation="out" />
    <element-reference macro="sn16" element="subn5" type="state-transition-
      element" activation="out" />
    <element-reference macro="sn16" element="subn6" type="state-transition-
      element" activation="out" />
    <element-reference macro="sn16" element="subn7" type="state-transition-
      element" activation="out" />
    <element-reference macro="sn16" element="subn8" type="state-transition-
      element" activation="out" />
    <element-reference macro="sn16" element="subn9" type="state-transition-
      element" activation="out" />
    <element-reference macro="sn16" element="subn10" type="state-
      transition-element" activation="out" />
    <element-reference macro="sn16" element="subn11" type="state-
      transition-element" activation="out" />
    <element-reference macro="sn16" element="subn12" type="state-
      transition-element" activation="out" />
    <element-reference macro="sn16" element="subn13" type="state-
      transition-element" activation="out" />
    <element-reference macro="sn16" element="subn14" type="state-
      transition-element" activation="out" />
    <element-reference macro="sn16" element="subn15" type="state-
      transition-element" activation="out" />
    <element-reference macro="sn16" element="subn16" type="state-
      transition-element" activation="out" />
  </port-definition>
- <macro id="tx17-1" name="term 17 symbols">
  - <port-definition>
      <element-reference element="term" type="state-transition-element"
        start="start-of-data" />
      <element-reference element="eob" type="state-transition-element"
        activation="out" />
    </port-definition>
  - <state-transition-element id="term" symbol-set="long-term effects" case-
      insensitive="true" start="start-of-data" latch="true">
      <activate-on-match element="eob" />
    </state-transition-element>
  - <state-transition-element id="eob" symbol-set="0xFF">
      <activate-on-match macro="q002" element="q0-t1" />
    </state-transition-element>
  </macro>
- <macro id="tx16-1" name="term 16 symbols">
  - <port-definition>
      <element-reference element="term" type="state-transition-element"
        start="start-of-data" />
      <element-reference element="eob" type="state-transition-element"
        activation="out" />
```

```
    </port-definition>
  - <state-transition-element id="term" symbol-set="lack of sunlight" case-
      insensitive="true" start="start-of-data" latch="true">
      <activate-on-match element="eob" />
    </state-transition-element>
  - <state-transition-element id="eob" symbol-set="0xFF">
      <activate-on-match macro="q002" element="q0-t1" />
    </state-transition-element>
  </macro>
- <macro id="tx6-1" name="term 6 symbols">
  - <port-definition>
      <element-reference element="term" type="state-transition-element"
        start="start-of-data" />
      <element-reference element="eob" type="state-transition-element"
        activation="out" />
    </port-definition>
  - <state-transition-element id="term" symbol-set="Micron" start="start-of-
      data" latch="true">
      <activate-on-match element="eob" />
    </state-transition-element>
  - <state-transition-element id="eob" symbol-set="0xFF">
      <activate-on-match macro="q002" element="q0-t2" />
    </state-transition-element>
  </macro>
- <macro id="tx11-1" name="term 11 symbols">
  - <port-definition>
      <element-reference element="term" type="state-transition-element"
        start="start-of-data" />
      <element-reference element="eob" type="state-transition-element"
        activation="out" />
    </port-definition>
  - <state-transition-element id="term" symbol-set="stock price" case-
      insensitive="true" start="start-of-data" latch="true">
      <activate-on-match element="eob" />
    </state-transition-element>
  - <state-transition-element id="eob" symbol-set="0xFF">
      <activate-on-match macro="q002" element="q0-t2" />
    </state-transition-element>
  </macro>
- <macro id="tx7-1" name="term 7 symbols">
  - <port-definition>
      <element-reference element="term" type="state-transition-element"
        start="start-of-data" />
      <element-reference element="eob" type="state-transition-element"
        activation="out" />
    </port-definition>
  - <state-transition-element id="term" symbol-set="Dilbert" start="start-of-
      data" latch="true">
      <activate-on-match element="eob" />
    </state-transition-element>
  - <state-transition-element id="eob" symbol-set="0xFF">
      <activate-on-match macro="q002" element="cycle" />
    </state-transition-element>
```

```
  </macro>
- <macro id="q002" name="query type 2">
  - <port-definition>
      <element-reference element="q0-t1" type="sum-of-products"
        activation="in" />
      <element-reference element="q0-t2" type="sum-of-products"
        activation="in" />
      <element-reference element="cycle" type="state-transition-element"
        activation="inout" />
    </port-definition>
  - <sum-of-products id="sop1">
      <product-term id="q0-t1" />
      <product-term id="q0-t2" />
      <activate-on-high element="cycle" />
    </sum-of-products>
  - <state-transition-element id="cycle" symbol-set="*">
      <activate-on-match macro="sn16" element="subn1" />
      <activate-on-match macro="sn16" element="subn2" />
      <activate-on-match macro="sn16" element="subn3" />
      <activate-on-match macro="sn16" element="subn4" />
      <activate-on-match macro="sn16" element="subn5" />
      <activate-on-match macro="sn16" element="subn6" />
      <activate-on-match macro="sn16" element="subn7" />
      <activate-on-match macro="sn16" element="subn8" />
      <activate-on-match macro="sn16" element="subn9" />
      <activate-on-match macro="sn16" element="subn10" />
      <activate-on-match macro="sn16" element="subn11" />
      <activate-on-match macro="sn16" element="subn12" />
      <activate-on-match macro="sn16" element="subn13" />
      <activate-on-match macro="sn16" element="subn14" />
      <activate-on-match macro="sn16" element="subn15" />
      <activate-on-match macro="sn16" element="subn16" />
    </state-transition-element>
  </macro>
- <macro name="sn16" id="subscriber network fanout 16">
  - <port-definition>
      <element-reference element="subn1" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn2" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn3" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn4" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn5" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn6" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn7" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn8" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn9" type="state-transition-element"
```

```
      activation="inout" />
        <element-reference element="subn10" type="state-transition-element"
          activation="inout" />
        <element-reference element="subn11" type="state-transition-element"
          activation="inout" />
        <element-reference element="subn12" type="state-transition-element"
          activation="inout" />
        <element-reference element="subn13" type="state-transition-element"
          activation="inout" />
        <element-reference element="subn14" type="state-transition-element"
          activation="inout" />
        <element-reference element="subn15" type="state-transition-element"
          activation="inout" />
        <element-reference element="subn16" type="state-transition-element"
          activation="inout" />
      </port-definition>
    - <state-transition-element id="subn1" symbol-set="*">
        <activate-on-match element="sc1" />
        <activate-on-match element="sc2" />
      </state-transition-element>
      <state-transition-element id="subn2" symbol-set="*" />
      <state-transition-element id="subn3" symbol-set="*" />
      <state-transition-element id="subn4" symbol-set="*" />
      <state-transition-element id="subn5" symbol-set="*" />
      <state-transition-element id="subn6" symbol-set="*" />
      <state-transition-element id="subn7" symbol-set="*" />
      <state-transition-element id="subn8" symbol-set="*" />
      <state-transition-element id="subn9" symbol-set="*" />
      <state-transition-element id="subn10" symbol-set="*" />
      <state-transition-element id="subn11" symbol-set="*" />
      <state-transition-element id="subn12" symbol-set="*" />
      <state-transition-element id="subn13" symbol-set="*" />
      <state-transition-element id="subn14" symbol-set="*" />
      <state-transition-element id="subn15" symbol-set="*" />
      <state-transition-element id="subn16" symbol-set="*" />
    </macro>
  </macro>
- <macro id="qc03" name="Garage Style">
  - <port-definition>
      <element-reference macro="sn16" element="subn1" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn2" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn3" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn4" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn5" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn6" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn7" type="state-transition-
        element" activation="out" />
```

```
    <element-reference macro="sn16" element="subn8" type="state-transition-
      element" activation="out" />
    <element-reference macro="sn16" element="subn9" type="state-transition-
      element" activation="out" />
    <element-reference macro="sn16" element="subn10" type="state-
      transition-element" activation="out" />
    <element-reference macro="sn16" element="subn11" type="state-
      transition-element" activation="out" />
    <element-reference macro="sn16" element="subn12" type="state-
      transition-element" activation="out" />
    <element-reference macro="sn16" element="subn13" type="state-
      transition-element" activation="out" />
    <element-reference macro="sn16" element="subn14" type="state-
      transition-element" activation="out" />
    <element-reference macro="sn16" element="subn15" type="state-
      transition-element" activation="out" />
    <element-reference macro="sn16" element="subn16" type="state-
      transition-element" activation="out" />
  </port-definition>
- <macro id="tx6-1" name="term 6 symbols">
  - <port-definition>
      <element-reference element="term" type="state-transition-element"
        start="start-of-data" />
      <element-reference element="eob" type="state-transition-element"
        activation="out" />
    </port-definition>
  - <state-transition-element id="term" symbol-set="garage" case-
      insensitive="true" start="start-of-data" latch="true">
      <activate-on-match element="eob" />
    </state-transition-element>
  - <state-transition-element id="eob" symbol-set="0xFF">
      <activate-on-match macro="q001" element="q0-t1" />
    </state-transition-element>
  </macro>
- <macro id="tx5-1" name="term 5 symbols">
  - <port-definition>
      <element-reference element="term" type="state-transition-element"
        start="start-of-data" />
      <element-reference element="eob" type="state-transition-element"
        activation="out" />
    </port-definition>
  - <state-transition-element id="term" symbol-set="girls" case-
      insensitive="true" start="start-of-data" latch="true">
      <activate-on-match element="eob" />
    </state-transition-element>
  - <state-transition-element id="eob" symbol-set="0xFF">
      <activate-on-match macro="q001" element="q0-t2" />
    </state-transition-element>
  </macro>
- <macro id="tx7-1" name="term 7 symbols">
  - <port-definition>
      <element-reference element="term" type="state-transition-element"
        start="start-of-data" />
```

```
      <element-reference element="eob" type="state-transition-element"
        activation="out" />
    </port-definition>
  - <state-transition-element id="term" symbol-set="divorce" case-
      insensitive="true" start="start-of-data" latch="true">
      <activate-on-match element="eob" />
    </state-transition-element>
  - <state-transition-element id="eob" symbol-set="0xFF">
      <activate-on-match macro="q001" element="q0-t2" />
    </state-transition-element>
  </macro>
- <macro id="tx5-2" name="term 5 symbols">
  - <port-definition>
      <element-reference element="term" type="state-transition-element"
        start="start-of-data" />
      <element-reference element="eob" type="state-transition-element"
        activation="out" />
    </port-definition>
  - <state-transition-element id="term" symbol-set="style" case-
      insensitive="false" start="start-of-data" latch="true">
      <activate-on-match element="eob" />
    </state-transition-element>
  - <state-transition-element id="eob" symbol-set="0xFF">
      <activate-on-match macro="q001" element="q0-t2" />
    </state-transition-element>
  </macro>
- <macro id="tx7-2" name="term 7 symbols">
  - <port-definition>
      <element-reference element="term" type="state-transition-element"
        start="start-of-data" />
      <element-reference element="eob" type="state-transition-element"
        activation="out" />
    </port-definition>
  - <state-transition-element id="term" symbol-set="remodel" case-
      insensitive="true" start="start-of-data" latch="true">
      <activate-on-match element="eob" />
    </state-transition-element>
  - <state-transition-element id="eob" symbol-set="0xFF">
      <activate-on-match macro="q001" element="q0-t2" />
    </state-transition-element>
  </macro>
- <macro id="tx9-1" name="term 9 symbols">
  - <port-definition>
      <element-reference element="term" type="state-transition-element"
        start="start-of-data" />
      <element-reference element="eob" type="state-transition-element"
        activation="out" />
    </port-definition>
  - <state-transition-element id="term" symbol-set="obsession" case-
      insensitive="true" start="start-of-data" latch="true">
      <activate-on-match element="eob" />
    </state-transition-element>
  - <state-transition-element id="eob" symbol-set="0xFF">
```

```
      <activate-on-match macro="q001" element="q0-t2" />
    </state-transition-element>
  </macro>
- <macro id="tx8-1" name="term 8 symbols">
  - <port-definition>
      <element-reference element="term" type="state-transition-element"
        start="start-of-data" />
      <element-reference element="eob" type="state-transition-element"
        activation="out" />
    </port-definition>
  - <state-transition-element id="term" symbol-set="makeover" start="start-of-
      data" case-insensitive="true" latch="true">
      <activate-on-match element="eob" />
    </state-transition-element>
  - <state-transition-element id="eob" symbol-set="0xFF">
      <activate-on-match macro="q001" element="q0-t2" />
    </state-transition-element>
  </macro>
- <macro id="q001" name="query type 1">
  - <port-definition>
      <element-reference element="q0-t1" type="product-of-sums"
        activation="in" />
      <element-reference element="q0-t2" type="product-of-sums"
        activation="in" />
      <element-reference element="cycle" type="state-transition-element"
        activation="out" />
    </port-definition>
  - <product-of-sums id="pos1">
      <sum-term id="q0-t1" />
      <sum-term id="q0-t2" />
      <activate-on-high element="cycle" />
    </product-of-sums>
  - <state-transition-element id="cycle" symbol-set="*">
      <activate-on-match macro="sn16" element="subn1" />
      <activate-on-match macro="sn16" element="subn2" />
      <activate-on-match macro="sn16" element="subn3" />
      <activate-on-match macro="sn16" element="subn4" />
      <activate-on-match macro="sn16" element="subn5" />
      <activate-on-match macro="sn16" element="subn6" />
      <activate-on-match macro="sn16" element="subn7" />
      <activate-on-match macro="sn16" element="subn8" />
      <activate-on-match macro="sn16" element="subn9" />
      <activate-on-match macro="sn16" element="subn10" />
      <activate-on-match macro="sn16" element="subn11" />
      <activate-on-match macro="sn16" element="subn12" />
      <activate-on-match macro="sn16" element="subn13" />
      <activate-on-match macro="sn16" element="subn14" />
      <activate-on-match macro="sn16" element="subn15" />
      <activate-on-match macro="sn16" element="subn16" />
    </state-transition-element>
  </macro>
- <macro name="sn16" id="subscriber network fanout 16">
```

```
- <port-definition>
    <element-reference element="subn1" type="state-transition-element"
      activation="inout" />
    <element-reference element="subn2" type="state-transition-element"
      activation="inout" />
    <element-reference element="subn3" type="state-transition-element"
      activation="inout" />
    <element-reference element="subn4" type="state-transition-element"
      activation="inout" />
    <element-reference element="subn5" type="state-transition-element"
      activation="inout" />
    <element-reference element="subn6" type="state-transition-element"
      activation="inout" />
    <element-reference element="subn7" type="state-transition-element"
      activation="inout" />
    <element-reference element="subn8" type="state-transition-element"
      activation="inout" />
    <element-reference element="subn9" type="state-transition-element"
      activation="inout" />
    <element-reference element="subn10" type="state-transition-element"
      activation="inout" />
    <element-reference element="subn11" type="state-transition-element"
      activation="inout" />
    <element-reference element="subn12" type="state-transition-element"
      activation="inout" />
    <element-reference element="subn13" type="state-transition-element"
      activation="inout" />
    <element-reference element="subn14" type="state-transition-element"
      activation="inout" />
    <element-reference element="subn15" type="state-transition-element"
      activation="inout" />
    <element-reference element="subn16" type="state-transition-element"
      activation="inout" />
  </port-definition>
- <state-transition-element id="subn1" symbol-set="*">
    <activate-on-match element="sc1" />
  </state-transition-element>
  <state-transition-element id="subn2" symbol-set="*" />
  <state-transition-element id="subn3" symbol-set="*" />
  <state-transition-element id="subn4" symbol-set="*" />
  <state-transition-element id="subn5" symbol-set="*" />
  <state-transition-element id="subn6" symbol-set="*" />
  <state-transition-element id="subn7" symbol-set="*" />
  <state-transition-element id="subn8" symbol-set="*" />
  <state-transition-element id="subn9" symbol-set="*" />
  <state-transition-element id="subn10" symbol-set="*" />
  <state-transition-element id="subn11" symbol-set="*" />
  <state-transition-element id="subn12" symbol-set="*" />
  <state-transition-element id="subn13" symbol-set="*" />
  <state-transition-element id="subn14" symbol-set="*" />
  <state-transition-element id="subn15" symbol-set="*" />
  <state-transition-element id="subn16" symbol-set="*" />
</macro>
```

```
  </macro>
- <macro id="qc04" name="Motocross for Old Guys">
  - <port-definition>
      <element-reference macro="sn16" element="subn1" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn2" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn3" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn4" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn5" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn6" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn7" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn8" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn9" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn10" type="state-
        transition-element" activation="out" />
      <element-reference macro="sn16" element="subn11" type="state-
        transition-element" activation="out" />
      <element-reference macro="sn16" element="subn12" type="state-
        transition-element" activation="out" />
      <element-reference macro="sn16" element="subn13" type="state-
        transition-element" activation="out" />
      <element-reference macro="sn16" element="subn14" type="state-
        transition-element" activation="out" />
      <element-reference macro="sn16" element="subn15" type="state-
        transition-element" activation="out" />
      <element-reference macro="sn16" element="subn16" type="state-
        transition-element" activation="out" />
    </port-definition>
  - <macro id="tx4-1" name="term 4 symbols">
    - <port-definition>
        <element-reference element="term" type="state-transition-element"
          start="start-of-data" />
        <element-reference element="eob" type="state-transition-element"
          activation="out" />
      </port-definition>
    - <state-transition-element id="term" symbol-set="dirt" case-insensitive="true"
        start="start-of-data" latch="true">
        <activate-on-match element="eob" />
      </state-transition-element>
    - <state-transition-element id="eob" symbol-set="0xFF">
        <activate-on-match macro="q005" element="q00-t1" />
      </state-transition-element>
    </macro>
  - <macro id="tx9-1" name="term 9 symbols">
    - <port-definition>
```

```
      <element-reference element="term" type="state-transition-element"
        start="start-of-data" />
      <element-reference element="eob" type="state-transition-element"
        activation="out" />
    </port-definition>
  - <state-transition-element id="term" symbol-set="motocross" case-
      insensitive="true" start="start-of-data" latch="true">
      <activate-on-match element="eob" />
    </state-transition-element>
  - <state-transition-element id="eob" symbol-set="0xFF">
      <activate-on-match macro="q005" element="q00-t1" />
    </state-transition-element>
  </macro>
- <macro id="tx6-1" name="term 6 symbols">
  - <port-definition>
      <element-reference element="term" type="state-transition-element"
        start="start-of-data" />
      <element-reference element="eob" type="state-transition-element"
        activation="out" />
    </port-definition>
  - <state-transition-element id="term" symbol-set="racing" case-
      insensitive="true" start="start-of-data" latch="true">
      <activate-on-match element="eob" />
    </state-transition-element>
  - <state-transition-element id="eob" symbol-set="0xFF">
      <activate-on-match macro="q005" element="q00-t2" />
    </state-transition-element>
  </macro>
- <macro id="tx5-1" name="term 5 symbols">
  - <port-definition>
      <element-reference element="term" type="state-transition-element"
        start="start-of-data" />
      <element-reference element="eob" type="state-transition-element"
        activation="out" />
    </port-definition>
  - <state-transition-element id="term" symbol-set="girls" case-
      insensitive="true" start="start-of-data" latch="true">
      <activate-on-match element="eob" />
    </state-transition-element>
  - <state-transition-element id="eob" symbol-set="0xFF">
      <activate-on-match macro="q005" element="q01-t1" />
    </state-transition-element>
  </macro>
- <macro id="tx8-1" name="term 8 symbols">
  - <port-definition>
      <element-reference element="term" type="state-transition-element"
        start="start-of-data" />
      <element-reference element="eob" type="state-transition-element"
        activation="out" />
    </port-definition>
  - <state-transition-element id="term" symbol-set="leathers" case-
      insensitive="true" start="start-of-data" latch="true">
      <activate-on-match element="eob" />
```

```
    </state-transition-element>
  - <state-transition-element id="eob" symbol-set="0xFF">
      <activate-on-match macro="q005" element="q01-t1" />
    </state-transition-element>
  </macro>
- <macro id="tx4-2" name="term 4 symbols">
  - <port-definition>
      <element-reference element="term" type="state-transition-element"
        start="start-of-data" />
      <element-reference element="eob" type="state-transition-element"
        activation="out" />
    </port-definition>
  - <state-transition-element id="term" symbol-set="bike" case-
      insensitive="true" start="start-of-data" latch="true">
      <activate-on-match element="eob" />
    </state-transition-element>
  - <state-transition-element id="eob" symbol-set="0xFF">
      <activate-on-match macro="q005" element="q01-t1" />
    </state-transition-element>
  </macro>
- <macro id="tx3-1" name="term 3 symbols">
  - <port-definition>
      <element-reference element="term" type="state-transition-element"
        start="start-of-data" />
      <element-reference element="eob" type="state-transition-element"
        activation="out" />
    </port-definition>
  - <state-transition-element id="term" symbol-set="hot" start="start-of-data"
      case-insensitive="true" latch="true">
      <activate-on-match element="eob" />
    </state-transition-element>
  - <state-transition-element id="eob" symbol-set="0xFF">
      <activate-on-match macro="q005" element="q01-t2" />
    </state-transition-element>
  </macro>
- <macro id="q006" name="query type 6">
  - <port-definition>
      <element-reference element="q00-t1" type="sum-of-products"
        activation="in" />
      <element-reference element="q00-t2" type="sum-of-products"
        activation="in" />
      <element-reference element="q01-t1" type="sum-of-products"
        activation="in" />
      <element-reference element="q01-t2" type="sum-of-products"
        activation="in" />
      <element-reference element="and1" type="state-transition-element"
        activation="out" />
    </port-definition>
  - <sum-of-products id="sop1">
      <product-term id="q00-t1" />
      <product-term id="q00-t2" />
      <activate-on-high element="cycle1-1" />
    </sum-of-products>
```

```
- <sum-of-products id="sop2">
    <product-term id="q01-t1" />
    <product-term id="q01-t2" />
    <activate-on-high element="cycle1-2" />
  </sum-of-products>
- <state-transition-element id="cycle1-1" symbol-set="*">
    <activate-on-match element="and1" />
  </state-transition-element>
- <state-transition-element id="cycle1-2" symbol-set="*">
    <activate-on-match element="and1" />
  </state-transition-element>
- <and id="and1">
    <activate-on-high macro="sn16" element="subn1" />
    <activate-on-high macro="sn16" element="subn2" />
    <activate-on-high macro="sn16" element="subn3" />
    <activate-on-high macro="sn16" element="subn4" />
    <activate-on-high macro="sn16" element="subn5" />
    <activate-on-high macro="sn16" element="subn6" />
    <activate-on-high macro="sn16" element="subn7" />
    <activate-on-high macro="sn16" element="subn8" />
    <activate-on-high macro="sn16" element="subn9" />
    <activate-on-high macro="sn16" element="subn10" />
    <activate-on-high macro="sn16" element="subn11" />
    <activate-on-high macro="sn16" element="subn12" />
    <activate-on-high macro="sn16" element="subn13" />
    <activate-on-high macro="sn16" element="subn14" />
    <activate-on-high macro="sn16" element="subn15" />
    <activate-on-high macro="sn16" element="subn16" />
  </and>
</macro>
- <macro name="sn16" id="subscriber network fanout 16">
  - <port-definition>
      <element-reference element="subn1" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn2" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn3" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn4" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn5" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn6" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn7" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn8" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn9" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn10" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn11" type="state-transition-element"
```

```
      activation="inout" />
        <element-reference element="subn12" type="state-transition-element"
          activation="inout" />
        <element-reference element="subn13" type="state-transition-element"
          activation="inout" />
        <element-reference element="subn14" type="state-transition-element"
          activation="inout" />
        <element-reference element="subn15" type="state-transition-element"
          activation="inout" />
        <element-reference element="subn16" type="state-transition-element"
          activation="inout" />
      </port-definition>
    - <state-transition-element id="subn1" symbol-set="*">
        <activate-on-match element="sc1" />
      </state-transition-element>
      <state-transition-element id="subn2" symbol-set="*" />
      <state-transition-element id="subn3" symbol-set="*" />
      <state-transition-element id="subn4" symbol-set="*" />
      <state-transition-element id="subn5" symbol-set="*" />
      <state-transition-element id="subn6" symbol-set="*" />
      <state-transition-element id="subn7" symbol-set="*" />
      <state-transition-element id="subn8" symbol-set="*" />
      <state-transition-element id="subn9" symbol-set="*" />
      <state-transition-element id="subn10" symbol-set="*" />
      <state-transition-element id="subn11" symbol-set="*" />
      <state-transition-element id="subn12" symbol-set="*" />
      <state-transition-element id="subn13" symbol-set="*" />
      <state-transition-element id="subn14" symbol-set="*" />
      <state-transition-element id="subn15" symbol-set="*" />
      <state-transition-element id="subn16" symbol-set="*" />
    </macro>
  </macro>
- <macro id="qc05" name="Cheesecake Factory Lover">
  - <port-definition>
      <element-reference macro="sn16" element="subn1" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn2" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn3" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn4" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn5" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn6" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn7" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn8" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn9" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn10" type="state-
```

```
transition-element" activation="out" />
    <element-reference macro="sn16" element="subn11" type="state-
      transition-element" activation="out" />
    <element-reference macro="sn16" element="subn12" type="state-
      transition-element" activation="out" />
    <element-reference macro="sn16" element="subn13" type="state-
      transition-element" activation="out" />
    <element-reference macro="sn16" element="subn14" type="state-
      transition-element" activation="out" />
    <element-reference macro="sn16" element="subn15" type="state-
      transition-element" activation="out" />
    <element-reference macro="sn16" element="subn16" type="state-
      transition-element" activation="out" />
  </port-definition>
- <macro id="tx18-1" name="term 18 symbols">
  - <port-definition>
      <element-reference element="term" type="state-transition-element"
        start="start-of-data" />
      <element-reference element="eob" type="state-transition-element"
        activation="out" />
    </port-definition>
  - <state-transition-element id="term" symbol-set="Cheesecake Factory"
      start="start-of-data" latch="true">
      <activate-on-match element="eob" />
    </state-transition-element>
  - <state-transition-element id="eob" symbol-set="0xFF">
      <activate-on-match macro="q004" element="and1" />
    </state-transition-element>
  </macro>
- <macro id="tx6-1" name="term 6 symbols">
  - <port-definition>
      <element-reference element="term" type="state-transition-element"
        start="start-of-data" />
      <element-reference element="eob" type="state-transition-element"
        activation="out" />
    </port-definition>
  - <state-transition-element id="term" symbol-set="dating" case-
      insensitive="true" start="start-of-data" latch="true">
      <activate-on-match element="eob" />
    </state-transition-element>
  - <state-transition-element id="eob" symbol-set="0xFF">
      <activate-on-match macro="q004" element="q0-t1" />
    </state-transition-element>
  </macro>
- <macro id="tx15-1" name="term 15 symbols">
  - <port-definition>
      <element-reference element="term" type="state-transition-element"
        start="start-of-data" />
      <element-reference element="eob" type="state-transition-element"
        activation="out" />
    </port-definition>
  - <state-transition-element id="term" symbol-set="plus-size girls" case-
      insensitive="true" start="start-of-data" latch="true">
```

```
      <activate-on-match element="eob" />
    </state-transition-element>
  - <state-transition-element id="eob" symbol-set="0xFF">
      <activate-on-match macro="q004" element="q0-t1" />
    </state-transition-element>
  </macro>
- <macro id="tx5-1" name="term 5 symbols">
  - <port-definition>
      <element-reference element="term" type="state-transition-element"
        start="start-of-data" />
      <element-reference element="eob" type="state-transition-element"
        activation="out" />
    </port-definition>
  - <state-transition-element id="term" symbol-set="Boise" start="start-of-
      data" latch="true">
      <activate-on-match element="eob" />
    </state-transition-element>
  - <state-transition-element id="eob" symbol-set="0xFF">
      <activate-on-match macro="q004" element="q0-t2" />
    </state-transition-element>
  </macro>
- <macro id="tx7-1" name="term 7 symbols">
  - <port-definition>
      <element-reference element="term" type="state-transition-element"
        start="start-of-data" />
      <element-reference element="eob" type="state-transition-element"
        activation="out" />
    </port-definition>
  - <state-transition-element id="term" symbol-set="careers" case-
      insensitive="true" start="start-of-data" latch="true">
      <activate-on-match element="eob" />
    </state-transition-element>
  - <state-transition-element id="eob" symbol-set="0xFF">
      <activate-on-match macro="q004" element="q0-t2" />
    </state-transition-element>
  </macro>
- <macro id="tx6-2" name="term 6 symbols">
  - <port-definition>
      <element-reference element="term" type="state-transition-element"
        start="start-of-data" />
      <element-reference element="eob" type="state-transition-element"
        activation="out" />
    </port-definition>
  - <state-transition-element id="term" symbol-set="coupon" case-
      insensitive="true" start="start-of-data" latch="true">
      <activate-on-match element="eob" />
    </state-transition-element>
  - <state-transition-element id="eob" symbol-set="0xFF">
      <activate-on-match macro="q004" element="cycle" />
    </state-transition-element>
  </macro>
- <macro id="tx8-1" name="term 8 symbols">
  - <port-definition>
```

```
      <element-reference element="term" type="state-transition-element"
        start="start-of-data" />
      <element-reference element="eob" type="state-transition-element"
        activation="out" />
    </port-definition>
  - <state-transition-element id="term" symbol-set="specials" start="start-of-
      data" case-insensitive="true" latch="true">
      <activate-on-match element="eob" />
    </state-transition-element>
  - <state-transition-element id="eob" symbol-set="0xFF">
      <activate-on-match macro="q004" element="cycle" />
    </state-transition-element>
  </macro>
- <macro id="q004" name="query type 4">
  - <port-definition>
      <element-reference element="and1" type="and" activation="inout" />
      <element-reference element="q0-t1" type="sum-of-products"
        activation="in" />
      <element-reference element="q0-t2" type="sum-of-products"
        activation="in" />
      <element-reference element="cycle" type="state-transition-element"
        activation="in" />
    </port-definition>
  - <sum-of-products id="sop1">
      <product-term id="q0-t1" />
      <product-term id="q0-t2" />
      <activate-on-high element="cycle" />
    </sum-of-products>
  - <state-transition-element id="cycle" symbol-set="*">
      <activate-on-match element="and1" />
    </state-transition-element>
  - <and id="and1">
      <activate-on-high macro="sn16" element="subn1" />
      <activate-on-high macro="sn16" element="subn2" />
      <activate-on-high macro="sn16" element="subn3" />
      <activate-on-high macro="sn16" element="subn4" />
      <activate-on-high macro="sn16" element="subn5" />
      <activate-on-high macro="sn16" element="subn6" />
      <activate-on-high macro="sn16" element="subn7" />
      <activate-on-high macro="sn16" element="subn8" />
      <activate-on-high macro="sn16" element="subn9" />
      <activate-on-high macro="sn16" element="subn10" />
      <activate-on-high macro="sn16" element="subn11" />
      <activate-on-high macro="sn16" element="subn12" />
      <activate-on-high macro="sn16" element="subn13" />
      <activate-on-high macro="sn16" element="subn14" />
      <activate-on-high macro="sn16" element="subn15" />
      <activate-on-high macro="sn16" element="subn16" />
    </and>
  </macro>
- <macro name="sn16" id="subscriber network fanout 16">
  - <port-definition>
```

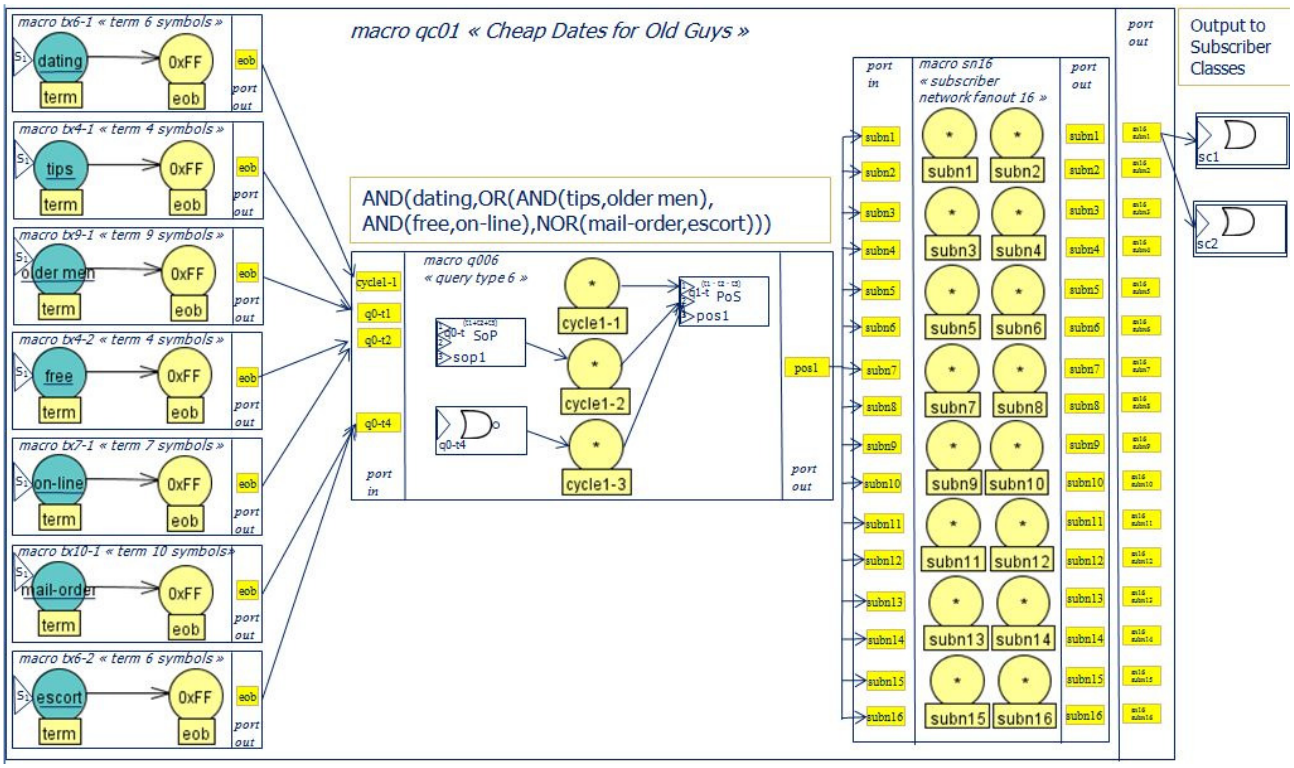
```
      <element-reference element="subn1" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn2" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn3" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn4" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn5" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn6" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn7" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn8" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn9" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn10" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn11" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn12" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn13" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn14" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn15" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn16" type="state-transition-element"
        activation="inout" />
    </port-definition>
  - <state-transition-element id="subn1" symbol-set="*">
      <activate-on-match element="sc1" />
    </state-transition-element>
    <state-transition-element id="subn2" symbol-set="*" />
    <state-transition-element id="subn3" symbol-set="*" />
    <state-transition-element id="subn4" symbol-set="*" />
    <state-transition-element id="subn5" symbol-set="*" />
    <state-transition-element id="subn6" symbol-set="*" />
    <state-transition-element id="subn7" symbol-set="*" />
    <state-transition-element id="subn8" symbol-set="*" />
    <state-transition-element id="subn9" symbol-set="*" />
    <state-transition-element id="subn10" symbol-set="*" />
    <state-transition-element id="subn11" symbol-set="*" />
    <state-transition-element id="subn12" symbol-set="*" />
    <state-transition-element id="subn13" symbol-set="*" />
    <state-transition-element id="subn14" symbol-set="*" />
    <state-transition-element id="subn15" symbol-set="*" />
    <state-transition-element id="subn16" symbol-set="*" />
  </macro>
</macro>
```

```
</automata-network>
```

PubSub Query Class Example 1 Cheap Dates

PubSub Query Class Example 1: Cheap Dates

ANML-G Macro



ANML Template

```
<?xml version="1.0" ?>
- <macro id="qc01" name="Cheap Dates for Old Guys"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="ANML_v6.xsd">
  - <port-definition>
      <element-reference macro="sn16" element="subn1" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn2" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn3" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn4" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn5" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn6" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn7" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn8" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn9" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn10" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn11" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn12" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn13" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn14" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn15" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn16" type="state-transition-
        element" activation="out" />
    </port-definition>
  - <macro id="tx6-1" name="term 6 symbols">
    - <port-definition>
        <element-reference element="term" type="state-transition-element"
          start="start-of-data" />
        <element-reference element="eob" type="state-transition-element"
          activation="out" />
      </port-definition>
    - <state-transition-element id="term" symbol-set="dating" case-insensitive="true"
        start="start-of-data" latch="true">
        <activate-on-match element="eob" />
      </state-transition-element>
    - <state-transition-element id="eob" symbol-set="0xFF">
        <activate-on-match macro="q006" element="cycle1-1" />
      </state-transition-element>
    </macro>
```

```
- <macro id="tx4-1" name="term 4 symbols">
  - <port-definition>
      <element-reference element="term" type="state-transition-element"
        start="start-of-data" />
      <element-reference element="eob" type="state-transition-element"
        activation="out" />
    </port-definition>
  - <state-transition-element id="term" symbol-set="tips" case-insensitive="true"
      start="start-of-data" latch="true">
      <activate-on-match element="eob" />
    </state-transition-element>
  - <state-transition-element id="eob" symbol-set="0xFF">
      <activate-on-match macro="q006" element="q0-t1" />
    </state-transition-element>
  </macro>
- <macro id="tx9-1" name="term 9 symbols">
  - <port-definition>
      <element-reference element="term" type="state-transition-element"
        start="start-of-data" />
      <element-reference element="eob" type="state-transition-element"
        activation="out" />
    </port-definition>
  - <state-transition-element id="term" symbol-set="older men" case-
      insensitive="true" start="start-of-data" latch="true">
      <activate-on-match element="eob" />
    </state-transition-element>
  - <state-transition-element id="eob" symbol-set="0xFF">
      <activate-on-match macro="q006" element="q0-t1" />
    </state-transition-element>
  </macro>
- <macro id="tx4-2" name="term 4 symbols">
  - <port-definition>
      <element-reference element="term" type="state-transition-element"
        start="start-of-data" />
      <element-reference element="eob" type="state-transition-element"
        activation="out" />
    </port-definition>
  - <state-transition-element id="term" symbol-set="free" case-insensitive="true"
      start="start-of-data" latch="true">
      <activate-on-match element="eob" />
    </state-transition-element>
  - <state-transition-element id="eob" symbol-set="0xFF">
      <activate-on-match macro="q006" element="q0-t2" />
    </state-transition-element>
  </macro>
- <macro id="tx7-1" name="term 7 symbols">
  - <port-definition>
      <element-reference element="term" type="state-transition-element"
        start="start-of-data" />
      <element-reference element="eob" type="state-transition-element"
        activation="out" />
    </port-definition>
  - <state-transition-element id="term" symbol-set="on-line" case-insensitive="true"
```

```
start="start-of-data" latch="true">
    <activate-on-match element="eob" />
  </state-transition-element>
- <state-transition-element id="eob" symbol-set="0xFF">
    <activate-on-match macro="q006" element="q0-t2" />
  </state-transition-element>
</macro>
- <macro id="tx10-1" name="term 10 symbols">
  - <port-definition>
      <element-reference element="term" type="state-transition-element"
        start="start-of-data" />
      <element-reference element="eob" type="state-transition-element"
        activation="out" />
    </port-definition>
  - <state-transition-element id="term" symbol-set="mail-order" case-
      insensitive="true" start="start-of-data" latch="true">
      <activate-on-match element="eob" />
    </state-transition-element>
  - <state-transition-element id="eob" symbol-set="0xFF">
      <activate-on-match macro="q006" element="q0-t4" />
    </state-transition-element>
  </macro>
- <macro id="tx6-2" name="term 6 symbols">
  - <port-definition>
      <element-reference element="term" type="state-transition-element"
        start="start-of-data" />
      <element-reference element="eob" type="state-transition-element"
        activation="out" />
    </port-definition>
  - <state-transition-element id="term" symbol-set="escort" start="start-of-data"
      case-insensitive="true" latch="true">
      <activate-on-match element="eob" />
    </state-transition-element>
  - <state-transition-element id="eob" symbol-set="0xFF">
      <activate-on-match macro="q006" element="q0-t4" />
    </state-transition-element>
  </macro>
- <macro id="q006" name="query type 6">
  - <port-definition>
      <element-reference element="cycle1-1" type="state-transition-element"
        activation="in" />
      <element-reference element="q0-t1" type="sum-of-products"
        activation="in" />
      <element-reference element="q0-t2" type="sum-of-products"
        activation="in" />
      <element-reference element="q0-t4" type="nor" activation="in" />
      <element-reference element="pos1" type="product-of-sums"
        activation="out" />
    </port-definition>
  - <sum-of-products id="sop1">
      <product-term id="q0-t1" />
      <product-term id="q0-t2" />
      <activate-on-high element="cycle1-2" />
```

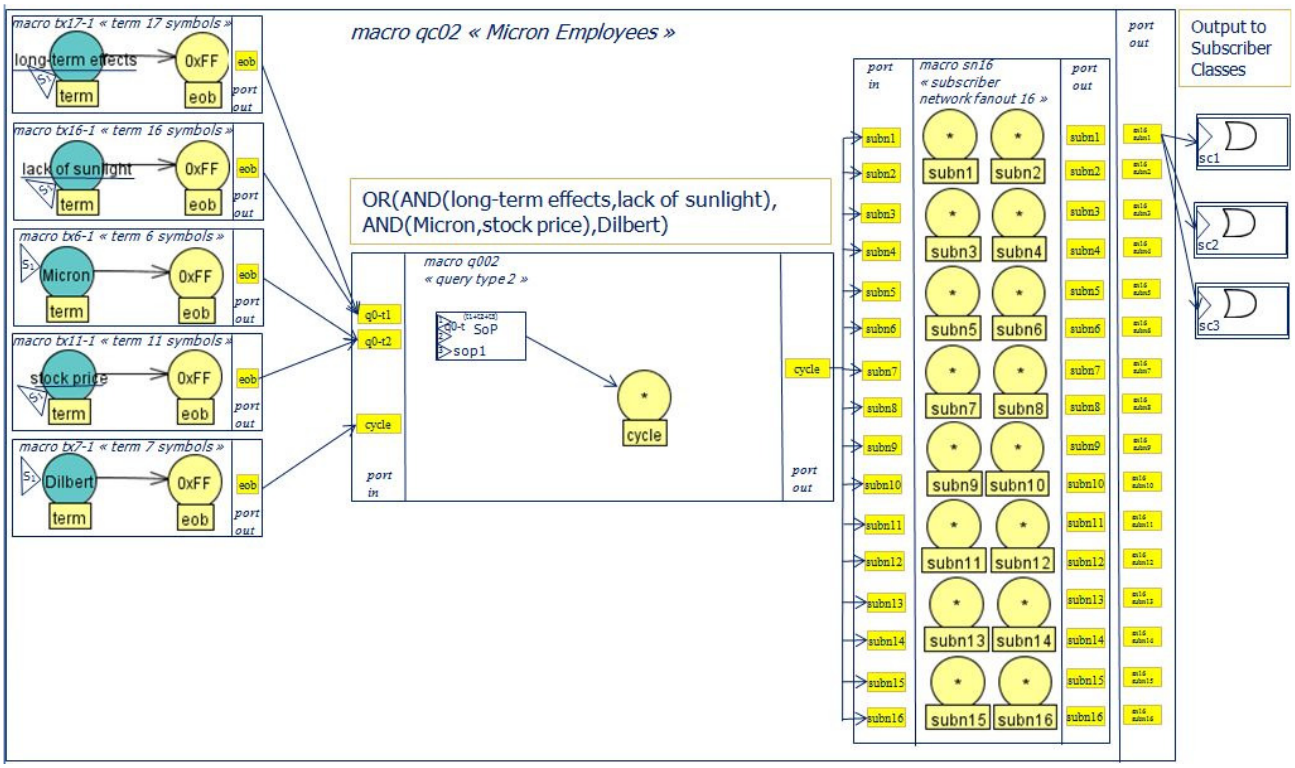
```
    </sum-of-products>
  - <nor id="q0-t4">
      <activate-on-high element="cycle1-3" />
    </nor>
  - <state-transition-element id="cycle1-1" symbol-set="*">
      <activate-on-match element="q1-t1" />
    </state-transition-element>
  - <state-transition-element id="cycle1-2" symbol-set="*">
      <activate-on-match element="q1-t2" />
    </state-transition-element>
  - <state-transition-element id="cycle1-3" symbol-set="*">
      <activate-on-match element="q1-t3" />
    </state-transition-element>
  - <product-of-sums id="pos1">
      <sum-term id="q1-t1" />
      <sum-term id="q1-t2" />
      <activate-on-high macro="sn16" element="subn1" />
      <activate-on-high macro="sn16" element="subn2" />
      <activate-on-high macro="sn16" element="subn3" />
      <activate-on-high macro="sn16" element="subn4" />
      <activate-on-high macro="sn16" element="subn5" />
      <activate-on-high macro="sn16" element="subn6" />
      <activate-on-high macro="sn16" element="subn7" />
      <activate-on-high macro="sn16" element="subn8" />
      <activate-on-high macro="sn16" element="subn9" />
      <activate-on-high macro="sn16" element="subn10" />
      <activate-on-high macro="sn16" element="subn11" />
      <activate-on-high macro="sn16" element="subn12" />
      <activate-on-high macro="sn16" element="subn13" />
      <activate-on-high macro="sn16" element="subn14" />
      <activate-on-high macro="sn16" element="subn15" />
      <activate-on-high macro="sn16" element="subn16" />
    </product-of-sums>
  </macro>
- <macro name="sn16" id="subscriber network fanout 16">
  - <port-definition>
      <element-reference element="subn1" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn2" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn3" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn4" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn5" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn6" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn7" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn8" type="state-transition-element"
        activation="inout" />
```

```
      <element-reference element="subn9" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn10" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn11" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn12" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn13" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn14" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn15" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn16" type="state-transition-element"
        activation="inout" />
    </port-definition>
  - <state-transition-element id="subn1" symbol-set="*">
      <activate-on-match element="sc1" />
      <activate-on-match element="sc2" />
    </state-transition-element>
    <state-transition-element id="subn2" symbol-set="*" />
    <state-transition-element id="subn3" symbol-set="*" />
    <state-transition-element id="subn4" symbol-set="*" />
    <state-transition-element id="subn5" symbol-set="*" />
    <state-transition-element id="subn6" symbol-set="*" />
    <state-transition-element id="subn7" symbol-set="*" />
    <state-transition-element id="subn8" symbol-set="*" />
    <state-transition-element id="subn9" symbol-set="*" />
    <state-transition-element id="subn10" symbol-set="*" />
    <state-transition-element id="subn11" symbol-set="*" />
    <state-transition-element id="subn12" symbol-set="*" />
    <state-transition-element id="subn13" symbol-set="*" />
    <state-transition-element id="subn14" symbol-set="*" />
    <state-transition-element id="subn15" symbol-set="*" />
    <state-transition-element id="subn16" symbol-set="*" />
  </macro>
</macro>
```

PubSub Query Class Example 2 Micron Employees

PubSub Query Class Example 2: Micron Employees

ANML-G Macro



ANML Template

```
<?xml version="1.0" ?>
- <macro id="qc02" name="Micron Employees"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="ANML_v6.xsd">
  - <port-definition>
      <element-reference macro="sn16" element="subn1" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn2" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn3" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn4" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn5" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn6" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn7" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn8" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn9" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn10" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn11" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn12" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn13" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn14" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn15" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn16" type="state-transition-
        element" activation="out" />
    </port-definition>
  - <macro id="tx17-1" name="term 17 symbols">
    - <port-definition>
        <element-reference element="term" type="state-transition-element"
          start="start-of-data" />
        <element-reference element="eob" type="state-transition-element"
          activation="out" />
      </port-definition>
    - <state-transition-element id="term" symbol-set="long-term effects" case-
        insensitive="true" start="start-of-data" latch="true">
        <activate-on-match element="eob" />
      </state-transition-element>
    - <state-transition-element id="eob" symbol-set="0xFF">
        <activate-on-match macro="q002" element="q0-t1" />
      </state-transition-element>
    </macro>
```

```
- <macro id="tx16-1" name="term 16 symbols">
  - <port-definition>
      <element-reference element="term" type="state-transition-element"
        start="start-of-data" />
      <element-reference element="eob" type="state-transition-element"
        activation="out" />
    </port-definition>
  - <state-transition-element id="term" symbol-set="lack of sunlight" case-
      insensitive="true" start="start-of-data" latch="true">
      <activate-on-match element="eob" />
    </state-transition-element>
  - <state-transition-element id="eob" symbol-set="0xFF">
      <activate-on-match macro="q002" element="q0-t1" />
    </state-transition-element>
  </macro>
- <macro id="tx6-1" name="term 6 symbols">
  - <port-definition>
      <element-reference element="term" type="state-transition-element"
        start="start-of-data" />
      <element-reference element="eob" type="state-transition-element"
        activation="out" />
    </port-definition>
  - <state-transition-element id="term" symbol-set="Micron" start="start-of-data"
      latch="true">
      <activate-on-match element="eob" />
    </state-transition-element>
  - <state-transition-element id="eob" symbol-set="0xFF">
      <activate-on-match macro="q002" element="q0-t2" />
    </state-transition-element>
  </macro>
- <macro id="tx11-1" name="term 11 symbols">
  - <port-definition>
      <element-reference element="term" type="state-transition-element"
        start="start-of-data" />
      <element-reference element="eob" type="state-transition-element"
        activation="out" />
    </port-definition>
  - <state-transition-element id="term" symbol-set="stock price" case-
      insensitive="true" start="start-of-data" latch="true">
      <activate-on-match element="eob" />
    </state-transition-element>
  - <state-transition-element id="eob" symbol-set="0xFF">
      <activate-on-match macro="q002" element="q0-t2" />
    </state-transition-element>
  </macro>
- <macro id="tx7-1" name="term 7 symbols">
  - <port-definition>
      <element-reference element="term" type="state-transition-element"
        start="start-of-data" />
      <element-reference element="eob" type="state-transition-element"
        activation="out" />
    </port-definition>
  - <state-transition-element id="term" symbol-set="Dilbert" start="start-of-data"
```

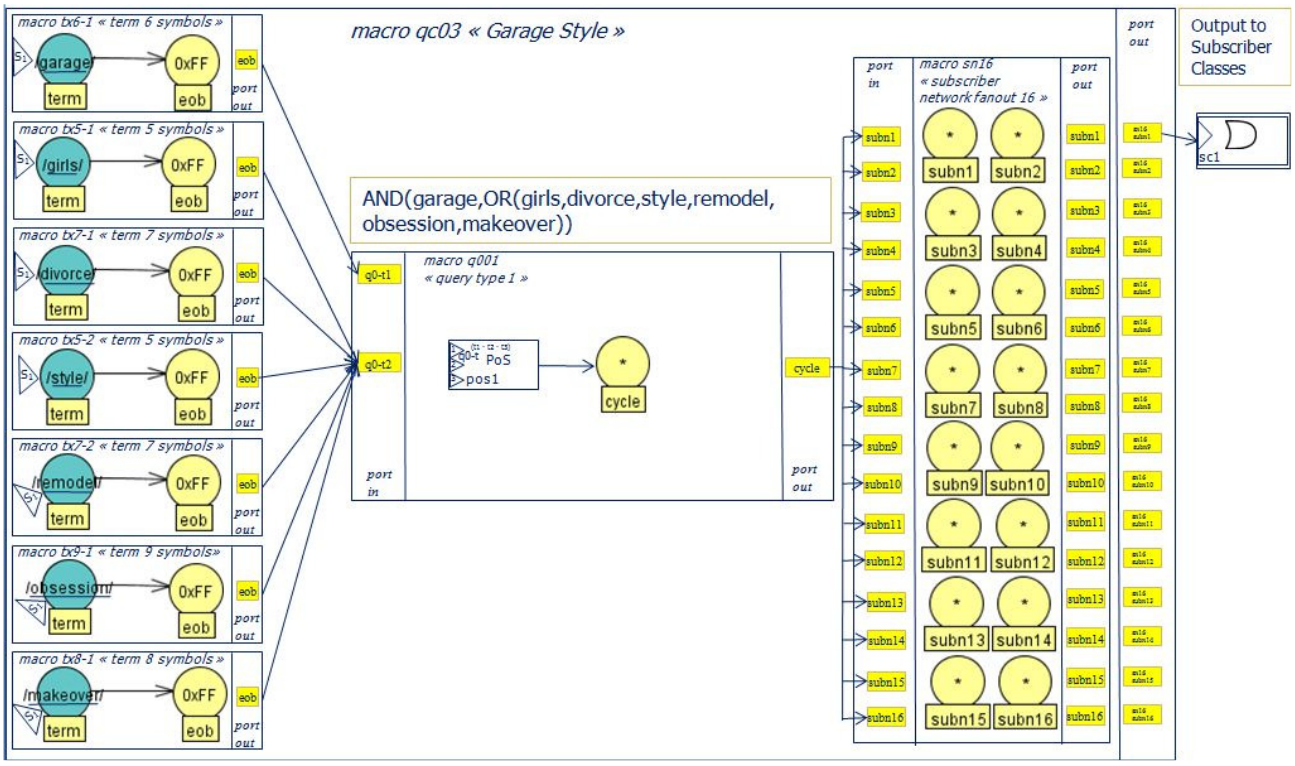
```
latch="true">
      <activate-on-match element="eob" />
    </state-transition-element>
  - <state-transition-element id="eob" symbol-set="0xFF">
      <activate-on-match macro="q002" element="cycle" />
    </state-transition-element>
  </macro>
- <macro id="q002" name="query type 2">
  - <port-definition>
      <element-reference element="q0-t1" type="sum-of-products"
        activation="in" />
      <element-reference element="q0-t2" type="sum-of-products"
        activation="in" />
      <element-reference element="cycle" type="state-transition-element"
        activation="inout" />
    </port-definition>
  - <sum-of-products id="sop1">
      <product-term id="q0-t1" />
      <product-term id="q0-t2" />
      <activate-on-high element="cycle" />
    </sum-of-products>
  - <state-transition-element id="cycle" symbol-set="*">
      <activate-on-match macro="sn16" element="subn1" />
      <activate-on-match macro="sn16" element="subn2" />
      <activate-on-match macro="sn16" element="subn3" />
      <activate-on-match macro="sn16" element="subn4" />
      <activate-on-match macro="sn16" element="subn5" />
      <activate-on-match macro="sn16" element="subn6" />
      <activate-on-match macro="sn16" element="subn7" />
      <activate-on-match macro="sn16" element="subn8" />
      <activate-on-match macro="sn16" element="subn9" />
      <activate-on-match macro="sn16" element="subn10" />
      <activate-on-match macro="sn16" element="subn11" />
      <activate-on-match macro="sn16" element="subn12" />
      <activate-on-match macro="sn16" element="subn13" />
      <activate-on-match macro="sn16" element="subn14" />
      <activate-on-match macro="sn16" element="subn15" />
      <activate-on-match macro="sn16" element="subn16" />
    </state-transition-element>
  </macro>
- <macro name="sn16" id="subscriber network fanout 16">
  - <port-definition>
      <element-reference element="subn1" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn2" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn3" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn4" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn5" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn6" type="state-transition-element"
```

```
    activation="inout" />
      <element-reference element="subn7" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn8" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn9" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn10" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn11" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn12" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn13" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn14" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn15" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn16" type="state-transition-element"
        activation="inout" />
    </port-definition>
  - <state-transition-element id="subn1" symbol-set="*">
      <activate-on-match element="sc1" />
      <activate-on-match element="sc2" />
    </state-transition-element>
    <state-transition-element id="subn2" symbol-set="*" />
    <state-transition-element id="subn3" symbol-set="*" />
    <state-transition-element id="subn4" symbol-set="*" />
    <state-transition-element id="subn5" symbol-set="*" />
    <state-transition-element id="subn6" symbol-set="*" />
    <state-transition-element id="subn7" symbol-set="*" />
    <state-transition-element id="subn8" symbol-set="*" />
    <state-transition-element id="subn9" symbol-set="*" />
    <state-transition-element id="subn10" symbol-set="*" />
    <state-transition-element id="subn11" symbol-set="*" />
    <state-transition-element id="subn12" symbol-set="*" />
    <state-transition-element id="subn13" symbol-set="*" />
    <state-transition-element id="subn14" symbol-set="*" />
    <state-transition-element id="subn15" symbol-set="*" />
    <state-transition-element id="subn16" symbol-set="*" />
  </macro>
</macro>
```

PubSub Query Class Example 3 Garage Style

PubSub Query Class Example 3: Garage Style

ANML-G Macro



ANML Template

```
<?xml version="1.0" ?>
- <macro id="qc03" name="Garage Style"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="ANML_v6.xsd">
  - <port-definition>
      <element-reference macro="sn16" element="subn1" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn2" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn3" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn4" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn5" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn6" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn7" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn8" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn9" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn10" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn11" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn12" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn13" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn14" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn15" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn16" type="state-transition-
        element" activation="out" />
    </port-definition>
  - <macro id="tx6-1" name="term 6 symbols">
    - <port-definition>
        <element-reference element="term" type="state-transition-element"
          start="start-of-data" />
        <element-reference element="eob" type="state-transition-element"
          activation="out" />
      </port-definition>
    - <state-transition-element id="term" symbol-set="garage" case-insensitive="true"
        start="start-of-data" latch="true">
        <activate-on-match element="eob" />
      </state-transition-element>
    - <state-transition-element id="eob" symbol-set="0xFF">
        <activate-on-match macro="q001" element="q0-t1" />
      </state-transition-element>
    </macro>
```

```
- <macro id="tx5-1" name="term 5 symbols">
  - <port-definition>
      <element-reference element="term" type="state-transition-element"
        start="start-of-data" />
      <element-reference element="eob" type="state-transition-element"
        activation="out" />
    </port-definition>
  - <state-transition-element id="term" symbol-set="girls" case-insensitive="true"
      start="start-of-data" latch="true">
      <activate-on-match element="eob" />
    </state-transition-element>
  - <state-transition-element id="eob" symbol-set="0xFF">
      <activate-on-match macro="q001" element="q0-t2" />
    </state-transition-element>
  </macro>
- <macro id="tx7-1" name="term 7 symbols">
  - <port-definition>
      <element-reference element="term" type="state-transition-element"
        start="start-of-data" />
      <element-reference element="eob" type="state-transition-element"
        activation="out" />
    </port-definition>
  - <state-transition-element id="term" symbol-set="divorce" case-insensitive="true"
      start="start-of-data" latch="true">
      <activate-on-match element="eob" />
    </state-transition-element>
  - <state-transition-element id="eob" symbol-set="0xFF">
      <activate-on-match macro="q001" element="q0-t2" />
    </state-transition-element>
  </macro>
- <macro id="tx5-2" name="term 5 symbols">
  - <port-definition>
      <element-reference element="term" type="state-transition-element"
        start="start-of-data" />
      <element-reference element="eob" type="state-transition-element"
        activation="out" />
    </port-definition>
  - <state-transition-element id="term" symbol-set="style" case-insensitive="false"
      start="start-of-data" latch="true">
      <activate-on-match element="eob" />
    </state-transition-element>
  - <state-transition-element id="eob" symbol-set="0xFF">
      <activate-on-match macro="q001" element="q0-t2" />
    </state-transition-element>
  </macro>
- <macro id="tx7-2" name="term 7 symbols">
  - <port-definition>
      <element-reference element="term" type="state-transition-element"
        start="start-of-data" />
      <element-reference element="eob" type="state-transition-element"
        activation="out" />
    </port-definition>
  - <state-transition-element id="term" symbol-set="remodel" case-
```

```
    insensitive="true" start="start-of-data" latch="true">
        <activate-on-match element="eob" />
      </state-transition-element>
    - <state-transition-element id="eob" symbol-set="0xFF">
        <activate-on-match macro="q001" element="q0-t2" />
      </state-transition-element>
    </macro>
  - <macro id="tx9-1" name="term 9 symbols">
    - <port-definition>
        <element-reference element="term" type="state-transition-element"
          start="start-of-data" />
        <element-reference element="eob" type="state-transition-element"
          activation="out" />
      </port-definition>
    - <state-transition-element id="term" symbol-set="obsession" case-
        insensitive="true" start="start-of-data" latch="true">
        <activate-on-match element="eob" />
      </state-transition-element>
    - <state-transition-element id="eob" symbol-set="0xFF">
        <activate-on-match macro="q001" element="q0-t2" />
      </state-transition-element>
    </macro>
  - <macro id="tx8-1" name="term 8 symbols">
    - <port-definition>
        <element-reference element="term" type="state-transition-element"
          start="start-of-data" />
        <element-reference element="eob" type="state-transition-element"
          activation="out" />
      </port-definition>
    - <state-transition-element id="term" symbol-set="makeover" start="start-of-
        data" case-insensitive="true" latch="true">
        <activate-on-match element="eob" />
      </state-transition-element>
    - <state-transition-element id="eob" symbol-set="0xFF">
        <activate-on-match macro="q001" element="q0-t2" />
      </state-transition-element>
    </macro>
  - <macro id="q001" name="query type 1">
    - <port-definition>
        <element-reference element="q0-t1" type="product-of-sums"
          activation="in" />
        <element-reference element="q0-t2" type="product-of-sums"
          activation="in" />
        <element-reference element="cycle" type="state-transition-element"
          activation="out" />
      </port-definition>
    - <product-of-sums id="pos1">
        <sum-term id="q0-t1" />
        <sum-term id="q0-t2" />
        <activate-on-high element="cycle" />
      </product-of-sums>
    - <state-transition-element id="cycle" symbol-set="*">
        <activate-on-match macro="sn16" element="subn1" />
```

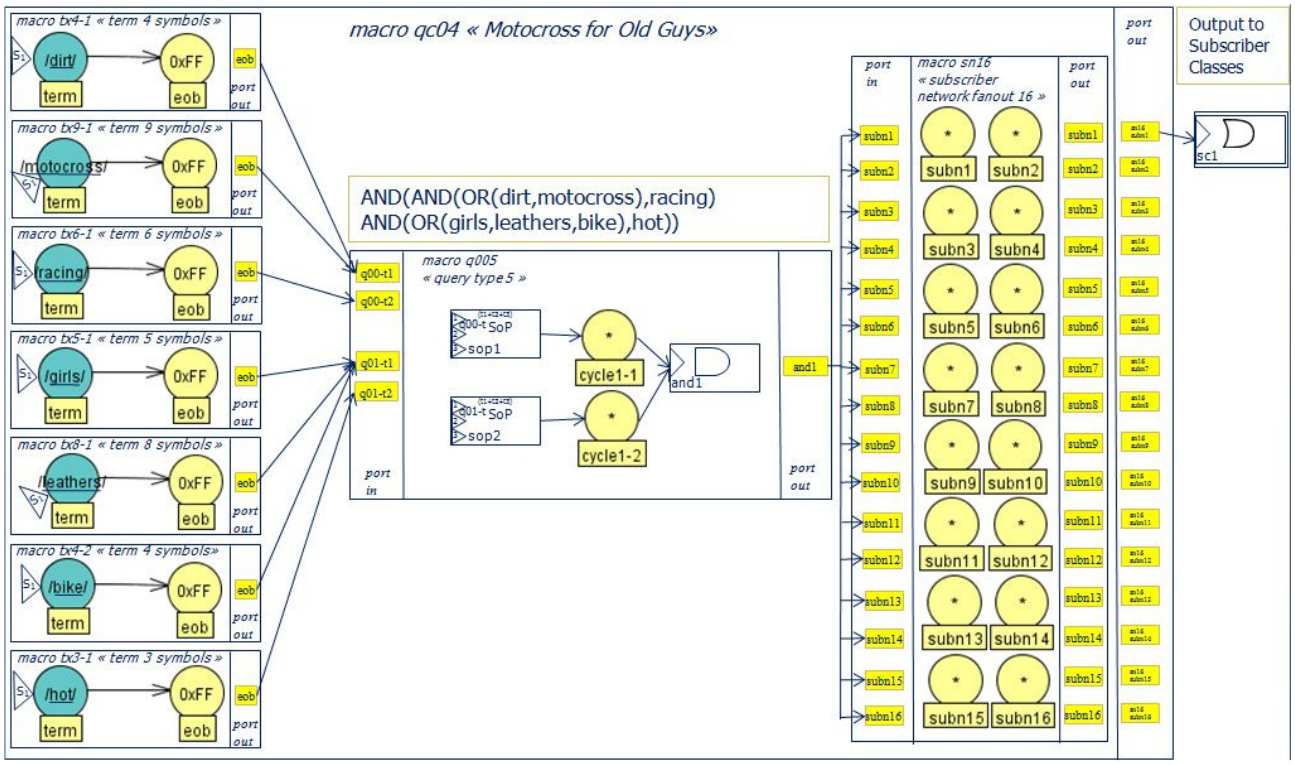
```
      <activate-on-match macro="sn16" element="subn2" />
      <activate-on-match macro="sn16" element="subn3" />
      <activate-on-match macro="sn16" element="subn4" />
      <activate-on-match macro="sn16" element="subn5" />
      <activate-on-match macro="sn16" element="subn6" />
      <activate-on-match macro="sn16" element="subn7" />
      <activate-on-match macro="sn16" element="subn8" />
      <activate-on-match macro="sn16" element="subn9" />
      <activate-on-match macro="sn16" element="subn10" />
      <activate-on-match macro="sn16" element="subn11" />
      <activate-on-match macro="sn16" element="subn12" />
      <activate-on-match macro="sn16" element="subn13" />
      <activate-on-match macro="sn16" element="subn14" />
      <activate-on-match macro="sn16" element="subn15" />
      <activate-on-match macro="sn16" element="subn16" />
    </state-transition-element>
  </macro>
- <macro name="sn16" id="subscriber network fanout 16">
  - <port-definition>
      <element-reference element="subn1" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn2" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn3" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn4" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn5" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn6" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn7" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn8" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn9" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn10" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn11" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn12" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn13" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn14" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn15" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn16" type="state-transition-element"
        activation="inout" />
    </port-definition>
  - <state-transition-element id="subn1" symbol-set="*">
```

```
      <activate-on-match element="sc1" />
    </state-transition-element>
    <state-transition-element id="subn2" symbol-set="*" />
    <state-transition-element id="subn3" symbol-set="*" />
    <state-transition-element id="subn4" symbol-set="*" />
    <state-transition-element id="subn5" symbol-set="*" />
    <state-transition-element id="subn6" symbol-set="*" />
    <state-transition-element id="subn7" symbol-set="*" />
    <state-transition-element id="subn8" symbol-set="*" />
    <state-transition-element id="subn9" symbol-set="*" />
    <state-transition-element id="subn10" symbol-set="*" />
    <state-transition-element id="subn11" symbol-set="*" />
    <state-transition-element id="subn12" symbol-set="*" />
    <state-transition-element id="subn13" symbol-set="*" />
    <state-transition-element id="subn14" symbol-set="*" />
    <state-transition-element id="subn15" symbol-set="*" />
    <state-transition-element id="subn16" symbol-set="*" />
  </macro>
</macro>
```

PubSub Query Class Example 4 Motocross

PubSub Query Class Example 4: Motocross

ANML-G Macro



ANML Template

```
<?xml version="1.0" ?>
- <macro id="qc04" name="Motocross for Old Guys"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="ANML_v6.xsd">
  - <port-definition>
      <element-reference macro="sn16" element="subn1" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn2" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn3" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn4" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn5" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn6" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn7" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn8" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn9" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn10" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn11" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn12" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn13" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn14" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn15" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn16" type="state-transition-
        element" activation="out" />
    </port-definition>
  - <macro id="tx4-1" name="term 4 symbols">
    - <port-definition>
        <element-reference element="term" type="state-transition-element"
          start="start-of-data" />
        <element-reference element="eob" type="state-transition-element"
          activation="out" />
      </port-definition>
    - <state-transition-element id="term" symbol-set="dirt" case-insensitive="true"
        start="start-of-data" latch="true">
        <activate-on-match element="eob" />
      </state-transition-element>
    - <state-transition-element id="eob" symbol-set="0xFF">
        <activate-on-match macro="q005" element="q00-t1" />
      </state-transition-element>
    </macro>
```

```
- <macro id="tx9-1" name="term 9 symbols">
  - <port-definition>
      <element-reference element="term" type="state-transition-element"
        start="start-of-data" />
      <element-reference element="eob" type="state-transition-element"
        activation="out" />
    </port-definition>
  - <state-transition-element id="term" symbol-set="motocross" case-
      insensitive="true" start="start-of-data" latch="true">
      <activate-on-match element="eob" />
    </state-transition-element>
  - <state-transition-element id="eob" symbol-set="0xFF">
      <activate-on-match macro="q005" element="q00-t1" />
    </state-transition-element>
  </macro>
- <macro id="tx6-1" name="term 6 symbols">
  - <port-definition>
      <element-reference element="term" type="state-transition-element"
        start="start-of-data" />
      <element-reference element="eob" type="state-transition-element"
        activation="out" />
    </port-definition>
  - <state-transition-element id="term" symbol-set="racing" case-insensitive="true"
      start="start-of-data" latch="true">
      <activate-on-match element="eob" />
    </state-transition-element>
  - <state-transition-element id="eob" symbol-set="0xFF">
      <activate-on-match macro="q005" element="q00-t2" />
    </state-transition-element>
  </macro>
- <macro id="tx5-1" name="term 5 symbols">
  - <port-definition>
      <element-reference element="term" type="state-transition-element"
        start="start-of-data" />
      <element-reference element="eob" type="state-transition-element"
        activation="out" />
    </port-definition>
  - <state-transition-element id="term" symbol-set="girls" case-insensitive="true"
      start="start-of-data" latch="true">
      <activate-on-match element="eob" />
    </state-transition-element>
  - <state-transition-element id="eob" symbol-set="0xFF">
      <activate-on-match macro="q005" element="q01-t1" />
    </state-transition-element>
  </macro>
- <macro id="tx8-1" name="term 8 symbols">
  - <port-definition>
      <element-reference element="term" type="state-transition-element"
        start="start-of-data" />
      <element-reference element="eob" type="state-transition-element"
        activation="out" />
    </port-definition>
  - <state-transition-element id="term" symbol-set="leathers" case-
```

```
    insensitive="true" start="start-of-data" latch="true">
        <activate-on-match element="eob" />
      </state-transition-element>
    - <state-transition-element id="eob" symbol-set="0xFF">
        <activate-on-match macro="q005" element="q01-t1" />
      </state-transition-element>
    </macro>
  - <macro id="tx4-2" name="term 4 symbols">
    - <port-definition>
        <element-reference element="term" type="state-transition-element"
          start="start-of-data" />
        <element-reference element="eob" type="state-transition-element"
          activation="out" />
      </port-definition>
    - <state-transition-element id="term" symbol-set="bike" case-insensitive="true"
        start="start-of-data" latch="true">
        <activate-on-match element="eob" />
      </state-transition-element>
    - <state-transition-element id="eob" symbol-set="0xFF">
        <activate-on-match macro="q005" element="q01-t1" />
      </state-transition-element>
    </macro>
  - <macro id="tx3-1" name="term 3 symbols">
    - <port-definition>
        <element-reference element="term" type="state-transition-element"
          start="start-of-data" />
        <element-reference element="eob" type="state-transition-element"
          activation="out" />
      </port-definition>
    - <state-transition-element id="term" symbol-set="hot" start="start-of-data" case-
        insensitive="true" latch="true">
        <activate-on-match element="eob" />
      </state-transition-element>
    - <state-transition-element id="eob" symbol-set="0xFF">
        <activate-on-match macro="q005" element="q01-t2" />
      </state-transition-element>
    </macro>
  - <macro id="q006" name="query type 6">
    - <port-definition>
        <element-reference element="q00-t1" type="sum-of-products"
          activation="in" />
        <element-reference element="q00-t2" type="sum-of-products"
          activation="in" />
        <element-reference element="q01-t1" type="sum-of-products"
          activation="in" />
        <element-reference element="q01-t2" type="sum-of-products"
          activation="in" />
        <element-reference element="and1" type="state-transition-element"
          activation="out" />
      </port-definition>
    - <sum-of-products id="sop1">
        <product-term id="q00-t1" />
        <product-term id="q00-t2" />
```

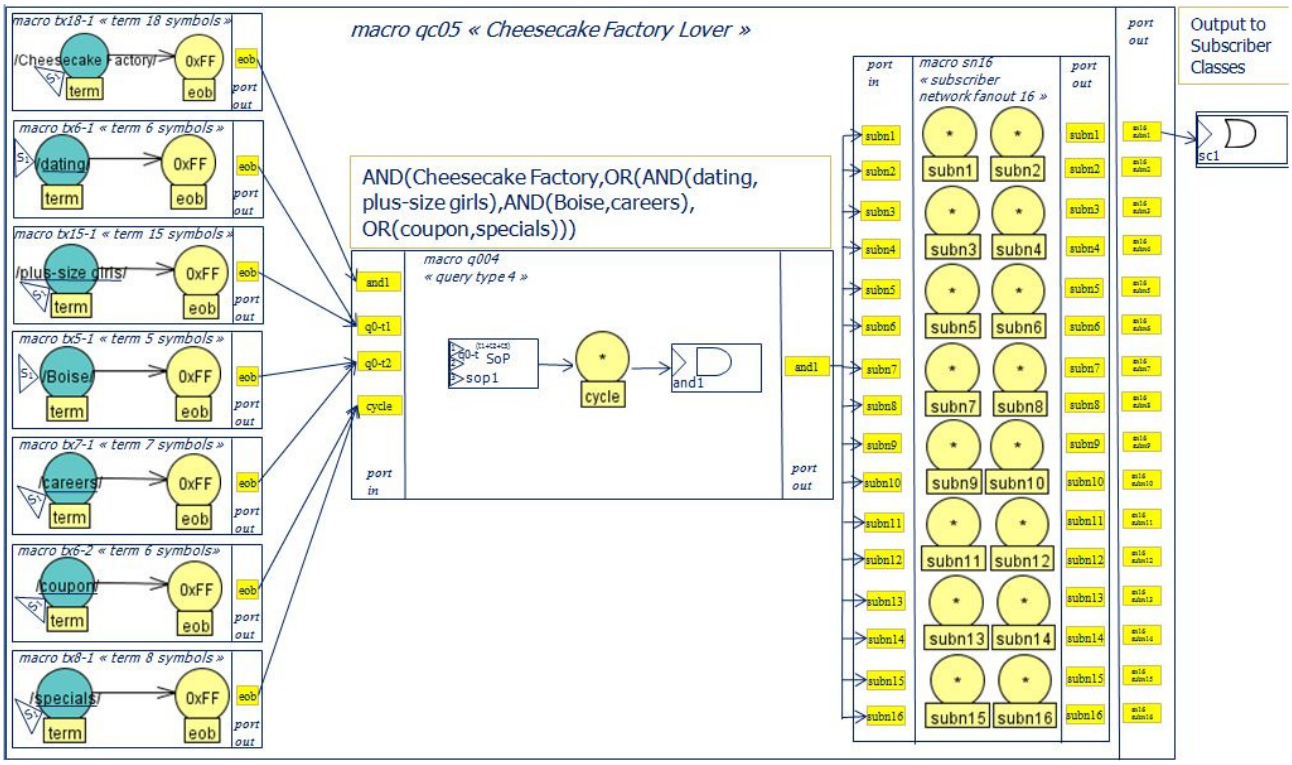
```
      <activate-on-high element="cycle1-1" />
    </sum-of-products>
  - <sum-of-products id="sop2">
      <product-term id="q01-t1" />
      <product-term id="q01-t2" />
      <activate-on-high element="cycle1-2" />
    </sum-of-products>
  - <state-transition-element id="cycle1-1" symbol-set="*">
      <activate-on-match element="and1" />
    </state-transition-element>
  - <state-transition-element id="cycle1-2" symbol-set="*">
      <activate-on-match element="and1" />
    </state-transition-element>
  - <and id="and1">
      <activate-on-high macro="sn16" element="subn1" />
      <activate-on-high macro="sn16" element="subn2" />
      <activate-on-high macro="sn16" element="subn3" />
      <activate-on-high macro="sn16" element="subn4" />
      <activate-on-high macro="sn16" element="subn5" />
      <activate-on-high macro="sn16" element="subn6" />
      <activate-on-high macro="sn16" element="subn7" />
      <activate-on-high macro="sn16" element="subn8" />
      <activate-on-high macro="sn16" element="subn9" />
      <activate-on-high macro="sn16" element="subn10" />
      <activate-on-high macro="sn16" element="subn11" />
      <activate-on-high macro="sn16" element="subn12" />
      <activate-on-high macro="sn16" element="subn13" />
      <activate-on-high macro="sn16" element="subn14" />
      <activate-on-high macro="sn16" element="subn15" />
      <activate-on-high macro="sn16" element="subn16" />
    </and>
  </macro>
- <macro name="sn16" id="subscriber network fanout 16">
  - <port-definition>
      <element-reference element="subn1" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn2" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn3" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn4" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn5" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn6" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn7" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn8" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn9" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn10" type="state-transition-element"
```

```
    activation="inout" />
      <element-reference element="subn11" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn12" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn13" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn14" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn15" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn16" type="state-transition-element"
        activation="inout" />
    </port-definition>
  - <state-transition-element id="subn1" symbol-set="*">
      <activate-on-match element="sc1" />
    </state-transition-element>
    <state-transition-element id="subn2" symbol-set="*" />
    <state-transition-element id="subn3" symbol-set="*" />
    <state-transition-element id="subn4" symbol-set="*" />
    <state-transition-element id="subn5" symbol-set="*" />
    <state-transition-element id="subn6" symbol-set="*" />
    <state-transition-element id="subn7" symbol-set="*" />
    <state-transition-element id="subn8" symbol-set="*" />
    <state-transition-element id="subn9" symbol-set="*" />
    <state-transition-element id="subn10" symbol-set="*" />
    <state-transition-element id="subn11" symbol-set="*" />
    <state-transition-element id="subn12" symbol-set="*" />
    <state-transition-element id="subn13" symbol-set="*" />
    <state-transition-element id="subn14" symbol-set="*" />
    <state-transition-element id="subn15" symbol-set="*" />
    <state-transition-element id="subn16" symbol-set="*" />
  </macro>
</macro>
```

PubSub Query Class Example 5 Cheesecake Factory Lover

PubSub Query Class Example 5: Cheesecake Factory Lover

ANML-G Macro



ANML Template

```
<?xml version="1.0" ?>
- <macro id="qc05" name="Cheesecake Factory Lover"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="ANML_v6.xsd">
  - <port-definition>
      <element-reference macro="sn16" element="subn1" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn2" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn3" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn4" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn5" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn6" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn7" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn8" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn9" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn10" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn11" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn12" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn13" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn14" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn15" type="state-transition-
        element" activation="out" />
      <element-reference macro="sn16" element="subn16" type="state-transition-
        element" activation="out" />
    </port-definition>
  - <macro id="tx18-1" name="term 18 symbols">
    - <port-definition>
        <element-reference element="term" type="state-transition-element"
          start="start-of-data" />
        <element-reference element="eob" type="state-transition-element"
          activation="out" />
      </port-definition>
    - <state-transition-element id="term" symbol-set="Cheesecake Factory"
        start="start-of-data" latch="true">
        <activate-on-match element="eob" />
      </state-transition-element>
    - <state-transition-element id="eob" symbol-set="0xFF">
        <activate-on-match macro="q004" element="and1" />
      </state-transition-element>
    </macro>
```

```
- <macro id="tx6-1" name="term 6 symbols">
  - <port-definition>
      <element-reference element="term" type="state-transition-element"
        start="start-of-data" />
      <element-reference element="eob" type="state-transition-element"
        activation="out" />
    </port-definition>
  - <state-transition-element id="term" symbol-set="dating" case-insensitive="true"
      start="start-of-data" latch="true">
      <activate-on-match element="eob" />
    </state-transition-element>
  - <state-transition-element id="eob" symbol-set="0xFF">
      <activate-on-match macro="q004" element="q0-t1" />
    </state-transition-element>
  </macro>
- <macro id="tx15-1" name="term 15 symbols">
  - <port-definition>
      <element-reference element="term" type="state-transition-element"
        start="start-of-data" />
      <element-reference element="eob" type="state-transition-element"
        activation="out" />
    </port-definition>
  - <state-transition-element id="term" symbol-set="plus-size girls" case-
      insensitive="true" start="start-of-data" latch="true">
      <activate-on-match element="eob" />
    </state-transition-element>
  - <state-transition-element id="eob" symbol-set="0xFF">
      <activate-on-match macro="q004" element="q0-t1" />
    </state-transition-element>
  </macro>
- <macro id="tx5-1" name="term 5 symbols">
  - <port-definition>
      <element-reference element="term" type="state-transition-element"
        start="start-of-data" />
      <element-reference element="eob" type="state-transition-element"
        activation="out" />
    </port-definition>
  - <state-transition-element id="term" symbol-set="Boise" start="start-of-data"
      latch="true">
      <activate-on-match element="eob" />
    </state-transition-element>
  - <state-transition-element id="eob" symbol-set="0xFF">
      <activate-on-match macro="q004" element="q0-t2" />
    </state-transition-element>
  </macro>
- <macro id="tx7-1" name="term 7 symbols">
  - <port-definition>
      <element-reference element="term" type="state-transition-element"
        start="start-of-data" />
      <element-reference element="eob" type="state-transition-element"
        activation="out" />
    </port-definition>
  - <state-transition-element id="term" symbol-set="careers" case-insensitive="true"
```

```
    start="start-of-data" latch="true">
      <activate-on-match element="eob" />
    </state-transition-element>
  - <state-transition-element id="eob" symbol-set="0xFF">
      <activate-on-match macro="q004" element="q0-t2" />
    </state-transition-element>
  </macro>
- <macro id="tx6-2" name="term 6 symbols">
  - <port-definition>
      <element-reference element="term" type="state-transition-element"
        start="start-of-data" />
      <element-reference element="eob" type="state-transition-element"
        activation="out" />
    </port-definition>
  - <state-transition-element id="term" symbol-set="coupon" case-insensitive="true"
      start="start-of-data" latch="true">
      <activate-on-match element="eob" />
    </state-transition-element>
  - <state-transition-element id="eob" symbol-set="0xFF">
      <activate-on-match macro="q004" element="cycle" />
    </state-transition-element>
  </macro>
- <macro id="tx8-1" name="term 8 symbols">
  - <port-definition>
      <element-reference element="term" type="state-transition-element"
        start="start-of-data" />
      <element-reference element="eob" type="state-transition-element"
        activation="out" />
    </port-definition>
  - <state-transition-element id="term" symbol-set="specials" start="start-of-data"
      case-insensitive="true" latch="true">
      <activate-on-match element="eob" />
    </state-transition-element>
  - <state-transition-element id="eob" symbol-set="0xFF">
      <activate-on-match macro="q004" element="cycle" />
    </state-transition-element>
  </macro>
- <macro id="q004" name="query type 4">
  - <port-definition>
      <element-reference element="and1" type="and" activation="inout" />
      <element-reference element="q0-t1" type="sum-of-products"
        activation="in" />
      <element-reference element="q0-t2" type="sum-of-products"
        activation="in" />
      <element-reference element="cycle" type="state-transition-element"
        activation="in" />
    </port-definition>
  - <sum-of-products id="sop1">
      <product-term id="q0-t1" />
      <product-term id="q0-t2" />
      <activate-on-high element="cycle" />
    </sum-of-products>
  - <state-transition-element id="cycle" symbol-set="*">
```

```
      <activate-on-match element="and1" />
    </state-transition-element>
  - <and id="and1">
      <activate-on-high macro="sn16" element="subn1" />
      <activate-on-high macro="sn16" element="subn2" />
      <activate-on-high macro="sn16" element="subn3" />
      <activate-on-high macro="sn16" element="subn4" />
      <activate-on-high macro="sn16" element="subn5" />
      <activate-on-high macro="sn16" element="subn6" />
      <activate-on-high macro="sn16" element="subn7" />
      <activate-on-high macro="sn16" element="subn8" />
      <activate-on-high macro="sn16" element="subn9" />
      <activate-on-high macro="sn16" element="subn10" />
      <activate-on-high macro="sn16" element="subn11" />
      <activate-on-high macro="sn16" element="subn12" />
      <activate-on-high macro="sn16" element="subn13" />
      <activate-on-high macro="sn16" element="subn14" />
      <activate-on-high macro="sn16" element="subn15" />
      <activate-on-high macro="sn16" element="subn16" />
    </and>
  </macro>
- <macro name="sn16" id="subscriber network fanout 16">
  - <port-definition>
      <element-reference element="subn1" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn2" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn3" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn4" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn5" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn6" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn7" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn8" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn9" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn10" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn11" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn12" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn13" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn14" type="state-transition-element"
        activation="inout" />
      <element-reference element="subn15" type="state-transition-element"
        activation="inout" />
```

```
      <element-reference element="subn16" type="state-transition-element"
        activation="inout" />
    </port-definition>
  - <state-transition-element id="subn1" symbol-set="*">
      <activate-on-match element="sc1" />
    </state-transition-element>
    <state-transition-element id="subn2" symbol-set="*" />
    <state-transition-element id="subn3" symbol-set="*" />
    <state-transition-element id="subn4" symbol-set="*" />
    <state-transition-element id="subn5" symbol-set="*" />
    <state-transition-element id="subn6" symbol-set="*" />
    <state-transition-element id="subn7" symbol-set="*" />
    <state-transition-element id="subn8" symbol-set="*" />
    <state-transition-element id="subn9" symbol-set="*" />
    <state-transition-element id="subn10" symbol-set="*" />
    <state-transition-element id="subn11" symbol-set="*" />
    <state-transition-element id="subn12" symbol-set="*" />
    <state-transition-element id="subn13" symbol-set="*" />
    <state-transition-element id="subn14" symbol-set="*" />
    <state-transition-element id="subn15" symbol-set="*" />
    <state-transition-element id="subn16" symbol-set="*" />
  </macro>
</macro>
```

Comparing Counts - anbn Problem

Comparing Counts - a^nb^n Problem

The automatons in this section will suggest approches to comparing arbitrary counts of objects, illustrated by the problem of recognizing a language in the form a^nb^n. The language a^nb^n consists of some non-zero number of 'a's followed by 'b's where the non-zero number of 'b's is either equal to or less than or equal to the number of 'a's.

Case 1: the number of 'a's and 'b's must be equal
Case 2: the number of 'b's must be equal or less than the number of 'a's

Case 1 the number of 'a's and 'b's must be equal

$a^n b^n$: the number of 'a's and 'b's must be equal

In the language for this example the number of b's seen must equal the number of a's that preceded the b's. For example,

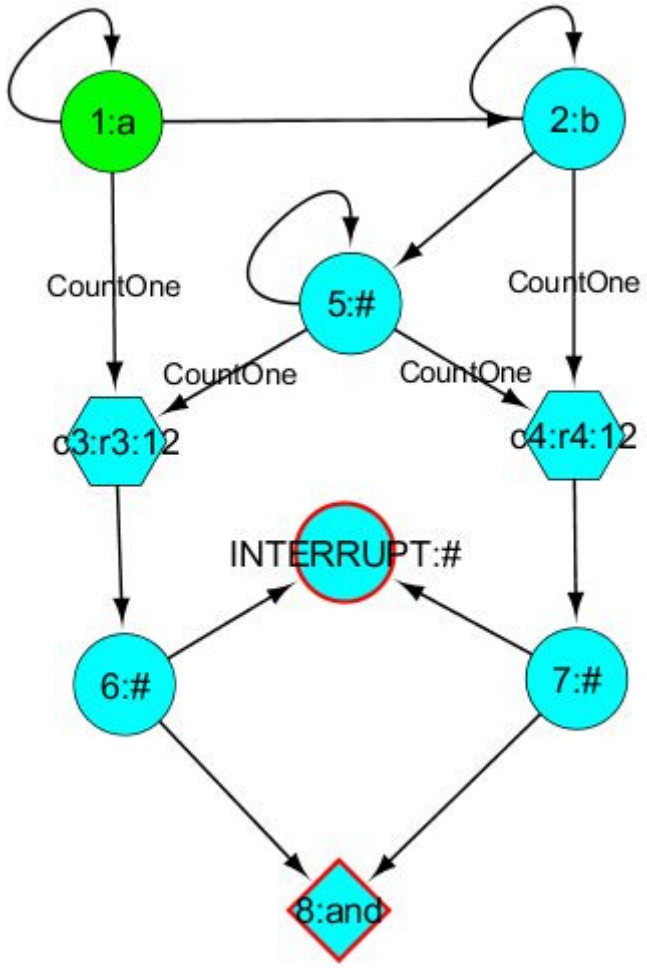
ab, aabb, aaabbb, aaaabbbb are all valid in the language and
aab, aabbb are not valid.

Our implementation will look at the entire input sequence, determining for the entire sequence if the sequence is in the language. So for aabbb we will not report accept after seeing "aabb". This is in part because I don't have a design for ANML that would report non-greedy accepts.

The implementation will also have the limitation that a maximum number for n must be selected and that additional symbols must be added to the end of the sequence to enable the result to be pumped out of the automaton. The number of additional symbols that must be added is equal to the one greater than the maximum number selected for n, however, the design may use an interrupt, if supported by the runtime, to stop processing when the result is obtained if this occurs before all extra symbols have been pumped. So, if n maximum is 12 an input stream could look like this:

aaabbb############

ANML-G



The automaton uses counters to keep track of how many a's and b's have been seen and uses the technique of continuing to trigger the counters until the target is reached used in fuzzy matching to generate an activation event, determining that the counts were equal by anding the 'a' and 'b' activation events. The actual activation for the a and b counters activates a state-transition-element which consumes a pumping character. The and element will only output if the two state-transition-elements following the counters consume the pumping symbol on the same symbol cycle, meaning the counts were equal. If the runtime support an interrupt on an output event processing will terminate at the and element if the counts were equal. If the counters did not reach target on the same symbol cycle whichever was first will trigger a final state-transition-element INTERRUPT which will end processing even if there are additional pumping symbols to be consumed one cycle after the and element test.

The 'a' and 'b' in this example should be assumed to represent more complex objects which the ANML circuit is counting. Even given this, if we assume that the actual comparison can be done on the host, we could build simpler and perhaps more efficient automata which, for example, would report each time an 'a' or 'b' is seen and we would only need to sum the number of matches for each to determine the relationship between the two counts. Since the maximum number of 'a's and 'b's is fixed in this automaton another equivalent approach would be to use the previous counters with display and the comparator. This exercise, however, aimed to explore the use of the counter element to solve this problem.

It may also be noted that a different type counter such an up-down counter which would up count the 'a's and down count the 'b's with the same counter could solve the problem far more elegantly, especially in that it would remove the need for the fixed maximum n size and the pumping symbols. A counter of this type is not available in ANML.

ANML

```
<?xml version="1.0" encoding="UTF-8" ?>
- <automata-network name="NewNetwork" id="NewNetwork"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="ANML_v6.xsd">
  - <state-transition-element id="1" symbol-set="a" start="start-of-data">
      <activate-on-match element="2" />
      <activate-on-match element="c3" />
      <activate-on-match element="1" />
    </state-transition-element>
  - <state-transition-element id="2" symbol-set="b">
      <activate-on-match element="5" />
      <activate-on-match element="c4" />
      <activate-on-match element="2" />
    </state-transition-element>
  - <counter countone="c3" reset="r3" target="12">
      <activate-on-target element="6" />
    </counter>
  - <counter countone="c4" reset="r4" target="12">
      <activate-on-target element="7" />
    </counter>
  - <state-transition-element id="5" symbol-set="#">
      <activate-on-match element="c3" />
      <activate-on-match element="c4" />
      <activate-on-match element="5" />
    </state-transition-element>
  - <state-transition-element id="6" symbol-set="#">
      <activate-on-match element="8" />
      <activate-on-match element="INTERRUPT" />
    </state-transition-element>
  - <state-transition-element id="7" symbol-set="#">
      <activate-on-match element="8" />
      <activate-on-match element="INTERRUPT" />
    </state-transition-element>
  - <and id="8">
      <report-on-high />
    </and>
  - <state-transition-element id="INTERRUPT" symbol-set="#">
      <report-on-match />
    </state-transition-element>
  - <and id="8">
      <report-on-high />
    </and>
  </automata-network>
```

Case 2 the number of 'b's must be equal or less than the number of 'a's

$a^n b^n$: the number of 'b's must be less than or equal to the number of 'a's

Please consult the article on the previous case for background on the problem and implementation.

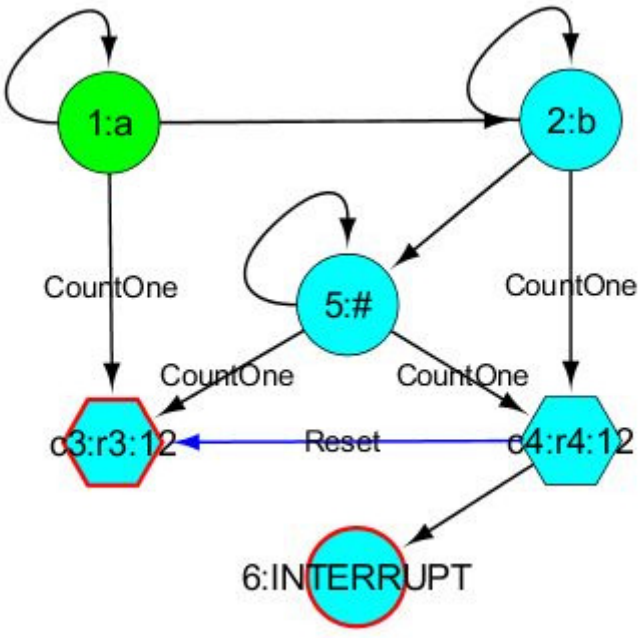
In the language for this example the number of b's seen must equal or be less than the number of a's that preceded the b's. For example,

ab, aaabbb, aaaab, aaabb are all valid in the language and
abb, aabbb are not valid.

As in the previous case, the maximum number of 'a' or 'b' symbols must be known and the input sequence must be followed by one greater than the maximum symbol number of pumping symbols.

aaaabbb############

ANML-G



ANML

If the c3 counter ('a's) reaches the target either before or at the same time as the c4 counter ('b's) our input sequence is in the language and the c3 counter outputs and stops processing through an interrupt if supported by the runtime. If the c4 counter triggers first the input sequence is not in the language and we output a negative result and stop processing on the next symbol cycle. The additional symbol cycle is needed to not output in the case where both counters reach the target at the same time. In addition, the c4 counter will reset the c3 counter to prevent both positive and negative conditions from being reported when the difference between the lengths of 'a' and 'b' sequences is 1.

Please refer to the previous case for discussion of the motivation of this example and its known deficits compared to other possible approaches.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <automata-network name="NewNetwork" id="NewNetwork"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="ANML_v6.xsd">
  - <state-transition-element id="1" symbol-set="a" start="start-of-data">
      <activate-on-match element="c3" />
      <activate-on-match element="2" />
      <activate-on-match element="1" />
    </state-transition-element>
  - <state-transition-element id="2" symbol-set="b">
      <activate-on-match element="c4" />
      <activate-on-match element="5" />
      <activate-on-match element="2" />
    </state-transition-element>
  - <counter countone="c3" reset="r3" target="12">
      <report-on-target />
    </counter>
  - <counter countone="c4" reset="r4" target="12">
      <activate-on-target element="6" />
      <activate-on-target element="r3" />
    </counter>
  - <state-transition-element id="5" symbol-set="#">
      <activate-on-match element="c4" />
      <activate-on-match element="c3" />
      <activate-on-match element="5" />
    </state-transition-element>
  - <state-transition-element id="6" symbol-set="INTERRUPT">
      <report-on-match />
    </state-transition-element>
</automata-network>
```

Number of Different Values in a Stream

Number of Different Values in a Stream

This automaton reports the number of different values seen in an input stream. For example, in the stream

4 10 4 10 9 4 9 9 10

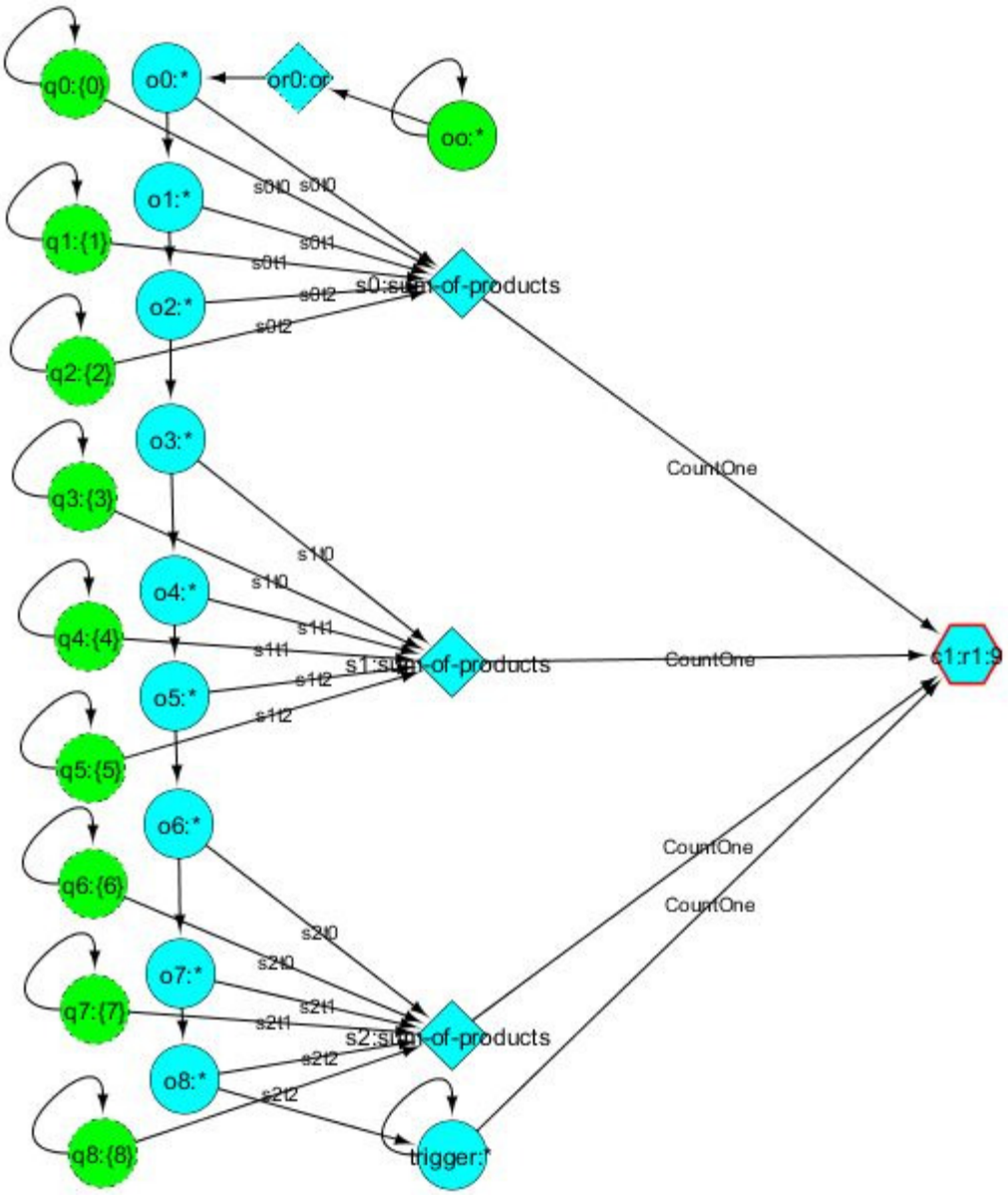
there are three different values, 4, 9 and 10. It is not a requirement that this automaton reports what the values are although one the variations does do this. The primary example below produces a single output event and the number of different values is determined by the input symbol on which this occurs. The variations produce more match results and require different techniques for determining the number of different values.

The input stream contains values associated with single symbols, expressed in the ANML bits-enabled notation as single bit values. The primary example used an end-of-data signal to mark the end of the stream of symbol values but continues processing after the EOD using "flush" data to pump out the answer. The number of cycles of flush data needed is greater than the number of possible unique values and less than twice the number of possible unique values. The examples developed here have only 9 possible unique values (0 to 8) so the following input stream would be guaranteed to work:

4 10 4 10 9 4 9 9 10 EOD # # # # # # # # # # # # # # # # # #

The '#' symbol is not required by the automaton, any symbol after the EOD will work. If the automata processor implementation does not support EOD as used here a unique marker symbol may be used instead, though at the cost of removing one of the possible symbol values.

ANML-G



ANML

The column of state-transition-elements on the left-hand side (q0-q8) each recognize one of the possible values that may occur in the input stream. These state-transition-elements are all-input starts that examine every input symbol and are latched so once a value is seen activate-on-matches will continue to be asserted. State-transition-element oo is also all-input but matches the wild-card so it will continually activate the OR element or0. or0, however, only activate-on-targets with an EOD so state-transition-element o0 won't be activated until the EOD is asserted. The state-transition-elements o0-o8 enable the activations latched by state-transition-elements q0-q8 to be propagated to a counter so the number of different value activations can be counted. They do this by going through an and in a sum-of-products element when they are asserted with the always-asserted latched activations of q0-q8. A sum-of-products element's three terms can be used for three values rather than 3 individual and elements. The chain of o0-o8 is needed so that counter is incremented once for each possible different value. If different values were connected simultaneously to the counter it would only count once even if there was more than one different value. The target value of

the counter is set to the maximum number of different symbols - in our example, 9. The counter value is "extracted" as done in other examples such as fuzzy matching and comparing counts by pumping additional symbol cycles into the counter using the trigger state-transition-element. When the counter produces output the number of different values will be known by the position of the symbol that triggered the output event.

The major disadvantage of this design is in the amount of additional data added to the input and cycles needed to pump out the result. If the input stream is very long this perhaps is not a problem. There are other approaches to the problem which lessen or entirely remove the pumping processing but which present other potential disadvantages. These variant approaches are described in the following articles:

Collating Many Single Bit Matches
Final Bit Vector in Match Results
Counting Output Pulses

```
<?xml version="1.0" encoding="UTF-8" ?>
- <automata-network name="NewNetwork" id="2"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="ANML_v6.xsd">
  - <state-transition-element id="q0" symbol-set="{0}" start="all-input" latch="true">
      <activate-on-match element="s0t0" />
    </state-transition-element>
  - <state-transition-element id="q1" symbol-set="{1}" start="all-input" latch="true">
      <activate-on-match element="s0t1" />
    </state-transition-element>
  - <state-transition-element id="q2" symbol-set="{2}" start="all-input" latch="true">
      <activate-on-match element="s0t2" />
    </state-transition-element>
  - <state-transition-element id="q3" symbol-set="{3}" start="all-input" latch="true">
      <activate-on-match element="s1t0" />
    </state-transition-element>
  - <state-transition-element id="q4" symbol-set="{4}" start="all-input" latch="true">
      <activate-on-match element="s1t1" />
    </state-transition-element>
  - <state-transition-element id="q5" symbol-set="{5}" start="all-input" latch="true">
      <activate-on-match element="s1t2" />
    </state-transition-element>
  - <state-transition-element id="q6" symbol-set="{6}" start="all-input" latch="true">
      <activate-on-match element="s2t0" />
    </state-transition-element>
  - <state-transition-element id="q7" symbol-set="{7}" start="all-input" latch="true">
      <activate-on-match element="s2t1" />
    </state-transition-element>
  - <state-transition-element id="q8" symbol-set="{8}" start="all-input" latch="true">
      <activate-on-match element="s2t2" />
    </state-transition-element>
  - <state-transition-element id="oo" symbol-set="*" start="all-input">
      <activate-on-match element="or0" />
    </state-transition-element>
  - <or id="or0" high-only-on-eod="true">
      <activate-on-high element="o0" />
    </or>
  - <state-transition-element id="o0" symbol-set="*">
      <activate-on-match element="s0t0" />
      <activate-on-match element="o1" />
    </state-transition-element>
  - <state-transition-element id="o1" symbol-set="*">
      <activate-on-match element="s0t1" />
      <activate-on-match element="o2" />
    </state-transition-element>
  - <state-transition-element id="o2" symbol-set="*">
      <activate-on-match element="s0t2" />
      <activate-on-match element="o3" />
    </state-transition-element>
  - <state-transition-element id="o3" symbol-set="*">
      <activate-on-match element="s1t0" />
      <activate-on-match element="o4" />
```

```
  </state-transition-element>
- <state-transition-element id="o4" symbol-set="*">
    <activate-on-match element="s1t1" />
    <activate-on-match element="o5" />
  </state-transition-element>
- <state-transition-element id="o5" symbol-set="*">
    <activate-on-match element="s1t2" />
    <activate-on-match element="o6" />
  </state-transition-element>
- <state-transition-element id="o6" symbol-set="*">
    <activate-on-match element="s2t0" />
    <activate-on-match element="o7" />
  </state-transition-element>
- <state-transition-element id="o7" symbol-set="*">
    <activate-on-match element="o8" />
    <activate-on-match element="s2t1" />
  </state-transition-element>
- <state-transition-element id="o8" symbol-set="*">
    <activate-on-match element="trigger" />
    <activate-on-match element="s2t2" />
  </state-transition-element>
- <state-transition-element id="trigger" symbol-set="*">
    <activate-on-match element="c1" />
    <activate-on-match element="trigger" />
  </state-transition-element>
- <sum-of-products id="s0">
    <product-term id="s0t0" />
    <product-term id="s0t1" />
    <product-term id="s0t2" />
    <activate-on-high element="c1" />
  </sum-of-products>
- <sum-of-products id="s1">
    <product-term id="s1t0" />
    <product-term id="s1t1" />
    <product-term id="s1t2" />
    <activate-on-high element="c1" />
  </sum-of-products>
- <sum-of-products id="s2">
    <product-term id="s2t0" />
    <product-term id="s2t1" />
    <product-term id="s2t2" />
    <activate-on-high element="c1" />
  </sum-of-products>
- <counter countone="c1" reset="r1" target="9">
    <report-on-target />
  </counter>
</automata-network>
```

Collating Many Single Bit Matches

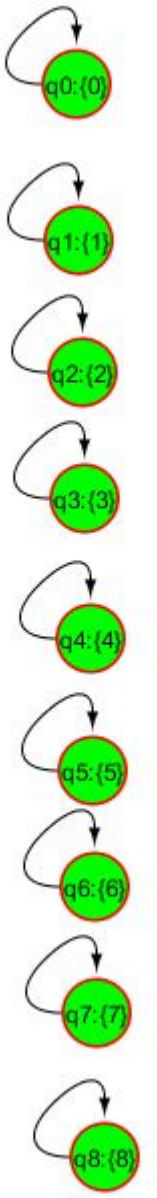
Number of Different Values - Collating Many Single Bit Matches

This automaton reports the number of different values seen in an input stream. For example, in the stream

4 10 4 10 9 4 9 9 10

there are three different values, 4, 9 and 10. Unlike the automaton in the previous article this automaton does not require that additional symbols be put in the input stream nor that the EOD be asserted prior to starting to read the additional symbols.

ANML-G



ANML

This automaton simply has one state-transition-element for each possible different value that might occur in the input stream. The state-transition-element reports when it sees the value. If a value is seen more than once it will be reported more than once. This design is, by far, the most economical in resources but requires processing of the output to eliminate duplicates and to count. This design also has the advantage that the actual values seen in the stream are identified.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <automata-network name="NewNetwork" id="NewNetwork"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="ANML_v6.xsd">
  - <state-transition-element id="q0" symbol-set="{0}" start="all-input">
      <report-on-match />
    </state-transition-element>
  - <state-transition-element id="q1" symbol-set="{1}" start="all-input">
      <report-on-match />
    </state-transition-element>
  - <state-transition-element id="q2" symbol-set="{2}" start="all-input">
      <report-on-match />
    </state-transition-element>
  - <state-transition-element id="q3" symbol-set="{3}" start="all-input">
      <report-on-match />
    </state-transition-element>
  - <state-transition-element id="q4" symbol-set="{4}" start="all-input">
      <report-on-match />
    </state-transition-element>
  - <state-transition-element id="q5" symbol-set="{5}" start="all-input">
      <report-on-match />
    </state-transition-element>
  - <state-transition-element id="q6" symbol-set="{6}" start="all-input">
      <report-on-match />
    </state-transition-element>
  - <state-transition-element id="q7" symbol-set="{7}" start="all-input">
      <report-on-match />
    </state-transition-element>
  - <state-transition-element id="q8" symbol-set="{8}" start="all-input">
      <report-on-match />
    </state-transition-element>
  </automata-network>
```

Final Bit Vector in Match Results

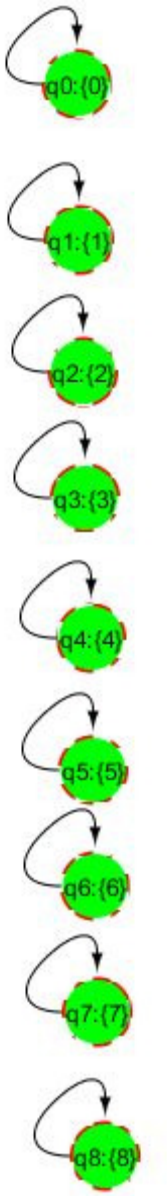
Number of Different Values - Final Bit Vector in Match Results

This automaton reports the number of different values seen in an input stream. For example, in the stream

4 10 4 10 9 4 9 9 10

there are three different values, 4, 9 and 10. Unlike the automaton in the main article this automaton does not require that additional symbols be put in the input stream nor that the EOD be asserted prior to starting to read the additional symbols. This automaton differs only slightly from the one in the previous article in how results are reported.

ANML-G



ANML

This automaton is very similar to the previous one except that each state-transition-element is latched. Once a value has been matched output will be generated for that state-transition-element on every subsequent symbol cycle. The possible advantage of this is that on the final symbol cycle every state-transition-element that has matched over the stream will output so the match bit vector for this cycle will contain the final result, only requiring that the number of match bits be counted. Both this approach and the previous one also identify which different values were seen.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <automata-network name="NewNetwork" id="NewNetwork"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="ANML_v6.xsd">
  - <state-transition-element id="q0" symbol-set="{0}" start="all-input" latch="true">
      <report-on-match />
    </state-transition-element>
  - <state-transition-element id="q1" symbol-set="{1}" start="all-input" latch="true">
      <report-on-match />
    </state-transition-element>
  - <state-transition-element id="q2" symbol-set="{2}" start="all-input" latch="true">
      <report-on-match />
    </state-transition-element>
  - <state-transition-element id="q3" symbol-set="{3}" start="all-input" latch="true">
      <report-on-match />
    </state-transition-element>
  - <state-transition-element id="q4" symbol-set="{4}" start="all-input" latch="true">
      <report-on-match />
    </state-transition-element>
  - <state-transition-element id="q5" symbol-set="{5}" start="all-input" latch="true">
      <report-on-match />
    </state-transition-element>
  - <state-transition-element id="q6" symbol-set="{6}" start="all-input" latch="true">
      <report-on-match />
    </state-transition-element>
  - <state-transition-element id="q7" symbol-set="{7}" start="all-input" latch="true">
      <report-on-match />
    </state-transition-element>
  - <state-transition-element id="q8" symbol-set="{8}" start="all-input" latch="true">
      <report-on-match />
    </state-transition-element>
  </automata-network>
```

Counting Output Pulses

Number of Different Values - Counting Output Pulses

This automaton modifies the primary example, removing a step for result processing that allows the number of different value count to be derived from a single match event, replacing it with result processing that requires counting match events.

The automaton reports the number of different values seen in an input stream. For example, in the stream

4 10 4 10 9 4 9 9 10

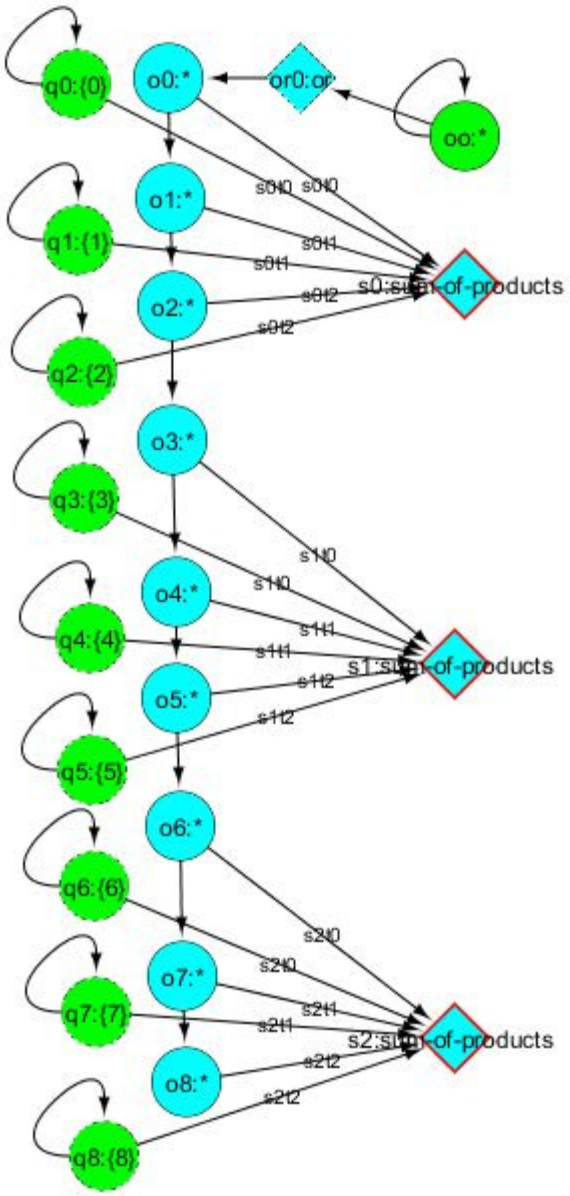
there are three different values, 4, 9 and 10.

The input stream contains values associated with single symbols, expressed in the ANML bits-enabled notation as single bit values. This example uses an end-of-data signal to mark the end of the stream of symbol values but continues processing after the EOD using "flush" data to pump out the answer. The number of cycles of flush data needed is equal to the number of possible unique values (half that required by the primary example). The example automaton below has only 9 possible unique values (0 to 8) so the following input stream would work:

4 10 4 10 9 4 9 9 10 EOD # # # # # # # # #

The '#' symbol is not required by the automaton, any symbol after the EOD will work. If the automata processor implementation does not support EOD as used here a unique marker symbol may be used instead, though at the cost of removing one of the possible symbol values.

ANML-G



ANML

The column of state-transition-elements on the left-hand side (q0-q8) each recognize one of the possible values that may occur in the input stream. These state-transition-elements are all-input starts that examine every input symbol and are latched so once a value is seen activate-on-matches will continue to be asserted. State-transition-element oo is also all-input but matches the wild-card so it will continually activate the OR element or0. or0, however, only activate-on-targets with an EOD so state-transition-element o0 won't be activated until the EOD is asserted. The state-transition-elements o0-o8 enable the activations latched by state-transition-elements q0-q8 to be propagated so the number of different value activations can be counted. They do this by going through an and in a sum-of-products element when they are asserted with the always-asserted latched activations of q0-q8. A sum-of-products element's three terms can be used for three values rather than 3 individual and elements. The sum-of-products generates output when one of terms has high input from both the value-recognizing state-transition-element and the o0-o8 output chain. The chain of o0-o8 is needed

so that a single output event is potentially generated once for each possible different value. If different values were fed to the sum-of-products at the same time it would only count once even if there was more than one different value.

The number of different values is known simply by counting the number of output events. It is also possible to know what the different values are from the symbol cycle of the pumping data associated with the output-on-high event. Advantages of this design with respect to processing are that the number of match events is exactly the number of different values, unlike the two previous designs where many spurious output events could occur and half the amount of pumping data and cycles are needed to obtain the result compared with the primary example. The previous two designs used less resources and the primary example used an additionl state-transition-element and a counter.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <automata-network name="NewNetwork" id="2"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="ANML_v6.xsd">
  - <state-transition-element id="q0" symbol-set="{0}" start="all-input" latch="true">
      <activate-on-match element="s0t0" />
    </state-transition-element>
  - <state-transition-element id="q1" symbol-set="{1}" start="all-input" latch="true">
      <activate-on-match element="s0t1" />
    </state-transition-element>
  - <state-transition-element id="q2" symbol-set="{2}" start="all-input" latch="true">
      <activate-on-match element="s0t2" />
    </state-transition-element>
  - <state-transition-element id="q3" symbol-set="{3}" start="all-input" latch="true">
      <activate-on-match element="s1t0" />
    </state-transition-element>
  - <state-transition-element id="q4" symbol-set="{4}" start="all-input" latch="true">
      <activate-on-match element="s1t1" />
    </state-transition-element>
  - <state-transition-element id="q5" symbol-set="{5}" start="all-input" latch="true">
      <activate-on-match element="s1t2" />
    </state-transition-element>
  - <state-transition-element id="q6" symbol-set="{6}" start="all-input" latch="true">
      <activate-on-match element="s2t0" />
    </state-transition-element>
  - <state-transition-element id="q7" symbol-set="{7}" start="all-input" latch="true">
      <activate-on-match element="s2t1" />
    </state-transition-element>
  - <state-transition-element id="q8" symbol-set="{8}" start="all-input" latch="true">
      <activate-on-match element="s2t2" />
    </state-transition-element>
  - <state-transition-element id="oo" symbol-set="*" start="all-input">
      <activate-on-match element="or0" />
    </state-transition-element>
  - <or id="or0" high-only-on-eod="true">
      <activate-on-high element="o0" />
    </or>
  - <state-transition-element id="o0" symbol-set="*">
      <activate-on-match element="s0t0" />
      <activate-on-match element="o1" />
    </state-transition-element>
  - <state-transition-element id="o1" symbol-set="*">
      <activate-on-match element="s0t1" />
      <activate-on-match element="o2" />
    </state-transition-element>
  - <state-transition-element id="o2" symbol-set="*">
      <activate-on-match element="s0t2" />
      <activate-on-match element="o3" />
    </state-transition-element>
  - <state-transition-element id="o3" symbol-set="*">
      <activate-on-match element="s1t0" />
      <activate-on-match element="o4" />
```

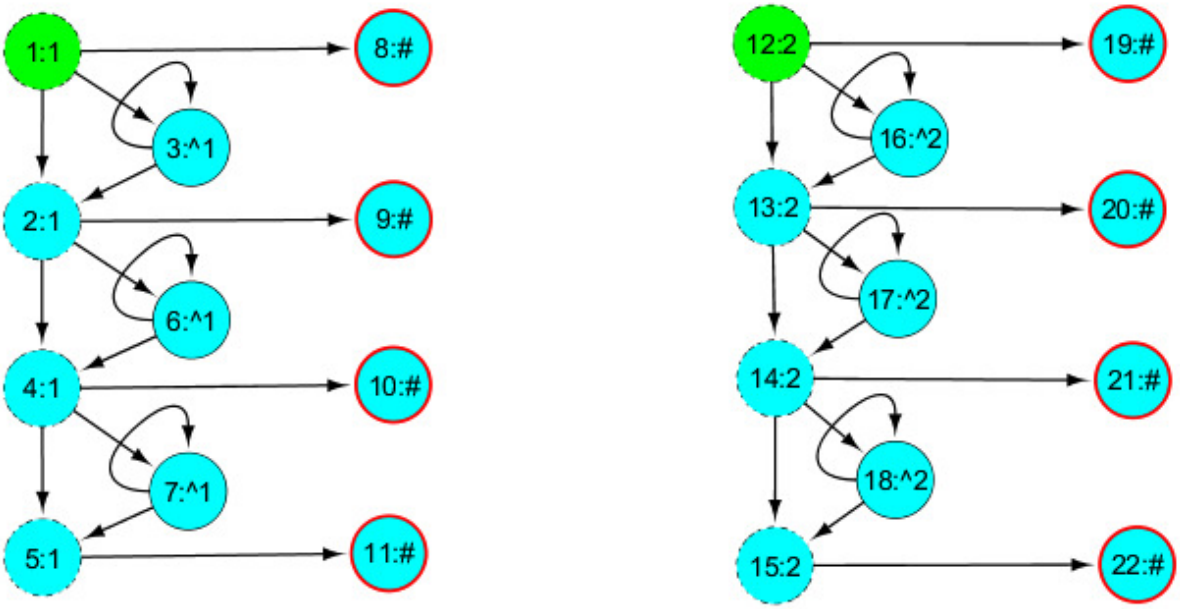
```
  </state-transition-element>
- <state-transition-element id="o4" symbol-set="*">
    <activate-on-match element="s1t1" />
    <activate-on-match element="o5" />
  </state-transition-element>
- <state-transition-element id="o5" symbol-set="*">
    <activate-on-match element="s1t2" />
    <activate-on-match element="o6" />
  </state-transition-element>
- <state-transition-element id="o6" symbol-set="*">
    <activate-on-match element="s2t0" />
    <activate-on-match element="o7" />
  </state-transition-element>
- <state-transition-element id="o7" symbol-set="*">
    <activate-on-match element="o8" />
    <activate-on-match element="s2t1" />
  </state-transition-element>
- <state-transition-element id="o8" symbol-set="*">
    <activate-on-match element="s2t2" />
  </state-transition-element>
- <sum-of-products id="s0">
    <product-term id="s0t0" />
    <product-term id="s0t1" />
    <product-term id="s0t2" />
    <report-on-high />
  </sum-of-products>
- <sum-of-products id="s1">
    <product-term id="s1t0" />
    <product-term id="s1t1" />
    <product-term id="s1t2" />
    <report-on-high />
  </sum-of-products>
- <sum-of-products id="s2">
    <product-term id="s2t0" />
    <product-term id="s2t1" />
    <product-term id="s2t2" />
    <report-on-high />
  </sum-of-products>
</automata-network>
```

Mean Machine (Number of Times a Value is Seen in a Stream)

Mean Machine

The Mean Machine automata reports the number of times a particular value is seen in the input data stream. Based on the information provided by the machine, the host can calculate the mean value. The Mean Machine automata has sub-automata to count each instance of a value seen. The max number of instances that can be counted by this machine is limited to the number of sub-automata used. The last sub-automata is an "overflow" detector, i.e. when the last sub-automata reports a match, it implies that the number of instances of a particular value has exceeded the hardware limit. The Mean Machine automata can be repeated multiple times to look for multiple values. For example, the figure below shows a Mean Machine that can count up to three instances of "1" and/or "2"; Automaton ID 11 and ID 22 report the "overflow" conditions. This machine requires a special character (#) at the end of each input data stream to work properly.

ANML-G



ANML

```
<?xml version="1.0" encoding="UTF-8" ?>
- <automata-network name="NewNetwork" id="3">
  - <state-transition-element id="10" symbol-set="#">
      <report-on-match />
    </state-transition-element>
  - <state-transition-element id="13" symbol-set="2" latch="true">
      <activate-on-match element="14" />
      <activate-on-match element="17" />
      <activate-on-match element="20" />
    </state-transition-element>
  - <state-transition-element id="15" symbol-set="2" latch="true">
      <activate-on-match element="22" />
    </state-transition-element>
  - <state-transition-element id="14" symbol-set="2" latch="true">
      <activate-on-match element="15" />
      <activate-on-match element="18" />
      <activate-on-match element="21" />
    </state-transition-element>
  - <state-transition-element id="1" symbol-set="1" start="start-of-data" latch="true">
      <activate-on-match element="3" />
      <activate-on-match element="2" />
      <activate-on-match element="8" />
    </state-transition-element>
  - <state-transition-element id="2" symbol-set="1" latch="true">
      <activate-on-match element="4" />
      <activate-on-match element="6" />
      <activate-on-match element="9" />
    </state-transition-element>
  - <state-transition-element id="5" symbol-set="1" latch="true">
      <activate-on-match element="11" />
    </state-transition-element>
  - <state-transition-element id="21" symbol-set="#">
      <report-on-match />
    </state-transition-element>
  - <state-transition-element id="8" symbol-set="#">
      <report-on-match />
    </state-transition-element>
  - <state-transition-element id="18" symbol-set="^2">
      <activate-on-match element="15" />
      <activate-on-match element="18" />
    </state-transition-element>
  - <state-transition-element id="22" symbol-set="#">
      <report-on-match />
    </state-transition-element>
  - <state-transition-element id="9" symbol-set="#">
      <report-on-match />
    </state-transition-element>
  - <state-transition-element id="16" symbol-set="^2">
      <activate-on-match element="13" />
      <activate-on-match element="16" />
    </state-transition-element>
  - <state-transition-element id="19" symbol-set="#">
```

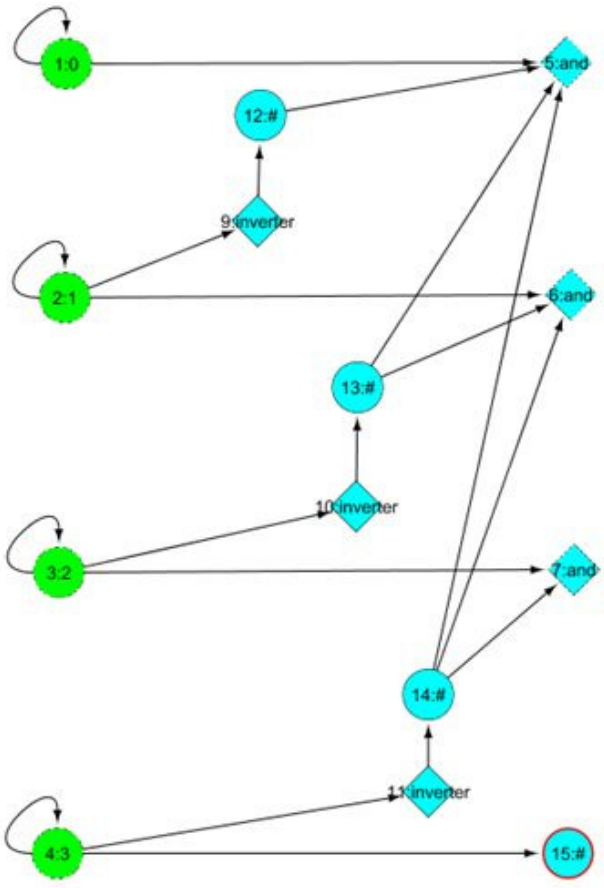
```
    <report-on-match />
  </state-transition-element>
- <state-transition-element id="20" symbol-set="#">
    <report-on-match />
  </state-transition-element>
- <state-transition-element id="11" symbol-set="#">
    <report-on-match />
  </state-transition-element>
- <state-transition-element id="6" symbol-set="^1">
    <activate-on-match element="4" />
    <activate-on-match element="6" />
  </state-transition-element>
- <state-transition-element id="7" symbol-set="^1">
    <activate-on-match element="5" />
    <activate-on-match element="7" />
  </state-transition-element>
- <state-transition-element id="4" symbol-set="1" latch="true">
    <activate-on-match element="5" />
    <activate-on-match element="7" />
    <activate-on-match element="10" />
  </state-transition-element>
- <state-transition-element id="12" symbol-set="2" start="start-of-data"
    latch="true">
    <activate-on-match element="13" />
    <activate-on-match element="19" />
    <activate-on-match element="16" />
  </state-transition-element>
- <state-transition-element id="17" symbol-set="^2">
    <activate-on-match element="14" />
    <activate-on-match element="17" />
  </state-transition-element>
- <state-transition-element id="3" symbol-set="^1">
    <activate-on-match element="2" />
    <activate-on-match element="3" />
  </state-transition-element>
</automata-network>
```

Highest Value

Highest Value

The Highest Value automata takes an input data stream and reports a single output event corresponding to the highest value in the input data stream. This automata can be modified to find the lowest value by programming the automata with values in the descending order, for example, program autmaton ID 1 with the highest expected value and the automaton ID 2 with the second highest expected value. The automata network, shown below, works for 1, 2, 3 and 4 only but it can be expanded to search up to 255 values; this automata requires a special character (#) at the end of each input data stream to work properly.

ANML-G



ANML

```
<?xml version="1.0" encoding="UTF-8" ?>
<automata-network name="NewNetwork" id="2">
  <inverter id="10">
    <activate-on-high element="13" />
  </inverter>
  <and id="5" high-only-on-eod="true" />
  <state-transition-element id="13" symbol-set="#">
    <activate-on-match element="5" />
    <activate-on-match element="6" />
  </state-transition-element>
  <inverter id="11">
    <activate-on-high element="14" />
  </inverter>
  <state-transition-element id="15" symbol-set="#">
    <report-on-match />
  </state-transition-element>
  <state-transition-element id="2" symbol-set="1" start="all-input" latch="true">
    <activate-on-match element="6" />
    <activate-on-match element="9" />
  </state-transition-element>
  <state-transition-element id="14" symbol-set="#">
    <activate-on-match element="5" />
    <activate-on-match element="6" />
    <activate-on-match element="7" />
  </state-transition-element>
  <inverter id="9">
    <activate-on-high element="12" />
  </inverter>
  <and id="7" high-only-on-eod="true" />
  <state-transition-element id="4" symbol-set="3" start="all-input" latch="true">
    <activate-on-match element="15" />
    <activate-on-match element="11" />
  </state-transition-element>
  <and id="6" high-only-on-eod="true" />
  <state-transition-element id="12" symbol-set="#">
    <activate-on-match element="5" />
  </state-transition-element>
  <state-transition-element id="3" symbol-set="2" start="all-input" latch="true">
    <activate-on-match element="7" />
    <activate-on-match element="10" />
  </state-transition-element>
  <state-transition-element id="1" symbol-set="0" start="all-input" latch="true">
    <activate-on-match element="5" />
  </state-transition-element>
</automata-network>
```

Using Counters As Store

Using Counters As Store

This automaton uses counters to store symbols and retrieve them at a later point in the processing. The example uses a language consisting of the 4 symbols used for DNA, a t c g. There are also two control symbols, the letter 'S' used as a signal to start storing symbols and the letter 'O' used as a signal to output the symbols that have been stored. The start and output symbols should be considered as placeholders for more complex processing. The stored symbols are sent to the output buffer but is equally possible to modify the design to trigger other processing without putting data in the output buffer. The symbols that are stored are symbols from the input buffer but it should also be equally possible to derive symbols to store from logic in previous steps.

There is a lag of 3 symbol cycles in calculating the symbol to be stored so the input stream should be 3 symbols longer than the length of the stored sequence. If necessary fill symbols can be used at the end. The length of stored sequence is fixed but the design of the automaton makes it possible to create an automaton supporting any length desired as long as the silicon implementation will support it. The ANML-G image below only shows a sequence of 1 symbol due to limitations at the time this article was published in the ANML-G software and handling of macros. The ANML file implements a machine with a stored sequence of 10 symbols.

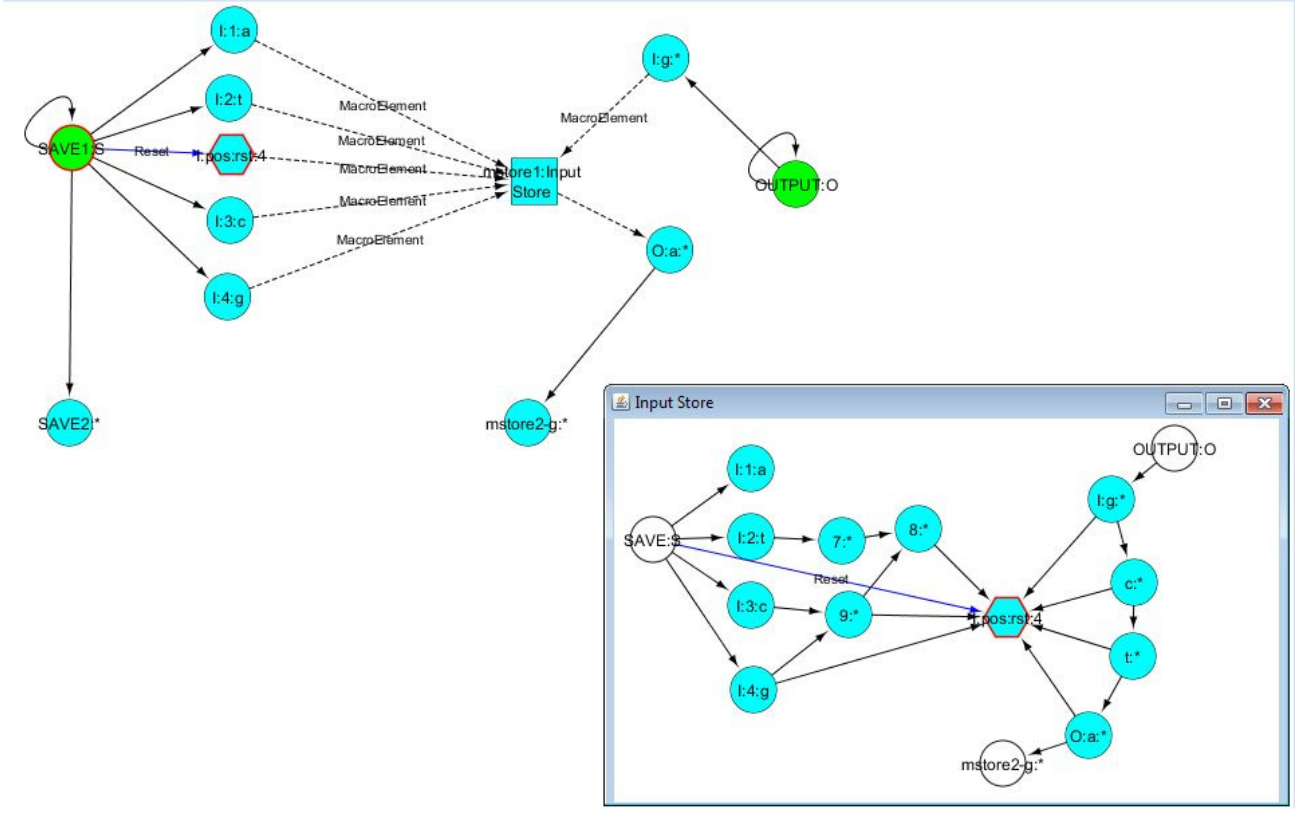
Here is an example of a possible input stream:

accgtccaaagatcSggaccaggctatacgactttccagtatgctcgaggcacOcgattccaggttcagcaccagatgatagattaca

The 10 stored sequence machine will, upon seeing the 'S', store the sequence "ggaccaggct". Upon seeing the 'O' the machine will write this sequence to the output buffer. The format of the output is that an output event (the matched symbol is irrelevant) will occur once in every four symbol cycles after the 'O' has been seen and if the event occurs in the first of the four the symbol is a 'g', the second a 'c', the third a 't', and the fourth an 'a'. Output will continue until all symbols in the stored sequence are reported, that is, in the 10 sequence case, 4 x 10 times or for 40 symbol cycles. The input buffer must be at least this long for all output to be completed. Fill symbols may be used.

ANML-G

The ANML-G image below shows only one macro, allowing for just one stored symbol. The extension of this automaton is indicated by transitions from state-transition-element SAVE1 to state-transition-element SAVE2 and the transition from macro element a to mstore2-g. In an implementation with more symbol stores there would an "Input Store" macro for each stored symbol. SAVE2 would trigger the second input store, a following SAVE state-transition element would trigger the next macro and so on. The macro a element would actually transition into the next macro, activating its g element.



ANML

The XML file included here implements a 10 symbol sequence store.

The symbol store is implemented in the "Input Store" macro. The store is implemented as a counter. If the input symbol is an 'a' the counter will remain unchanged at 4, for a 't' the counter will be decremented to '3', for a 'c' 2 and for a 'g' 1. Input elements 1 2 3 and 4 recognize the four symbols a t c g respectively. If an 'a' is seen there is no further action, for 't' the next input symbol trigger one pulse to the counter, for 'c' the next two input symbols trigger two pulses to the counter, and for 'g' the next three input symbols trigger three pulses.

The SAVE state-transition-elements form a serial chain that causes each successive input symbol to trigger the next macro in the sequence. The first SAVE, SAVE1 is trigger by seeing an 'S' in the input stream.

To get the symbol value out of the counter the counter is pulsed 4 times. If an output event occurs from the counter occurs on the first pulse the symbol was a 'g', if it takes two pulses the symbol was a 'c', three pulses a 't' and 4 an 'a'. The state-transition-element chain that pulses the counter is labelled with ids g c t a corresponding the value an output event from the counter would indicate. The output proces is triggered with a symbol 'O' seen in the input stream and recognized by the state-transition-element OUTPUT. All stored symbols are output by chaining the last state-transition-element a to the first state-transition-element g in the

```
<?xml version="1.0" encoding="UTF-8" ?>
- <automata-network name="NewNetwork" id="0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="ANML_v6.xsd">
  - <state-transition-element id="SAVE1" symbol-set="S" start="all-input">
      <report-on-match />
      <activate-on-match element="SAVE2" />
      <activate-on-match macro="mstore1" element="1" />
      <activate-on-match macro="mstore1" element="2" />
      <activate-on-match macro="mstore1" element="3" />
      <activate-on-match macro="mstore1" element="4" />
      <activate-on-match macro="mstore1" element="rst" />
    </state-transition-element>
  - <state-transition-element id="SAVE2" symbol-set="*">
      <report-on-match />
      <activate-on-match element="SAVE3" />
      <activate-on-match macro="mstore2" element="1" />
      <activate-on-match macro="mstore2" element="2" />
      <activate-on-match macro="mstore2" element="3" />
      <activate-on-match macro="mstore2" element="4" />
      <activate-on-match macro="mstore2" element="rst" />
    </state-transition-element>
  - <state-transition-element id="SAVE3" symbol-set="*">
      <report-on-match />
      <activate-on-match element="SAVE4" />
      <activate-on-match macro="mstore3" element="1" />
      <activate-on-match macro="mstore3" element="2" />
      <activate-on-match macro="mstore3" element="3" />
      <activate-on-match macro="mstore3" element="4" />
      <activate-on-match macro="mstore3" element="rst" />
    </state-transition-element>
  - <state-transition-element id="SAVE4" symbol-set="*">
      <report-on-match />
      <activate-on-match element="SAVE5" />
      <activate-on-match macro="mstore4" element="1" />
      <activate-on-match macro="mstore4" element="2" />
      <activate-on-match macro="mstore4" element="3" />
      <activate-on-match macro="mstore4" element="4" />
      <activate-on-match macro="mstore4" element="rst" />
    </state-transition-element>
  - <state-transition-element id="SAVE5" symbol-set="*">
      <report-on-match />
      <activate-on-match element="SAVE6" />
      <activate-on-match macro="mstore5" element="1" />
      <activate-on-match macro="mstore5" element="2" />
      <activate-on-match macro="mstore5" element="3" />
      <activate-on-match macro="mstore5" element="4" />
      <activate-on-match macro="mstore5" element="rst" />
    </state-transition-element>
  - <state-transition-element id="SAVE6" symbol-set="*">
      <report-on-match />
      <activate-on-match element="SAVE7" />
```

```
    <activate-on-match macro="mstore6" element="1" />
    <activate-on-match macro="mstore6" element="2" />
    <activate-on-match macro="mstore6" element="3" />
    <activate-on-match macro="mstore6" element="4" />
    <activate-on-match macro="mstore6" element="rst" />
  </state-transition-element>
- <state-transition-element id="SAVE7" symbol-set="*">
    <report-on-match />
    <activate-on-match element="SAVE8" />
    <activate-on-match macro="mstore7" element="1" />
    <activate-on-match macro="mstore7" element="2" />
    <activate-on-match macro="mstore7" element="3" />
    <activate-on-match macro="mstore7" element="4" />
    <activate-on-match macro="mstore7" element="rst" />
  </state-transition-element>
- <state-transition-element id="SAVE8" symbol-set="*">
    <report-on-match />
    <activate-on-match element="SAVE9" />
    <activate-on-match macro="mstore8" element="1" />
    <activate-on-match macro="mstore8" element="2" />
    <activate-on-match macro="mstore8" element="3" />
    <activate-on-match macro="mstore8" element="4" />
    <activate-on-match macro="mstore8" element="rst" />
  </state-transition-element>
- <state-transition-element id="SAVE9" symbol-set="*">
    <report-on-match />
    <activate-on-match element="SAVE10" />
    <activate-on-match macro="mstore9" element="1" />
    <activate-on-match macro="mstore9" element="2" />
    <activate-on-match macro="mstore9" element="3" />
    <activate-on-match macro="mstore9" element="4" />
    <activate-on-match macro="mstore9" element="rst" />
  </state-transition-element>
- <state-transition-element id="SAVE10" symbol-set="*">
    <report-on-match />
    <activate-on-match macro="mstore10" element="1" />
    <activate-on-match macro="mstore10" element="2" />
    <activate-on-match macro="mstore10" element="3" />
    <activate-on-match macro="mstore10" element="4" />
    <activate-on-match macro="mstore10" element="rst" />
  </state-transition-element>
- <state-transition-element id="OUTPUT" symbol-set="O" start="all-input">
    <activate-on-match macro="mstore1" element="g" />
  </state-transition-element>
- <macro id="mstore1" name="Input Store">
  - <port-definition>
      <element-reference element="1" type="state-transition-element"
        activation="in" />
      <element-reference element="2" type="state-transition-element"
        activation="in" />
      <element-reference element="3" type="state-transition-element"
        activation="in" />
```

```
      <element-reference element="4" type="state-transition-element"
        activation="in" />
      <element-reference element="rst" type="counter" activation="in"
        reporting="true" />
      <element-reference element="g" type="state-transition-element"
        activation="in" />
      <element-reference element="a" type="state-transition-element"
        activation="out" />
    </port-definition>
    <state-transition-element id="1" symbol-set="a" />
  - <state-transition-element id="2" symbol-set="t">
      <activate-on-match element="7" />
    </state-transition-element>
  - <state-transition-element id="3" symbol-set="c">
      <activate-on-match element="9" />
    </state-transition-element>
  - <state-transition-element id="4" symbol-set="g">
      <activate-on-match element="9" />
      <activate-on-match element="pos" />
    </state-transition-element>
  - <state-transition-element id="7" symbol-set="*">
      <activate-on-match element="8" />
    </state-transition-element>
  - <state-transition-element id="8" symbol-set="*">
      <activate-on-match element="pos" />
    </state-transition-element>
  - <state-transition-element id="9" symbol-set="*">
      <activate-on-match element="8" />
      <activate-on-match element="pos" />
    </state-transition-element>
  - <counter countone="pos" reset="rst" target="4">
      <report-on-target />
    </counter>
  - <state-transition-element id="g" symbol-set="*">
      <activate-on-match element="pos" />
      <activate-on-match element="c" />
    </state-transition-element>
  - <state-transition-element id="c" symbol-set="*">
      <activate-on-match element="pos" />
      <activate-on-match element="t" />
    </state-transition-element>
  - <state-transition-element id="t" symbol-set="*">
      <activate-on-match element="pos" />
      <activate-on-match element="a" />
    </state-transition-element>
  - <state-transition-element id="a" symbol-set="*">
      <activate-on-match element="pos" />
      <activate-on-match macro="mstore2" element="g" />
    </state-transition-element>
  </macro>
- <macro id="mstore2" name="Input Store">
  - <port-definition>
```

```
    <element-reference element="1" type="state-transition-element"
      activation="in" />
    <element-reference element="2" type="state-transition-element"
      activation="in" />
    <element-reference element="3" type="state-transition-element"
      activation="in" />
    <element-reference element="4" type="state-transition-element"
      activation="in" />
    <element-reference element="rst" type="counter" activation="in"
      reporting="true" />
    <element-reference element="g" type="state-transition-element"
      activation="in" />
    <element-reference element="a" type="state-transition-element"
      activation="out" />
  </port-definition>
  <state-transition-element id="1" symbol-set="a" />
- <state-transition-element id="2" symbol-set="t">
    <activate-on-match element="7" />
  </state-transition-element>
- <state-transition-element id="3" symbol-set="c">
    <activate-on-match element="9" />
  </state-transition-element>
- <state-transition-element id="4" symbol-set="g">
    <activate-on-match element="9" />
    <activate-on-match element="pos" />
  </state-transition-element>
- <state-transition-element id="7" symbol-set="*">
    <activate-on-match element="8" />
  </state-transition-element>
- <state-transition-element id="8" symbol-set="*">
    <activate-on-match element="pos" />
  </state-transition-element>
- <state-transition-element id="9" symbol-set="*">
    <activate-on-match element="8" />
    <activate-on-match element="pos" />
  </state-transition-element>
- <counter countone="pos" reset="rst" target="4">
    <report-on-target />
  </counter>
- <state-transition-element id="g" symbol-set="*">
    <activate-on-match element="pos" />
    <activate-on-match element="c" />
  </state-transition-element>
- <state-transition-element id="c" symbol-set="*">
    <activate-on-match element="pos" />
    <activate-on-match element="t" />
  </state-transition-element>
- <state-transition-element id="t" symbol-set="*">
    <activate-on-match element="pos" />
    <activate-on-match element="a" />
  </state-transition-element>
- <state-transition-element id="a" symbol-set="*">
```

```
      <activate-on-match element="pos" />
      <activate-on-match macro="mstore3" element="g" />
    </state-transition-element>
  </macro>
- <macro id="mstore3" name="Input Store">
  - <port-definition>
      <element-reference element="1" type="state-transition-element"
        activation="in" />
      <element-reference element="2" type="state-transition-element"
        activation="in" />
      <element-reference element="3" type="state-transition-element"
        activation="in" />
      <element-reference element="4" type="state-transition-element"
        activation="in" />
      <element-reference element="rst" type="counter" activation="in"
        reporting="true" />
      <element-reference element="g" type="state-transition-element"
        activation="in" />
      <element-reference element="a" type="state-transition-element"
        activation="out" />
    </port-definition>
    <state-transition-element id="1" symbol-set="a" />
  - <state-transition-element id="2" symbol-set="t">
      <activate-on-match element="7" />
    </state-transition-element>
  - <state-transition-element id="3" symbol-set="c">
      <activate-on-match element="9" />
    </state-transition-element>
  - <state-transition-element id="4" symbol-set="g">
      <activate-on-match element="9" />
      <activate-on-match element="pos" />
    </state-transition-element>
  - <state-transition-element id="7" symbol-set="*">
      <activate-on-match element="8" />
    </state-transition-element>
  - <state-transition-element id="8" symbol-set="*">
      <activate-on-match element="pos" />
    </state-transition-element>
  - <state-transition-element id="9" symbol-set="*">
      <activate-on-match element="8" />
      <activate-on-match element="pos" />
    </state-transition-element>
  - <counter countone="pos" reset="rst" target="4">
      <report-on-target />
    </counter>
  - <state-transition-element id="g" symbol-set="*">
      <activate-on-match element="pos" />
      <activate-on-match element="c" />
    </state-transition-element>
  - <state-transition-element id="c" symbol-set="*">
      <activate-on-match element="pos" />
      <activate-on-match element="t" />
```

```
    </state-transition-element>
  - <state-transition-element id="t" symbol-set="*">
      <activate-on-match element="pos" />
      <activate-on-match element="a" />
    </state-transition-element>
  - <state-transition-element id="a" symbol-set="*">
      <activate-on-match element="pos" />
      <activate-on-match macro="mstore4" element="g" />
    </state-transition-element>
  </macro>
- <macro id="mstore4" name="Input Store">
  - <port-definition>
      <element-reference element="1" type="state-transition-element"
        activation="in" />
      <element-reference element="2" type="state-transition-element"
        activation="in" />
      <element-reference element="3" type="state-transition-element"
        activation="in" />
      <element-reference element="4" type="state-transition-element"
        activation="in" />
      <element-reference element="rst" type="counter" activation="in"
        reporting="true" />
      <element-reference element="g" type="state-transition-element"
        activation="in" />
      <element-reference element="a" type="state-transition-element"
        activation="out" />
    </port-definition>
    <state-transition-element id="1" symbol-set="a" />
  - <state-transition-element id="2" symbol-set="t">
      <activate-on-match element="7" />
    </state-transition-element>
  - <state-transition-element id="3" symbol-set="c">
      <activate-on-match element="9" />
    </state-transition-element>
  - <state-transition-element id="4" symbol-set="g">
      <activate-on-match element="9" />
      <activate-on-match element="pos" />
    </state-transition-element>
  - <state-transition-element id="7" symbol-set="*">
      <activate-on-match element="8" />
    </state-transition-element>
  - <state-transition-element id="8" symbol-set="*">
      <activate-on-match element="pos" />
    </state-transition-element>
  - <state-transition-element id="9" symbol-set="*">
      <activate-on-match element="8" />
      <activate-on-match element="pos" />
    </state-transition-element>
  - <counter countone="pos" reset="rst" target="4">
      <report-on-target />
    </counter>
  - <state-transition-element id="g" symbol-set="*">
```

```
      <activate-on-match element="pos" />
      <activate-on-match element="c" />
    </state-transition-element>
  - <state-transition-element id="c" symbol-set="*">
      <activate-on-match element="pos" />
      <activate-on-match element="t" />
    </state-transition-element>
  - <state-transition-element id="t" symbol-set="*">
      <activate-on-match element="pos" />
      <activate-on-match element="a" />
    </state-transition-element>
  - <state-transition-element id="a" symbol-set="*">
      <activate-on-match element="pos" />
      <activate-on-match macro="mstore5" element="g" />
    </state-transition-element>
  </macro>
- <macro id="mstore5" name="Input Store">
  - <port-definition>
      <element-reference element="1" type="state-transition-element"
        activation="in" />
      <element-reference element="2" type="state-transition-element"
        activation="in" />
      <element-reference element="3" type="state-transition-element"
        activation="in" />
      <element-reference element="4" type="state-transition-element"
        activation="in" />
      <element-reference element="rst" type="counter" activation="in"
        reporting="true" />
      <element-reference element="g" type="state-transition-element"
        activation="in" />
      <element-reference element="a" type="state-transition-element"
        activation="out" />
    </port-definition>
    <state-transition-element id="1" symbol-set="a" />
  - <state-transition-element id="2" symbol-set="t">
      <activate-on-match element="7" />
    </state-transition-element>
  - <state-transition-element id="3" symbol-set="c">
      <activate-on-match element="9" />
    </state-transition-element>
  - <state-transition-element id="4" symbol-set="g">
      <activate-on-match element="9" />
      <activate-on-match element="pos" />
    </state-transition-element>
  - <state-transition-element id="7" symbol-set="*">
      <activate-on-match element="8" />
    </state-transition-element>
  - <state-transition-element id="8" symbol-set="*">
      <activate-on-match element="pos" />
    </state-transition-element>
  - <state-transition-element id="9" symbol-set="*">
      <activate-on-match element="8" />
```

```
      <activate-on-match element="pos" />
    </state-transition-element>
  - <counter countone="pos" reset="rst" target="4">
      <report-on-target />
    </counter>
  - <state-transition-element id="g" symbol-set="*">
      <activate-on-match element="pos" />
      <activate-on-match element="c" />
    </state-transition-element>
  - <state-transition-element id="c" symbol-set="*">
      <activate-on-match element="pos" />
      <activate-on-match element="t" />
    </state-transition-element>
  - <state-transition-element id="t" symbol-set="*">
      <activate-on-match element="pos" />
      <activate-on-match element="a" />
    </state-transition-element>
  - <state-transition-element id="a" symbol-set="*">
      <activate-on-match element="pos" />
      <activate-on-match macro="mstore6" element="g" />
    </state-transition-element>
  </macro>
- <macro id="mstore6" name="Input Store">
  - <port-definition>
      <element-reference element="1" type="state-transition-element"
        activation="in" />
      <element-reference element="2" type="state-transition-element"
        activation="in" />
      <element-reference element="3" type="state-transition-element"
        activation="in" />
      <element-reference element="4" type="state-transition-element"
        activation="in" />
      <element-reference element="rst" type="counter" activation="in"
        reporting="true" />
      <element-reference element="g" type="state-transition-element"
        activation="in" />
      <element-reference element="a" type="state-transition-element"
        activation="out" />
    </port-definition>
    <state-transition-element id="1" symbol-set="a" />
  - <state-transition-element id="2" symbol-set="t">
      <activate-on-match element="7" />
    </state-transition-element>
  - <state-transition-element id="3" symbol-set="c">
      <activate-on-match element="9" />
    </state-transition-element>
  - <state-transition-element id="4" symbol-set="g">
      <activate-on-match element="9" />
      <activate-on-match element="pos" />
    </state-transition-element>
  - <state-transition-element id="7" symbol-set="*">
      <activate-on-match element="8" />
```

```
    </state-transition-element>
  - <state-transition-element id="8" symbol-set="*">
      <activate-on-match element="pos" />
    </state-transition-element>
  - <state-transition-element id="9" symbol-set="*">
      <activate-on-match element="8" />
      <activate-on-match element="pos" />
    </state-transition-element>
  - <counter countone="pos" reset="rst" target="4">
      <report-on-target />
    </counter>
  - <state-transition-element id="g" symbol-set="*">
      <activate-on-match element="pos" />
      <activate-on-match element="c" />
    </state-transition-element>
  - <state-transition-element id="c" symbol-set="*">
      <activate-on-match element="pos" />
      <activate-on-match element="t" />
    </state-transition-element>
  - <state-transition-element id="t" symbol-set="*">
      <activate-on-match element="pos" />
      <activate-on-match element="a" />
    </state-transition-element>
  - <state-transition-element id="a" symbol-set="*">
      <activate-on-match element="pos" />
      <activate-on-match macro="mstore7" element="g" />
    </state-transition-element>
  </macro>
- <macro id="mstore7" name="Input Store">
  - <port-definition>
      <element-reference element="1" type="state-transition-element"
        activation="in" />
      <element-reference element="2" type="state-transition-element"
        activation="in" />
      <element-reference element="3" type="state-transition-element"
        activation="in" />
      <element-reference element="4" type="state-transition-element"
        activation="in" />
      <element-reference element="rst" type="counter" activation="in"
        reporting="true" />
      <element-reference element="g" type="state-transition-element"
        activation="in" />
      <element-reference element="a" type="state-transition-element"
        activation="out" />
    </port-definition>
    <state-transition-element id="1" symbol-set="a" />
  - <state-transition-element id="2" symbol-set="t">
      <activate-on-match element="7" />
    </state-transition-element>
  - <state-transition-element id="3" symbol-set="c">
      <activate-on-match element="9" />
    </state-transition-element>
```

```
- <state-transition-element id="4" symbol-set="g">
    <activate-on-match element="9" />
    <activate-on-match element="pos" />
  </state-transition-element>
- <state-transition-element id="7" symbol-set="*">
    <activate-on-match element="8" />
  </state-transition-element>
- <state-transition-element id="8" symbol-set="*">
    <activate-on-match element="pos" />
  </state-transition-element>
- <state-transition-element id="9" symbol-set="*">
    <activate-on-match element="8" />
    <activate-on-match element="pos" />
  </state-transition-element>
- <counter countone="pos" reset="rst" target="4">
    <report-on-target />
  </counter>
- <state-transition-element id="g" symbol-set="*">
    <activate-on-match element="pos" />
    <activate-on-match element="c" />
  </state-transition-element>
- <state-transition-element id="c" symbol-set="*">
    <activate-on-match element="pos" />
    <activate-on-match element="t" />
  </state-transition-element>
- <state-transition-element id="t" symbol-set="*">
    <activate-on-match element="pos" />
    <activate-on-match element="a" />
  </state-transition-element>
- <state-transition-element id="a" symbol-set="*">
    <activate-on-match element="pos" />
    <activate-on-match macro="mstore8" element="g" />
  </state-transition-element>
</macro>
- <macro id="mstore8" name="Input Store">
- <port-definition>
    <element-reference element="1" type="state-transition-element"
      activation="in" />
    <element-reference element="2" type="state-transition-element"
      activation="in" />
    <element-reference element="3" type="state-transition-element"
      activation="in" />
    <element-reference element="4" type="state-transition-element"
      activation="in" />
    <element-reference element="rst" type="counter" activation="in"
      reporting="true" />
    <element-reference element="g" type="state-transition-element"
      activation="in" />
    <element-reference element="a" type="state-transition-element"
      activation="out" />
  </port-definition>
  <state-transition-element id="1" symbol-set="a" />
```

```
- <state-transition-element id="2" symbol-set="t">
    <activate-on-match element="7" />
  </state-transition-element>
- <state-transition-element id="3" symbol-set="c">
    <activate-on-match element="9" />
  </state-transition-element>
- <state-transition-element id="4" symbol-set="g">
    <activate-on-match element="9" />
    <activate-on-match element="pos" />
  </state-transition-element>
- <state-transition-element id="7" symbol-set="*">
    <activate-on-match element="8" />
  </state-transition-element>
- <state-transition-element id="8" symbol-set="*">
    <activate-on-match element="pos" />
  </state-transition-element>
- <state-transition-element id="9" symbol-set="*">
    <activate-on-match element="8" />
    <activate-on-match element="pos" />
  </state-transition-element>
- <counter countone="pos" reset="rst" target="4">
    <report-on-target />
  </counter>
- <state-transition-element id="g" symbol-set="*">
    <activate-on-match element="pos" />
    <activate-on-match element="c" />
  </state-transition-element>
- <state-transition-element id="c" symbol-set="*">
    <activate-on-match element="pos" />
    <activate-on-match element="t" />
  </state-transition-element>
- <state-transition-element id="t" symbol-set="*">
    <activate-on-match element="pos" />
    <activate-on-match element="a" />
  </state-transition-element>
- <state-transition-element id="a" symbol-set="*">
    <activate-on-match element="pos" />
    <activate-on-match macro="mstore9" element="g" />
  </state-transition-element>
</macro>
- <macro id="mstore9" name="Input Store">
- <port-definition>
    <element-reference element="1" type="state-transition-element"
      activation="in" />
    <element-reference element="2" type="state-transition-element"
      activation="in" />
    <element-reference element="3" type="state-transition-element"
      activation="in" />
    <element-reference element="4" type="state-transition-element"
      activation="in" />
    <element-reference element="rst" type="counter" activation="in"
      reporting="true" />
```

```
      <element-reference element="g" type="state-transition-element"
        activation="in" />
      <element-reference element="a" type="state-transition-element"
        activation="out" />
    </port-definition>
    <state-transition-element id="1" symbol-set="a" />
  - <state-transition-element id="2" symbol-set="t">
      <activate-on-match element="7" />
    </state-transition-element>
  - <state-transition-element id="3" symbol-set="c">
      <activate-on-match element="9" />
    </state-transition-element>
  - <state-transition-element id="4" symbol-set="g">
      <activate-on-match element="9" />
      <activate-on-match element="pos" />
    </state-transition-element>
  - <state-transition-element id="7" symbol-set="*">
      <activate-on-match element="8" />
    </state-transition-element>
  - <state-transition-element id="8" symbol-set="*">
      <activate-on-match element="pos" />
    </state-transition-element>
  - <state-transition-element id="9" symbol-set="*">
      <activate-on-match element="8" />
      <activate-on-match element="pos" />
    </state-transition-element>
  - <counter countone="pos" reset="rst" target="4">
      <report-on-target />
    </counter>
  - <state-transition-element id="g" symbol-set="*">
      <activate-on-match element="pos" />
      <activate-on-match element="c" />
    </state-transition-element>
  - <state-transition-element id="c" symbol-set="*">
      <activate-on-match element="pos" />
      <activate-on-match element="t" />
    </state-transition-element>
  - <state-transition-element id="t" symbol-set="*">
      <activate-on-match element="pos" />
      <activate-on-match element="a" />
    </state-transition-element>
  - <state-transition-element id="a" symbol-set="*">
      <activate-on-match element="pos" />
      <activate-on-match macro="mstore10" element="g" />
    </state-transition-element>
  </macro>
- <macro id="mstore10" name="Input Store">
  - <port-definition>
      <element-reference element="1" type="state-transition-element"
        activation="in" />
      <element-reference element="2" type="state-transition-element"
        activation="in" />
```

```
      <element-reference element="3" type="state-transition-element"
        activation="in" />
      <element-reference element="4" type="state-transition-element"
        activation="in" />
      <element-reference element="rst" type="counter" activation="in"
        reporting="true" />
      <element-reference element="g" type="state-transition-element"
        activation="in" />
      <element-reference element="a" type="state-transition-element"
        activation="out" />
    </port-definition>
    <state-transition-element id="1" symbol-set="a" />
  - <state-transition-element id="2" symbol-set="t">
      <activate-on-match element="7" />
    </state-transition-element>
  - <state-transition-element id="3" symbol-set="c">
      <activate-on-match element="9" />
    </state-transition-element>
  - <state-transition-element id="4" symbol-set="g">
      <activate-on-match element="9" />
      <activate-on-match element="pos" />
    </state-transition-element>
  - <state-transition-element id="7" symbol-set="*">
      <activate-on-match element="8" />
    </state-transition-element>
  - <state-transition-element id="8" symbol-set="*">
      <activate-on-match element="pos" />
    </state-transition-element>
  - <state-transition-element id="9" symbol-set="*">
      <activate-on-match element="8" />
      <activate-on-match element="pos" />
    </state-transition-element>
  - <counter countone="pos" reset="rst" target="4">
      <report-on-target />
    </counter>
  - <state-transition-element id="g" symbol-set="*">
      <activate-on-match element="pos" />
      <activate-on-match element="c" />
    </state-transition-element>
  - <state-transition-element id="c" symbol-set="*">
      <activate-on-match element="pos" />
      <activate-on-match element="t" />
    </state-transition-element>
  - <state-transition-element id="t" symbol-set="*">
      <activate-on-match element="pos" />
      <activate-on-match element="a" />
    </state-transition-element>
  - <state-transition-element id="a" symbol-set="*">
      <activate-on-match element="pos" />
    </state-transition-element>
  </macro>
</automata-network>
```

Copying Counter Values

Copying Counter Values

This automaton shows how to copy the value in a counter to another counter. In this example the counter is connected to a "bank" of six counters, enabling the counter to be reused and up to six different counter values to be stored in the bank. The maximum count of a counter and the corresponding value that can copied is limited to the configured target value. This example uses a target value of 15.

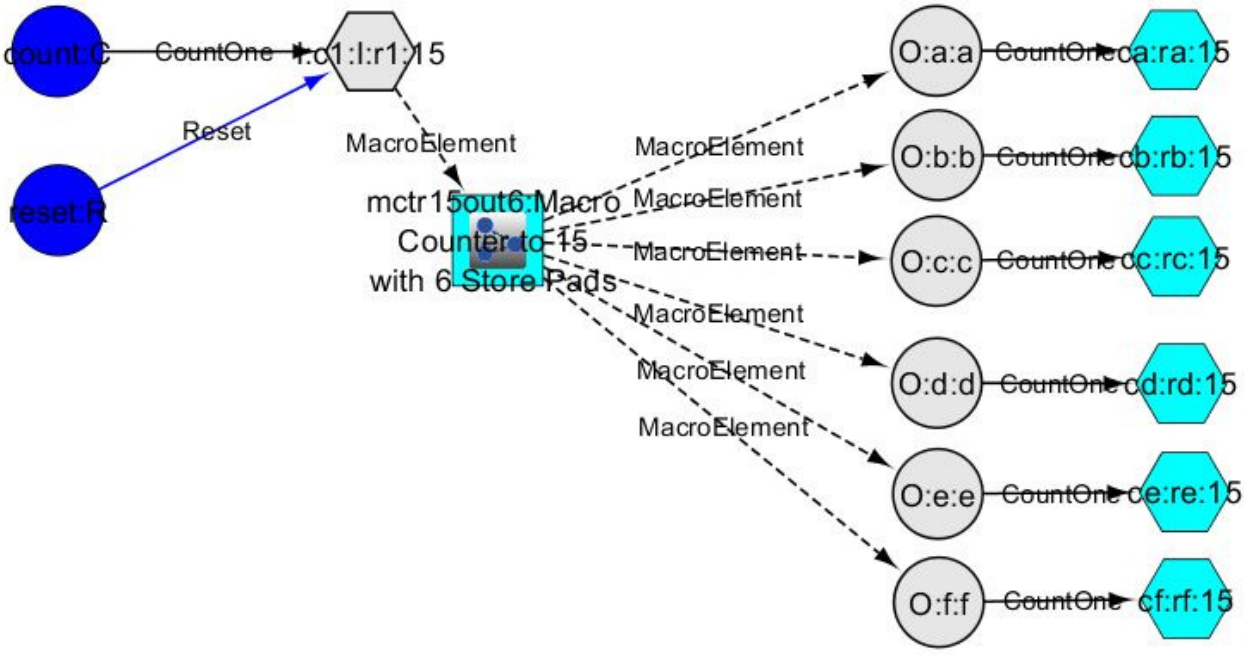
The automaton requires the input stream to signal when it wants to copy the counter value and to provide pumping symbols to move the count value into another designated counter. The number of pumping symbols **must** be equal to the target value of the counter. In the example the symbol 'C' cause the counter to count, 'R' for it to reset and a pumping symbols of 'a', 'b', 'c', 'd', 'e' or 'f' cause the counter value to be pumped into the store counter, with the pumping symbol also indicating which of the six store counters should be used. For example,

RCCCaaaaaaaaaaaaaaRCCCCCbbbbbbbbbbbbbbbRCCCCCCCCCCCfffffffffffffffR

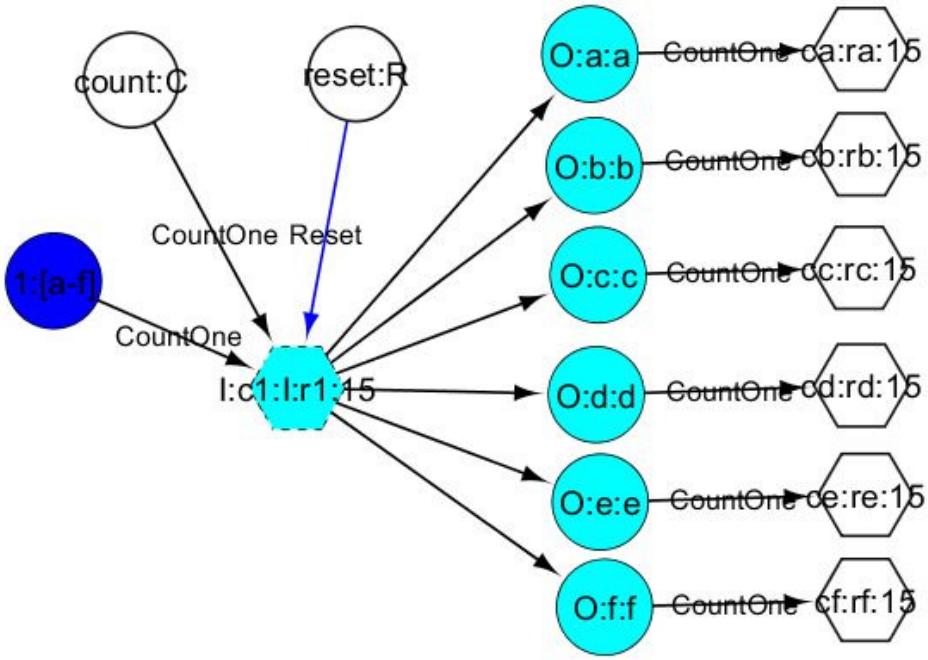
will cause a 3 to be stored in the 'a' counter, a 5 to be stored in the 'b' counter, and a 10 to be stored in the 'f' counter.

ANML-G

The image below shows the ANML-G for the automata network containing one copying counter macro connected to state-transition-elements for counting and resetting and to the store counters. ANML-G for the macro follows this image.



The image below shows the ANML-G for the macro which contains the counter to be copied and the logic for pumping the value out of the counter and into the store counters.



ANML

The counter value is pumped out by the pumping symbol all-input state-transition-element which recognizes the store character symbols a-f. The pumping symbols must be reserved since pumping is started by the recognition of the reserved symbols. The macro requires that all pumping symbols be in the character class a-f. The pumping symbols pump the counter until an output event is triggered and by continuing to pump until the number of symbols pumped equals the target value the latched counter will pulse for the number of symbols equal to its value at the time pumping started. The copied counter remains at the target value since it is latched; it must be reset to be used again. Once the output event has occured at the counter at each symbol cycle a store counter 'pad', that is, a state-transition-element which activates the store counter, will be activated and recognition of the pumping symbol will cause the value in the store counter to increment. The store counter is selected by the pumping symbol - for example, a pumping symbol 'a' is recognized by the pad for store counter 'a' and the count goes there. Another design alternative is to distribute the count drawn out of the main counter by using different pumping symbols in the input. For example, the values stored in the counters could be incremented in a round-robin fashion to "distribute" the copied counter value across a number of counters.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <automata-network name="Copy Counter Bank" id="cc06"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="ANML_v6.xsd">
  - <state-transition-element id="count" symbol-set="C" start="all-input">
      <activate-on-match macro="mctr15out6" element="c1" />
    </state-transition-element>
  - <state-transition-element id="reset" symbol-set="R" start="all-input">
      <activate-on-match macro="mctr15out6" element="r1" />
    </state-transition-element>
  - <macro id="mctr15out6" name="Macro Counter to 15 with 6 Store Pads">
    - <port-definition>
        <element-reference element="1" type="state-transition-element" start="all-
          input" />
        <element-reference element="c1" type="counter" activation="in" />
        <element-reference element="r1" type="counter" activation="in" />
        <element-reference element="a" type="state-transition-element"
          activation="out" />
        <element-reference element="b" type="state-transition-element"
          activation="out" />
        <element-reference element="c" type="state-transition-element"
          activation="out" />
        <element-reference element="d" type="state-transition-element"
          activation="out" />
        <element-reference element="e" type="state-transition-element"
          activation="out" />
        <element-reference element="f" type="state-transition-element"
          activation="out" />
      </port-definition>
    - <state-transition-element id="1" symbol-set="[a-f]" start="all-input">
        <activate-on-match element="c1" />
      </state-transition-element>
    - <state-transition-element id="a" symbol-set="a">
        <activate-on-match macro="cc06" element="ca" />
      </state-transition-element>
    - <state-transition-element id="b" symbol-set="b">
        <activate-on-match macro="cc06" element="cb" />
      </state-transition-element>
    - <state-transition-element id="c" symbol-set="c">
        <activate-on-match macro="cc06" element="cc" />
      </state-transition-element>
    - <state-transition-element id="d" symbol-set="d">
        <activate-on-match macro="cc06" element="cd" />
      </state-transition-element>
    - <state-transition-element id="e" symbol-set="e">
        <activate-on-match macro="cc06" element="ce" />
      </state-transition-element>
    - <state-transition-element id="f" symbol-set="f">
        <activate-on-match macro="cc06" element="cf" />
      </state-transition-element>
    - <counter countone="c1" reset="r1" target="15" at_target="latch">
        <report-on-target />
        <activate-on-target element="a" />
```

```
        <activate-on-target element="b" />
        <activate-on-target element="e" />
        <activate-on-target element="f" />
        <activate-on-target element="c" />
        <activate-on-target element="d" />
      </counter>
    </macro>
    <counter countone="ca" reset="ra" target="15" />
    <counter countone="cb" reset="rb" target="15" />
    <counter countone="cc" reset="rc" target="15" />
    <counter countone="cd" reset="rd" target="15" />
    <counter countone="ce" reset="re" target="15" />
    <counter countone="cf" reset="rf" target="15" />
</automata-network>
```

Reverse Counter

Reverse Counter

The Reverse Counter automaton outputs a pulse each time it counts and stops generating pulses when it reaches the target.

This automaton requires an initial configuration step (priming) - it must be run through one entire count cycle to its target before it will operate correctly. In this example this is accomplished by inputing the symbol 'P' target value number of times into the automaton after start-of-data. Reverse counting is started by a match at the state-transition-element connected to the two inputs of the reverse counter. The pulses from each reverse count drive the activation of state-transition-elements connected to the output of the reverse counter.

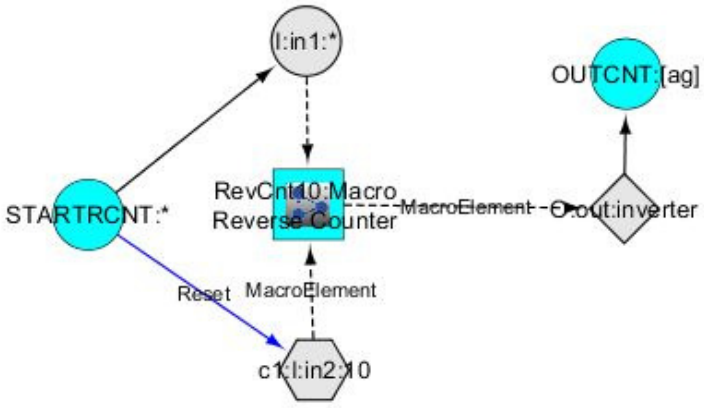
The example implements a reverse counter of 10. With priming a sample input stream and corresponding output could look like this ('*' is any symbol, 'x' is don't care):

```
input stream sod PPPPPPPPPP**************************************** eod
transition                          ^ match at reverse counter input
output          xxxxxxxxxx0000000000001111111110000000000000000000
```

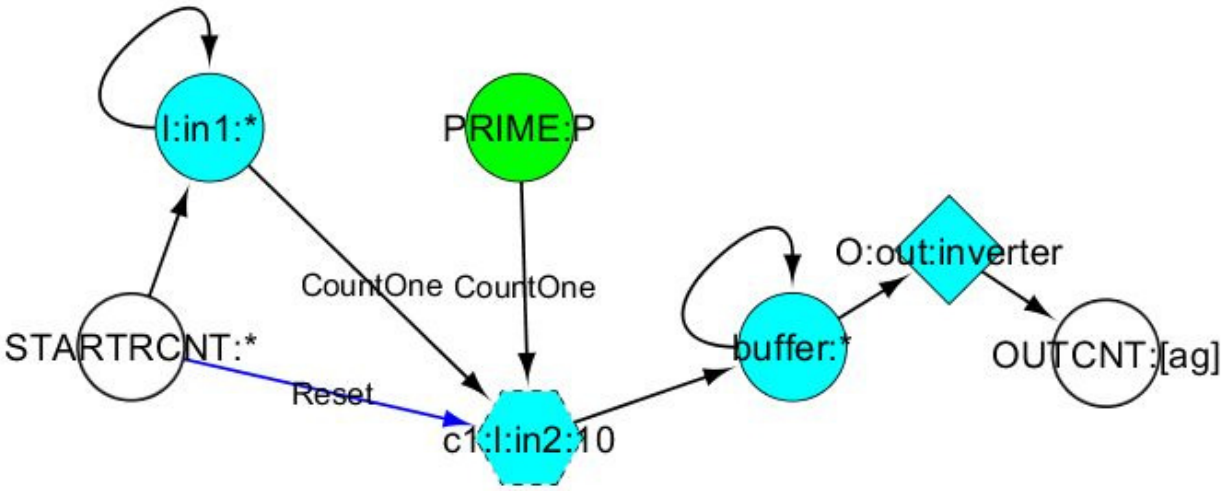
This implementation uses a reverse counter macro containing a counter element. An alternate implementation which uses state-transition-elements instead of a counter element is shown in the next article. The count of the reverse counter which uses a counter element can be modified just by changing the counter's target value while the second version can only be modified by changing the number of state-transition-elements.

ANML-G - Version Using Counter Element

The automata network shows the input and output to and from the reverse counter. The reverse counter itself is contained in a macro, shown in the image which follows the one below.



The image below shows the macro containing the reverse counter, implemented using a counter element.



ANML - Version Using Counter Element

The reverse counter must be primed so the output from the reverse counter is low prior to the reverse count being triggered. If we removed priming the output of the counter would be low and the inverter would cause a high output which would not go away until the counter was triggered. A possible problem during the priming period is that the output of the reverse counter will be high. If it is necessary to hold the reverse counter low during priming the PRIME state-transition-element could activate the buffer state-transition-element and the buffer state-transition-element could also be set to activate on start-of-data so that it pulls the inverter low on the first symbol cycle.

When the counter element is primed to the target it will output a latched high, causing the buffer state-transition-element to continually drive the inverter low. The reverse count is started by activation of the STARTRCNT state-transition-element. This resets the counter element which will now output low until the target is reached, causing a high output through the inverter.

The STARTRCNT state-transition-element also activates the in1 state-transition-element which counts on every symbol cycle. Once the target is reached the counter element again latches high so the inverter will drive a low signal to the output. This cycle may be repeated by another match event on the STARTRCNT state-transition-element.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <automata-network name="Counter Automata Using Reverse Counter" id="Reverse"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="ANML_v6.xsd">
  - <state-transition-element id="STARTRCNT" symbol-set="*">
      <activate-on-match macro="RevCnt10" element="in1" />
      <activate-on-match macro="RevCnt10" element="in2" />
    </state-transition-element>
  - <macro id="RevCnt10" name="Macro Reverse Counter">
    - <port-definition>
        <element-reference element="PRIME" type="state-transition-element"
          start="start-of-data" />
        <element-reference element="in1" type="state-transition-element"
          activation="in" />
        <element-reference element="in2" type="counter" activation="in" />
        <element-reference element="out" type="inverter" activation="out" />
      </port-definition>
    - <state-transition-element id="PRIME" symbol-set="P" start="start-of-data">
        <activate-on-match element="c1" />
      </state-transition-element>
    - <state-transition-element id="in1" symbol-set="*">
        <activate-on-match element="c1" />
        <activate-on-match element="in1" />
      </state-transition-element>
    - <counter countone="c1" reset="in2" target="10" at_target="latch">
        <activate-on-target element="buffer" />
      </counter>
    - <state-transition-element id="buffer" symbol-set="*">
        <activate-on-match element="out" />
        <activate-on-match element="buffer" />
      </state-transition-element>
    - <inverter id="out">
        <activate-on-high macro="Reverse" element="OUTCNT" />
      </inverter>
    </macro>
    <state-transition-element id="OUTCNT" symbol-set="[ag]" />
  </automata-network>
```

Reverse Counter without a Counter Element

Reverse Counter without a Counter Element

The Reverse Counter automaton outputs a pulse each time it counts and stops generating pulses when it reaches the target. This functionality of this automaton is similar to the previous Reverse Counter but because this example does not use a counter element there are some functional differences.

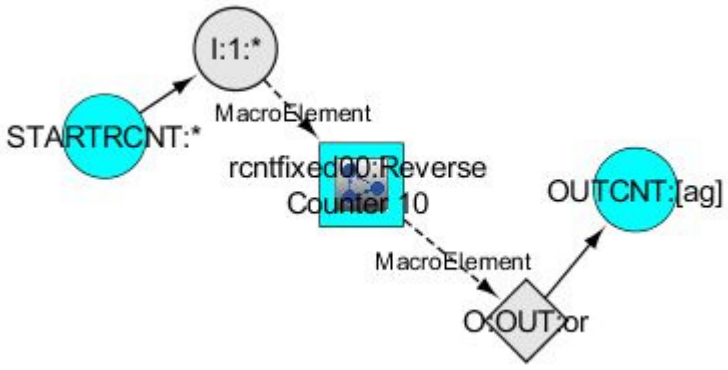
This reverse counter does not require priming to be put in the right state to begin operation and it only has one input signal, to count, to the macro in which the reverse counter is implemented, instead of two in the previous example for reset and for counting. Resetting during the count is not possible with this design but could be done with the previous one.

Without priming, an input stream and output for this circuit could look like this:

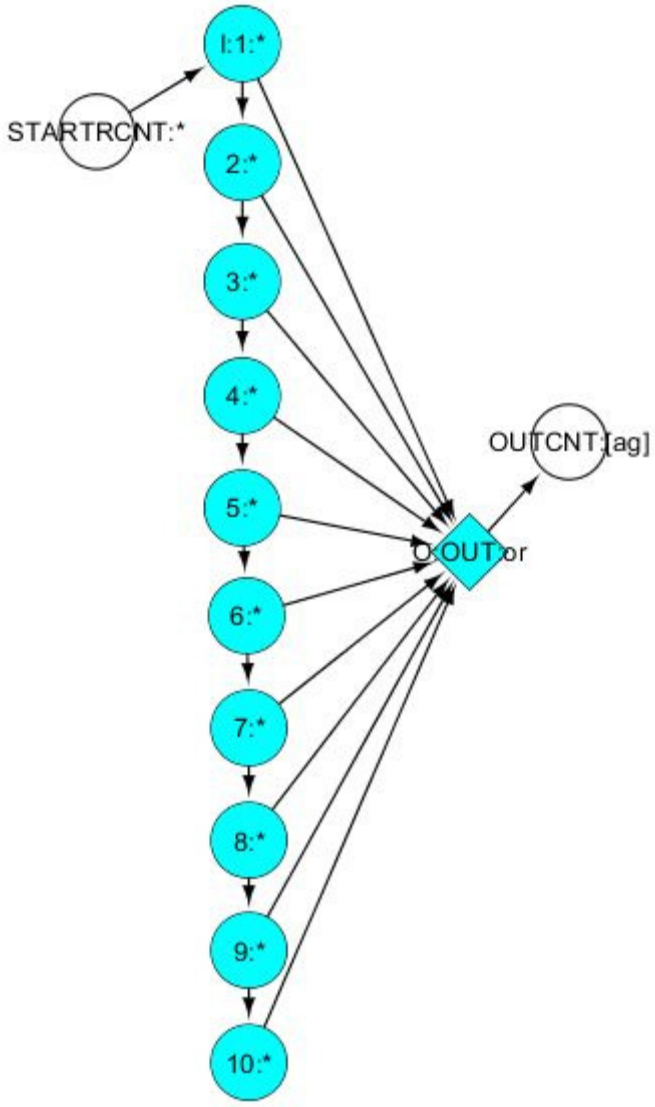
```
input stream sod ************************************** eod
transition                    ^ match at reverse counter input
output          0000000000001111111111000000000000000000
```

ANML-G

The automata network shows the input and output to and from the reverse counter. The reverse counter itself is contained in a macro, shown in the image which follows the one below.



The image below shows the macro containing the reverse counter, implemented with state-transition-elements and an or element.



ANML

The reverse counter macro simply contains a chain of state-transition-elements of length equal to target count value. As each state-transition-elements recognizes a symbol it sends a signal out through the or element. The count is started by an activation to the first state-transition-element of the chain from an external element and the or element can activate external elements for output.

While this design has the advantage of simplicity the macro structure only support a fixed count. The previous reverse counter automaton can support any count simply by changing the target value in counter element. It also will be more resource efficient for large counts.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <automata-network name="Reverse Counter Automaton" id="rcauto01"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="ANML_v6.xsd">
  - <state-transition-element id="STARTRCNT" symbol-set="*">
      <activate-on-match macro="rcntfixed00" element="1" />
    </state-transition-element>
  - <macro id="rcntfixed00" name="Reverse Counter 10">
    - <port-definition>
        <element-reference element="1" type="state-transition-element"
          activation="in" />
        <element-reference element="OUT" type="or" activation="out" />
      </port-definition>
    - <state-transition-element id="1" symbol-set="*">
        <activate-on-match element="OUT" />
        <activate-on-match element="2" />
      </state-transition-element>
    - <state-transition-element id="2" symbol-set="*">
        <activate-on-match element="OUT" />
        <activate-on-match element="3" />
      </state-transition-element>
    - <state-transition-element id="3" symbol-set="*">
        <activate-on-match element="OUT" />
        <activate-on-match element="4" />
      </state-transition-element>
    - <state-transition-element id="4" symbol-set="*">
        <activate-on-match element="OUT" />
        <activate-on-match element="5" />
      </state-transition-element>
    - <state-transition-element id="5" symbol-set="*">
        <activate-on-match element="OUT" />
        <activate-on-match element="6" />
      </state-transition-element>
    - <state-transition-element id="6" symbol-set="*">
        <activate-on-match element="OUT" />
        <activate-on-match element="7" />
      </state-transition-element>
    - <state-transition-element id="7" symbol-set="*">
        <activate-on-match element="OUT" />
        <activate-on-match element="8" />
      </state-transition-element>
    - <state-transition-element id="8" symbol-set="*">
        <activate-on-match element="OUT" />
        <activate-on-match element="9" />
      </state-transition-element>
    - <state-transition-element id="9" symbol-set="*">
        <activate-on-match element="OUT" />
        <activate-on-match element="10" />
      </state-transition-element>
    - <state-transition-element id="10" symbol-set="*">
        <activate-on-match element="OUT" />
      </state-transition-element>
```

```
  – <or id="OUT">
      <activate-on-high macro="rcauto01" element="OUTCNT" />
    </or>
  </macro>
  <state-transition-element id="OUTCNT" symbol-set="[ag]" />
</automata-network>
```

Multipying a Counter by 2

Multipying a Counter by 2

This automaton multiplies a counter by two, equivalent to a binary bit shift. Counters in ANML have configured target values which has the implication that the number of multiply or shift operations is limited by the target value. For example, the maximum number of multipy operations from a counter with a target value of 256 is 8, 1->2, 2->4, 4->8, 8->16, 16->32, 32->64, 64->128, 128->256. The design of this automaton will require the input stream is set up to enable the multiplication operation with two times the target value of the counter in the stream used as pumping symbols. In the case of our 256 counter 512 pumping symbols will be required to do one multiply.

This automaton is not supposed to be a practically useful device, at least not in any scenario we can imagine now, although it perhaps should not be assumed that no practical use of it will ever be encountered. The design does, though, serve the purpose of illustrating ANML design themes and also has a theoretical function is showing the potential computing power of an ANML device as well as its limitations. One theoretically significant application of this automaton is the implementation of a counter-based binary stack. This is described in the following article.

In this example there four control symbols placed in the input stream which enable operation of the multiplying counter. A 'C' causes a normal countone, an 'R' a normal counter reset operation. An 'X' begins a multipy operation. The number of 'X' symbols in the input stream must be equal to the target value of the counter and these must be followed by 'Y' symbols also equal number to the target value of the counter. The multiply operation is not guaranteed to have completed until the last 'Y' is seen. Once the last 'Y' has been seen the counter value that was in the counter at the start will have been multiplied by 2.

Here is an example of the input stream and counter value for an automaton with a counter target value of 8. The counter will have counted to three and after the multiply the value in the counter will be 6.

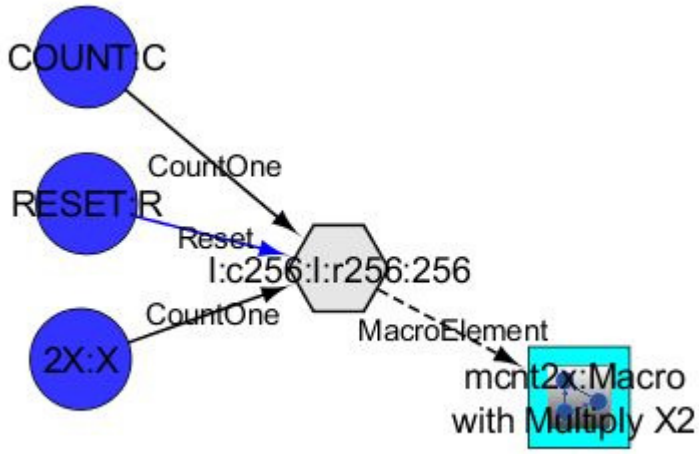
input stream	R	C	C	C	X	X	X	X	X	X	X	X	Y	Y	Y	Y	Y	Y	Y	Y
counter value	0	1	2	3																6

In this example the reader may wonder why if in the input stream we have provided 3 count pulses we do not "know" that we need to provide 3 more to multiply the counter by 2. This is a simplified example; in an actual circuit we may not know how many times the count has been triggered. This automaton provides a general way to multiply by 2 without knowing the counter's count.

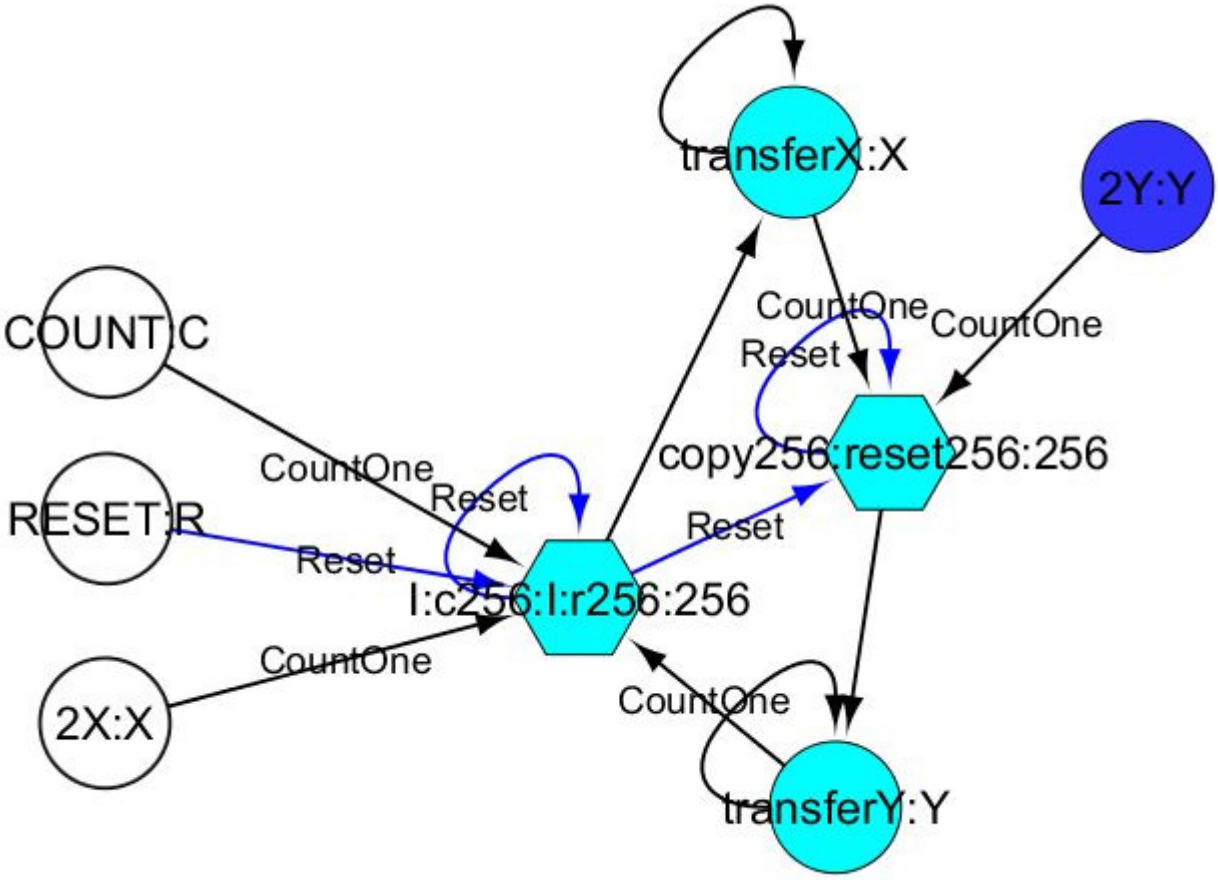
ANML-G

The automata-network is shown below. The state-transition-element that recognize the

control symbols are outside the macro containing the multiplying counter. They are all-input so counting does not have to occur in a continguous stream.



The macro containing the multiplying counter and associated logic is shown below.



ANML

There are two counters used to implement the automaton, the main counter c256 (the example shows a counter with a target value of 256) which has the initial count value and the multiplied value and the "copy" countercopy256 which works something like a copy counter in

the previous copying counters article with the addition that it "feeds" the copied counter value back into the counter it copied from, thereby doubling the value in that counter. The main counter is in *roll* mode so that it returns to the value at which it started after counting off to enable the copy counter. The copy counter is also in *roll* mode and when the multipy sequence is finished it will have the value the main counter started at.

The main counter will be triggered by the state-transition-element 2X target value number of times, causing an activation of state-transition-element transferX at (target value - count) symbol cycles. Since the number of X pumping symbols must be equal to the target value transferX will self-activate and pulse the copy counter count number of times while the *roll* mode main counter will return to count. At this point the count value has been copied to the copy counter and both main and copy counters hold the count value. Now, target value number of Ys are input. The all-input state-transition-element 2Y will continue to pulse the copy counter, causing the state-transition-element transferY to be activated at (target value - count). Self-activating transferY will pulse the main counter count times, at which point the multiply sequence in the input stream will be complete and the main counter will have the value count X 2 and the copy counter will have the value count.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <automata-network name="Counter with Multiply by 2" id="Counter2X"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="ANML_v6.xsd">
  - <state-transition-element id="COUNT" symbol-set="C" start="all-input">
      <activate-on-match macro="mcnt2x" element="c256" />
    </state-transition-element>
  - <state-transition-element id="RESET" symbol-set="R" start="all-input">
      <activate-on-match macro="mcnt2x" element="r256" />
    </state-transition-element>
  - <state-transition-element id="2X" symbol-set="X" start="all-input">
      <activate-on-match macro="mcnt2x" element="c256" />
    </state-transition-element>
  - <macro id="mcnt2x" name="Macro with Multiply X2">
    - <port-definition>
        <element-reference element="c256" type="counter" activation="in" />
        <element-reference element="r256" type="counter" activation="in" />
        <element-reference element="2Y" type="state-transition-element"
          start="all-input" />
      </port-definition>
    - <counter countone="c256" reset="r256" target="256" at_target="roll">
        <activate-on-target element="transferX" />
        <activate-on-target element="reset256" />
      </counter>
    - <state-transition-element id="transferX" symbol-set="X">
        <activate-on-match element="copy256" />
        <activate-on-match element="transferX" />
      </state-transition-element>
    - <state-transition-element id="2Y" symbol-set="Y" start="all-input">
        <activate-on-match element="copy256" />
      </state-transition-element>
    - <counter countone="copy256" reset="reset256" target="256" at_target="roll">
        <activate-on-target element="transferY" />
      </counter>
    - <state-transition-element id="transferY" symbol-set="Y">
        <activate-on-match element="c256" />
        <activate-on-match element="transferY" />
      </state-transition-element>
    </macro>
  </automata-network>
```

Implementing a Bit Stack with the 2X Counter

Implementing a Bit Stack with the 2X Counter

The Counter with multiply by 2 developed in the previous article can be used to implement a binary stack. The counter value, in binary, can be interpreted as a stack of Os and 1s. For example, the counter value of 73, in binary, is 1 0 0 1 0 0 1, or, as stack

1
0
0
1
0
0
1

If we wish to push a zero onto our stack we only have to multiply our counter value by 2. 73x2 = 146, in binary, 1 0 0 1 0 0 1 0, or, interpreted as a stack (the number closest to the bottom of the page is our "top" of the stack.

1
0
0
1
0
0
1
0 <- top of the stack

If we wish to push a one onto our stack we only have to multiply the counter value by 2 and send one more 'C' symbol to add one. For example, 146x2 + 1= 293, in binary, 1 0 0 1 0 0 1 0 1, and interpreted as stack:

1
0
0
1
0
0
1
1 <- top of the stack

Short pattern recognition with mismatches using the Automata Processor as an accelerator

Short pattern recognition with mismatches using the Automata Processor as an accelerator

Problem:

For any input at fixed length *l*, return the patterns with at most *m* mismatches.

Example:

Input length *l=3*, mismatches allowed *m = 1.*

Inputting subject string: aab

Output results should be: aaa, aab, abb, bab

Procedures:

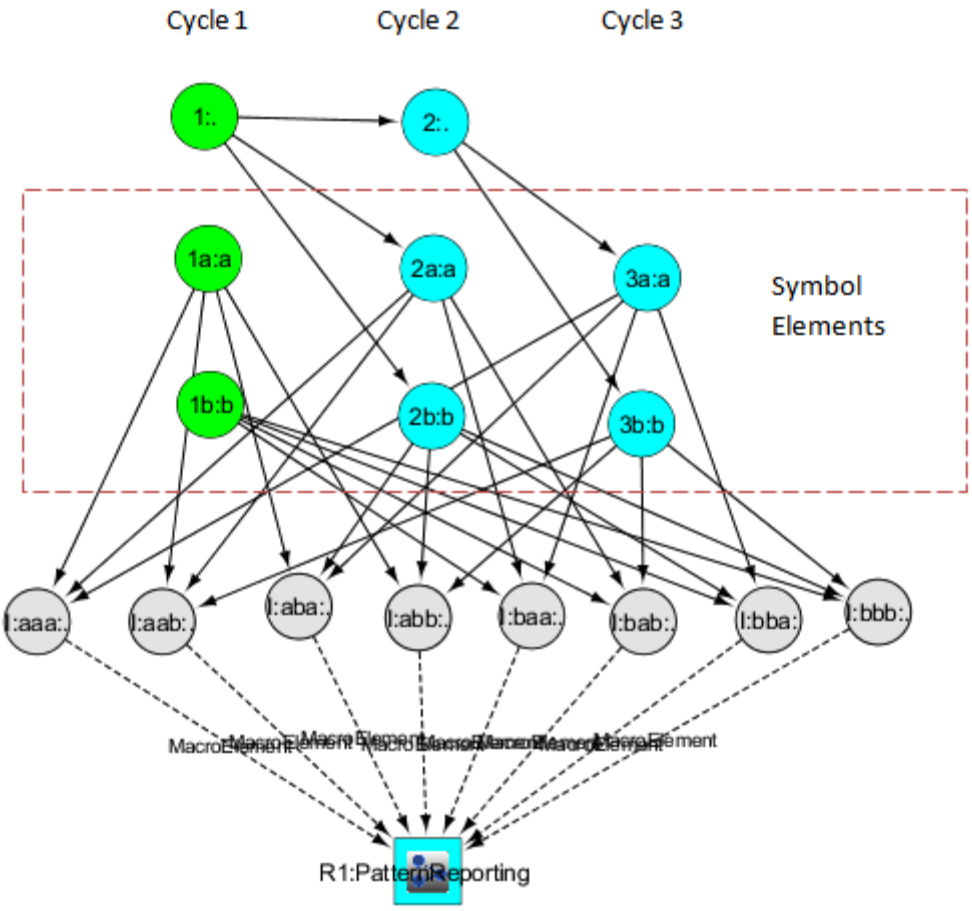
Step one on Lochsa:

Function: for each inputting symbol at position x (x=1, 2, or3), the patterns with a matched symbol at the same position will be reported.

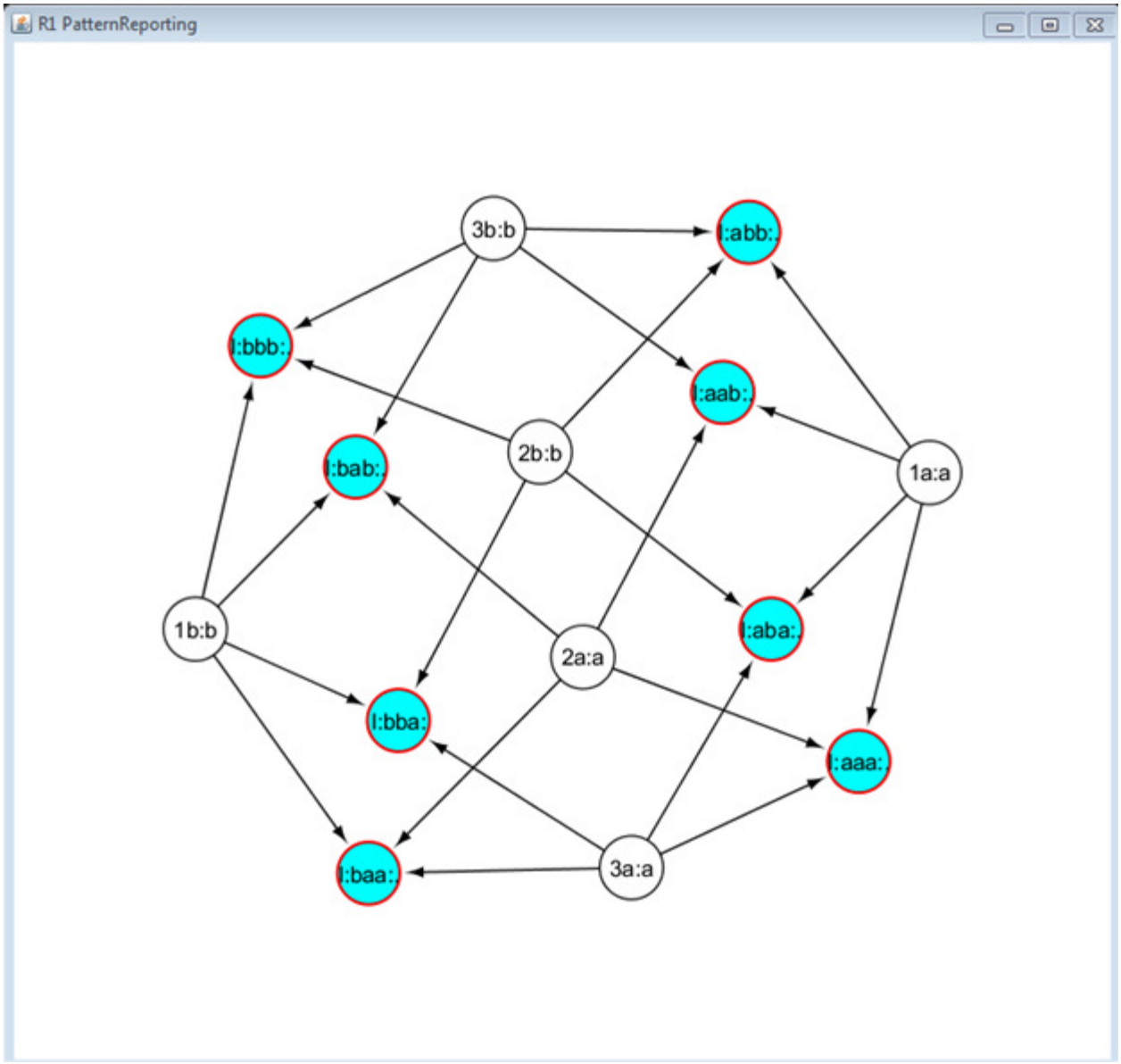
E.g., aab is the inputting string. The output would be as following: (1=match reported, 0=no match reported)

Inputting Cycle	Inputting Simple	aaa	aab	aba	abb	baa	bab	bba	bbb
Cycle 1	a	1	1	1	1	0	0	0	0
Cycle 2	a	1	1	0	0	1	1	0	0
Cycle 3	b	0	1	0	1	0	1	0	1

Automata-network design:


Cycle 1
Cycle 2
Cycle 3
1:.
2:.
1a:a
2a:a
3a:a
Symbol Elements
1b:b
2b:b
3b:b
l:aaa:.
l:aab:.
l:aba:.
l:abb:.
l:baa:.
l:bab:.
l:bba:
l:bbb:.
R1:Pattern Reporting

R1 Macro:



ANML-G

ANML

CPU post-processing:

Take the output of automata processor in the format of binary matching vectors, CPU calculate the sum of every pattern.

E.g. for the output of inputting string aab:

	Inputting Simple	aaa	aab	aba	abb	baa	bab	bba	bbb
Cycle 1	a	1	1	1	1	0	0	0	0

Cycle 2	a	1	1	0	0	1	1	0	0
Cycle 3	b	0	1	0	1	0	1	0	1
Sum		2	3	1	2	1	2	0	1

CPU calculates the sum of each column, than identify patterns according to the matching criterion.

For no mismatches, sum should be equal to the length of the inputting string, 3 in this case, only exact match “aab” fit this criterion.

If allow 1 mismatch, we look for patterns with sum>=2, which are “aaa”, “aab”, “abb”, “bab”

Discussion:

Time efficiency

This solution can makes use of CPU and automata processor in parallel and as long as the processing time on CPU for each matching output from automata processor is less than a symbol cycle of automata processor, the post-processing time on CPU can be hidden. Otherwise, the total processing time would be restricted by the speed of CPU.

Space efficiency

Number of elements needed for this type of machine

$N = a^l + (a+1) * l - 1$

Where a is the size of the alphabet, and l is the fixed length of the inputting string.

Compare to a tree structure to find exact matches of fixed length string, where $N = a^l + a^{l-1} + \ldots + a$, while $a>2$ and $l>2$, this proposed method would be more space efficient and has the flexibility to set any number of mismatches to be allowed without extra time and space costs.

In/Out degree

The in degree of each reporting element is l and the out degree of each symbol element is a^{l-1}.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <automata-network name="ShortPatternSearch" id="S1">
  - <state-transition-element id="1" symbol-set="." start="start-of-data">
      <activate-on-match element="2" />
      <activate-on-match element="2a" />
      <activate-on-match element="2b" />
    </state-transition-element>
  - <state-transition-element id="1a" symbol-set="a" start="start-of-data">
      <activate-on-match macro="R1" element="aaa" />
      <activate-on-match macro="R1" element="aab" />
      <activate-on-match macro="R1" element="aba" />
      <activate-on-match macro="R1" element="abb" />
    </state-transition-element>
  - <state-transition-element id="1b" symbol-set="b" start="start-of-data">
      <activate-on-match macro="R1" element="bab" />
      <activate-on-match macro="R1" element="bba" />
      <activate-on-match macro="R1" element="bbb" />
      <activate-on-match macro="R1" element="baa" />
    </state-transition-element>
  - <state-transition-element id="2" symbol-set=".">
      <activate-on-match element="3a" />
      <activate-on-match element="3b" />
    </state-transition-element>
  - <state-transition-element id="2a" symbol-set="a">
      <activate-on-match macro="R1" element="bab" />
      <activate-on-match macro="R1" element="aaa" />
      <activate-on-match macro="R1" element="aab" />
      <activate-on-match macro="R1" element="baa" />
    </state-transition-element>
  - <state-transition-element id="2b" symbol-set="b">
      <activate-on-match macro="R1" element="bba" />
      <activate-on-match macro="R1" element="bbb" />
      <activate-on-match macro="R1" element="aba" />
      <activate-on-match macro="R1" element="abb" />
    </state-transition-element>
  - <state-transition-element id="3a" symbol-set="a">
      <activate-on-match macro="R1" element="bba" />
      <activate-on-match macro="R1" element="aaa" />
      <activate-on-match macro="R1" element="aba" />
      <activate-on-match macro="R1" element="baa" />
    </state-transition-element>
  - <state-transition-element id="3b" symbol-set="b">
      <activate-on-match macro="R1" element="bab" />
      <activate-on-match macro="R1" element="bbb" />
      <activate-on-match macro="R1" element="aab" />
      <activate-on-match macro="R1" element="abb" />
    </state-transition-element>
  - <macro id="R1" name="PatternReporting">
    - <port-definition>
        <element-reference element="aaa" type="state-transition-element"
          activation="in" reporting="true" />
        <element-reference element="aab" type="state-transition-element"
```

```
    activation="in" reporting="true" />
      <element-reference element="aba" type="state-transition-element"
        activation="in" reporting="true" />
      <element-reference element="abb" type="state-transition-element"
        activation="in" reporting="true" />
      <element-reference element="baa" type="state-transition-element"
        activation="in" reporting="true" />
      <element-reference element="bab" type="state-transition-element"
        activation="in" reporting="true" />
      <element-reference element="bba" type="state-transition-element"
        activation="in" reporting="true" />
      <element-reference element="bbb" type="state-transition-element"
        activation="in" reporting="true" />
    </port-definition>
  - <state-transition-element id="aaa" symbol-set=".">
      <report-on-match />
    </state-transition-element>
  - <state-transition-element id="aab" symbol-set=".">
      <report-on-match />
    </state-transition-element>
  - <state-transition-element id="aba" symbol-set=".">
      <report-on-match />
    </state-transition-element>
  - <state-transition-element id="abb" symbol-set=".">
      <report-on-match />
    </state-transition-element>
  - <state-transition-element id="baa" symbol-set=".">
      <report-on-match />
    </state-transition-element>
  - <state-transition-element id="bab" symbol-set=".">
      <report-on-match />
    </state-transition-element>
  - <state-transition-element id="bba" symbol-set="">
      <report-on-match />
    </state-transition-element>
  - <state-transition-element id="bbb" symbol-set=".">
      <report-on-match />
    </state-transition-element>
  </macro>
</automata-network>
```

Fixed-length pattern recognition with serial-to-parallel shift-register match aggregation

Fixed-length Pattern Recognition with Serial-to-Parallel Shift-Register Match Aggregation

This ANML machine illustrates a "serial-to-parallel shift-register" technique for aggregating matches. Aggregating matches may improve performance if the implementation of the automata processor takes more time to transfer a single match occuring on each of *n* symbol cycles than *n* matches occuring on a single symbol cycle.

The example automaton tests each set of three symbols against some number of three-symbol patterns and aggregates results so that they can reported on a single cycle on an end-of-data signal. The depth of the aggregation structure is fixed - in the example illustrated here the results of 8 sets of three symbols can be aggregated with a shift-register structure 7 deep (7 results are "stored" in the shift-register and the 8th comes directly off the pattern match). Once 24 symbols are seen an EOD must be issued and the results read out to avoid shifting out results.

In the example automaton we are testing sets of three symbols against the patterns 'abc', 'tac', and 'add'. We need to know the position of each match in the input but we do not need to know which of the three patterns matched. We run through the first 23 symbols, aggregating matches and on the 24th symbol, terminated with an EOD, we output the match status of each of the 8 three-symbol positions in the input (first three symbols is position 0, second three symbols is position 1 and so on).

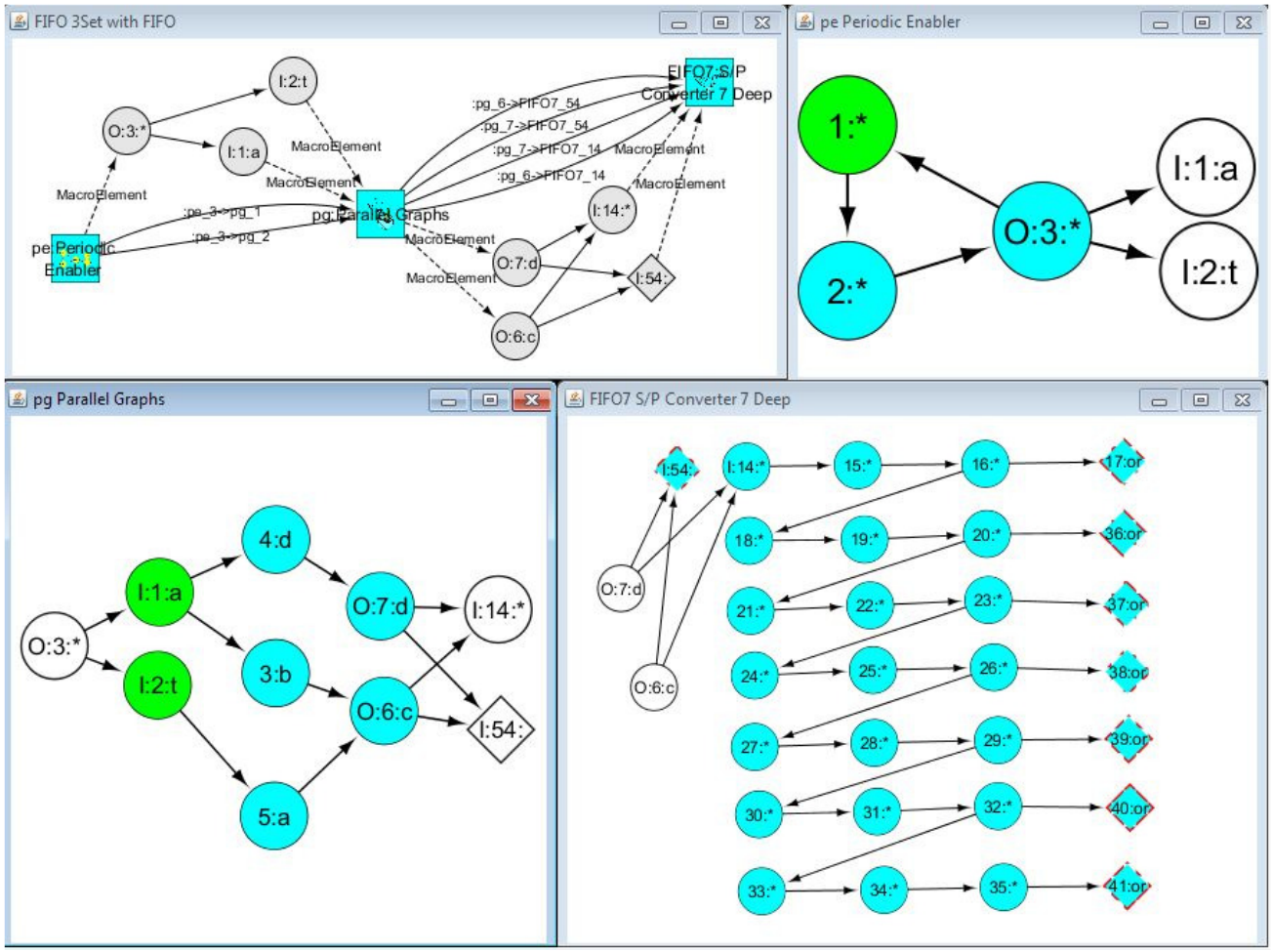
An input sequence and the resulting match results are shown below. The 24-symbol input sequence contains 4 3-symbol matches

input: xxxabctacxxxxxxaddxxxadd[EOD]

```
match results:                      |pos 0: 0
                                    |pos 1: 1
                                    |pos 2: 1
                                    |pos 3: 0
                                    |pos 4: 0
                                    |pos 5: 1
                                    |pos 6: 0
                                    |pos 7: 1
```

ANML-G

The automaton is implemented using three interconnected macros, the Periodic Enabler which causes the input to match in sets of three symbols, Parallel Graphs which tests the sets of three symbols against patterns, and the S/P Converter 7 Deep which aggregates match results and outputs them on the EOD signal.



ANML

This design can be modified to accommodate different sizes of symbol sets and different depths of the Serial-to-Parallel shift-register to aggregate more or less match results although the design does require a fixed size for symbol sets and a fixed shift-register depth. The Periodic Enabler macro simply "counts" every three symbols, reactivating every start position in the Parallel Graphs macro at this interval to cause evaluation of discrete sets. State-transition-element 1 is start-of-data activated for the first symbol and is reactivated at each interval by the preceding state-transition-element in the cycle. Adding state-transition-elements to the cycle will increase the interval length. Parallel graphs contain the patterns we are testing for. Each initial state-transition-element implementing the patterns is start-of-data enabled so that the first set of symbols in processed as the Periodic Enabler only activates these state-transition-elements at the beginning of the second cycle. The "parallel graphs" all activate the input to the shift-register and a reporting element in the shift-register, with the implication that we cannot determine which path was taken in a match since all path are ORed together to those two elements in the shift-register. The implementation of Parallel Graphs in the example compresses the graph, using one state-transition-element for the 'a' in the first position of two patterns and one state-transition-element for the final 'c' in two patterns. S/P Converter 7 Deep "shifts" the results of Parallel Graphs into a line of singly-connected state-transition-elements. If the final state-transition-element in Parallel Graphs is activated and matches a matching pattern of 3 symbols has been detected. These state-transition-elements activate state-transition-element 14 in S/P Converter 7 Deep and because every subsequent state-transition-element matches any symbol that match will propagate on every symbol cycle. If the final state-transition-element in Parallel Graphs is not activated or is activated but doesn't match state-transition-element 14 in S/P Converter 7 Deep is not activated and for that set of three symbols no match value is propagated in the chain. As the input is processed in sets of three the match results will be propagated to the last state-transition-element in each row of the shift-register. At an EOD occuring on the third symbol of the last set the match status of each of these state-transition-elements will be propagated to the EOD-enabled OR element connected to it and it is from these reporting elements that 3-set position of each match can be determined. OR element 54 performs the same function for the last 3-set of symbols "still in" Parallel Graphs, making the automaton capable of aggregrating matches for up to 8 3-sets of symbols.

The "width" of the shift-register can be modified by adding or removing state-transition-elements in the rows, changing the size of the input symbol sets it handles and the depth can be modified by adding more rows.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <automata-network name="3Set with FIFO" id="FIFO"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="..\..\ANML\cookbook\ANML_v6.xsd"> <!--  -->

  - <macro id="pe" name="Periodic Enabler">
      <!-- (...)*  -->
    - <port-definition>
        <element-reference element="1" type="state-transition-element"
          start="start-of-data" />
        <element-reference element="3" type="state-transition-element"
          activation="out" />
      </port-definition>
    - <state-transition-element id="1" symbol-set="*" start="start-of-data">
        <activate-on-match element="2" />
      </state-transition-element>
    - <state-transition-element id="2" symbol-set="*">
        <activate-on-match element="3" />
      </state-transition-element>
    - <state-transition-element id="3" symbol-set="*">
        <activate-on-match element="1" />
        <activate-on-match macro="pg" element="1" />
        <activate-on-match macro="pg" element="2" />
      </state-transition-element>
    </macro>
  - <macro id="pg" name="Parallel Graphs">
      <!-- (abc|tac|add)  -->
    - <port-definition>
        <element-reference element="1" type="state-transition-element"
          activation="in" start="start-of-data" />
        <element-reference element="2" type="state-transition-element"
          activation="in" start="start-of-data" />
        <element-reference element="6" type="state-transition-element"
          activation="out" />
        <element-reference element="7" type="state-transition-element"
          activation="out" />
      </port-definition>
    - <state-transition-element id="1" symbol-set="a" start="start-of-data">
        <activate-on-match element="3" />
        <activate-on-match element="4" />
      </state-transition-element>
    - <state-transition-element id="2" symbol-set="t" start="start-of-data">
        <activate-on-match element="5" />
      </state-transition-element>
    - <state-transition-element id="3" symbol-set="b">
        <activate-on-match element="6" />
      </state-transition-element>
    - <state-transition-element id="4" symbol-set="d">
        <activate-on-match element="7" />
      </state-transition-element>
    - <state-transition-element id="5" symbol-set="a">
        <activate-on-match element="6" />
      </state-transition-element>
```

```
  - <state-transition-element id="6" symbol-set="c">
      <activate-on-match macro="FIFO7" element="54" />
      <activate-on-match macro="FIFO7" element="14" />
    </state-transition-element>
  - <state-transition-element id="7" symbol-set="d">
      <activate-on-match macro="FIFO7" element="54" />
      <activate-on-match macro="FIFO7" element="14" />
    </state-transition-element>
  </macro>
- <macro id="FIFO7" name="S/P Converter 7 Deep">
    <!--  (...){0,7}  -->
  - <port-definition>
      <element-reference element="54" type="or" activation="in"
        reporting="true" />
      <element-reference element="14" type="state-transition-element"
        activation="in" />
    </port-definition>
  - <or id="54" high-only-on-eod="true">
      <report-on-high />
    </or>
  - <state-transition-element id="14" symbol-set="*">
      <activate-on-match element="15" />
    </state-transition-element>
  - <state-transition-element id="15" symbol-set="*">
      <activate-on-match element="16" />
    </state-transition-element>
  - <state-transition-element id="16" symbol-set="*">
      <activate-on-match element="17" />
      <activate-on-match element="18" />
    </state-transition-element>
  - <state-transition-element id="18" symbol-set="*">
      <activate-on-match element="19" />
    </state-transition-element>
  - <state-transition-element id="19" symbol-set="*">
      <activate-on-match element="20" />
    </state-transition-element>
  - <state-transition-element id="20" symbol-set="*">
      <activate-on-match element="21" />
      <activate-on-match element="36" />
    </state-transition-element>
  - <state-transition-element id="21" symbol-set="*">
      <activate-on-match element="22" />
    </state-transition-element>
  - <state-transition-element id="22" symbol-set="*">
      <activate-on-match element="23" />
    </state-transition-element>
  - <state-transition-element id="23" symbol-set="*">
      <activate-on-match element="24" />
      <activate-on-match element="37" />
    </state-transition-element>
  - <state-transition-element id="24" symbol-set="*">
      <activate-on-match element="25" />
    </state-transition-element>
```

```
– <state-transition-element id="25" symbol-set="*">
    <activate-on-match element="26" />
  </state-transition-element>
– <state-transition-element id="26" symbol-set="*">
    <activate-on-match element="27" />
    <activate-on-match element="38" />
  </state-transition-element>
– <state-transition-element id="27" symbol-set="*">
    <activate-on-match element="28" />
  </state-transition-element>
– <state-transition-element id="28" symbol-set="*">
    <activate-on-match element="29" />
  </state-transition-element>
– <state-transition-element id="29" symbol-set="*">
    <activate-on-match element="39" />
    <activate-on-match element="30" />
  </state-transition-element>
– <state-transition-element id="30" symbol-set="*">
    <activate-on-match element="31" />
  </state-transition-element>
– <state-transition-element id="31" symbol-set="*">
    <activate-on-match element="32" />
  </state-transition-element>
– <state-transition-element id="32" symbol-set="*">
    <activate-on-match element="40" />
    <activate-on-match element="33" />
  </state-transition-element>
– <state-transition-element id="33" symbol-set="*">
    <activate-on-match element="34" />
  </state-transition-element>
– <state-transition-element id="34" symbol-set="*">
    <activate-on-match element="35" />
  </state-transition-element>
– <state-transition-element id="35" symbol-set="*">
    <activate-on-match element="41" />
  </state-transition-element>
– <or id="17" high-only-on-eod="true">
    <report-on-high />
  </or>
– <or id="36" high-only-on-eod="true">
    <report-on-high />
  </or>
– <or id="37" high-only-on-eod="true">
    <report-on-high />
  </or>
– <or id="38" high-only-on-eod="true">
    <report-on-high />
  </or>
– <or id="39" high-only-on-eod="true">
    <report-on-high />
  </or>
– <or id="40" high-only-on-eod="true">
```

```
      <report-on-high />
    </or>
  – <or id="41" high-only-on-eod="true">
      <report-on-high />
    </or>
  </macro>
</automata-network>
```

Appendix 1 Easy XML

Easy XML

ANML is a language defined using eXtensible Markup Language, XML. XML is an international standard maintained by the W3C, used widely in the internet and web technologies and has been employed to define thousands of languages across an extremely diverse sets of fields. You can read the actual core standard here and find a nice general introduction here. The remainder of this article gives the few details of XML necessary to understand this user guide.

The basic clause of XML is the *element*, an element is a thing, action, or object, anything that you would talk about as an entity in the system you are building. In ANML the most important element is, of course, one of the many types of automaton elements. (Unfortunately, ANML and XML boths use the term *element* but with different specific meanings. In this guide we mainly rely on the context to signal to the reader which is which but where they may be confusion "XML element" or "automaton element" will be used to make it clear which is meant.) An element starts with a start tag, beginning with an angle bracket followed by the name of element

<state-transition-element

followed by space-separated *attribute-value* pairs which provide more details about the XML element

<state-transition-element id="a12"

saying that this state-transition-element thing has an id of a12. After the attribute-value pairs the start tag is terminated either by a > or a /> depending on whether the element has or does not have *content*. Content are elements nested (children) in the element (parent) or text. ANML has only one element with text content, the <regex> element in macros.
The state-transition-element element has children so after the start tag we'll insert the children elements, followed by state-transition-element's end tag, the element name preceded by </ and followed by >.

<state-transition-element id="a12"> ... children elements ... </state-transition-element>

One may thing of an element with children as a container that "holds" its children elements, although sometimes it literally is a larger thing containing smaller things and sometimes the children are more like actions or concepts related to the parent element. The former situation is the case when we have automaton elements in cells

```
<macro ... >
  <state-transition-element ...> ... </state-transition-element>
  <state-transition-element ...> ... </state-transition-element>
</macro>
```

(we can indent elements to make the nesting clearer) and the latter situation more the case with the children of state-transition-element elements

```
<state-transition-element ...>
  <activate-on-match element="a12"/>
  <activate-on-match element="a44"/>
</state-transition-element>
```

The activate-on-match child elements (whiich happen to be of the element type that doesn't have content) of the state-transition-element are not really things inside the state-transition-element but are more like actions associated with the parent state-transition-element element.

One way in XML to express relationships between elements is through the parent-child hierarchy just presented. Sometimes, however, elements are connected in some way but don't have a parent-child hierarchical relationship. There are a number of ways to express these relationships in XML; ANML uses a very simple mechanism of linking through *id*s. Every "thing" in ANML has some type of id and anyother thing that wants to connect to it uses uses its id to express the relationship. In the last example a connection to two state-transition-element elements is expressed, the state-transition-element elements having the ids a12 and a44. Because activate-on-match is an action and not a thing, the connections are actually between the parent state-transition-element element and the state-transition-element elements indicated by the ids and activate-on-match, as an action, expresses the type of connected effected between the state-transition-element elements.

In XML one can express how many of each type of element is required in various contexts. For example, an optional element would have an occurence of zero or one, an element for which at least one is required would have an occurence of one or more.

XML does have datatypes. Most values in ANML are simply strings but datatypes are used to create different id types for each automaton element type but the ANML schema does make use of enumerations and booleans.

Appendix 2 ANML XML Schema

ANML XML Schema

Here is the hierarchical view of ANML elements. This structure is reflected in the schema but may be easier to grasp the basic structure from the outline format below.

automata-network

  logic-region
    macro
    state-transition-element, counter, inverter, and, nand, or, nor, sum-of-products, product-of-sums
  macro
    port-definition
      element-reference
    macro
     regex
    state-transition-element, counter, inverter, and, nand, or, nor, sum-of-products, product-of-sums
  state-transition-element, counter, inverter, and, nand, or, nor, sum-of-products, product-of-sums

state-transition-element
 report-on-match
 activate-on-match
counter
  report-on-target
  activate-on-target
inverter, and, nand, or, nor
  report-on-high
  activate-on-high
sum-of-products
  product-term
  report-on-high
  activate-on-high
product-of-sums
  sum-term
  report-on-high
  activate-on-high

```
<?xml version="1.0" encoding="UTF-8" ?>
<!--
    04/18/2012 Draft version v6
        Removed OR2 element.
            Removed logic-region element. In its place added boolean power-region attribute to macro.
            Modified attributes of port-definition to allow specification that an element-reference to be
            of an combination of activating, reporting, and start. Also added attribute to document the type of
            element referenced by the element-reference (i.e., state-transition-element, counter, combinatorial).
-->
<!--
    04/05/2012 Draft version v5
        Corrected the omission of macro and regex elements permitted inside of the macro element.
            Added the attribute case-insensitive to state-transition-elements. It is a boolean type with a
            default value of false. It can affect the interpretation of symbol-set characters, character-classes,
            and multiple values.
-->
<!--
    03/12/2012 Draft version v4
        Changed element-reference port-definition subelement to maxoccurs="unbounded"
            as content model was only allowing a single element-reference
-->
<!--
    03/05/2012 Draft version v3
        Changed macro port-definition subelement to minoccurs="0" since macro may have no in or outport elements
-->
<!-- 02/22/2012 Draft version v2 -->
<xs:schema version="draft-6" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:simpleType name="AT_type">
    <xs:restriction base="xs:string">
      <xs:enumeration value="state-transition-element" />
      <xs:enumeration value="counter" />
      <xs:enumeration value="or" />
      <xs:enumeration value="and" />
      <xs:enumeration value="nand" />
      <xs:enumeration value="nor" />
      <xs:enumeration value="sum-of-products" />
      <xs:enumeration value="product-of-sums" />
      <xs:enumeration value="inverter" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="AT_activation">
    <xs:restriction base="xs:string">
      <xs:enumeration value="in" />
      <xs:enumeration value="out" />
      <xs:enumeration value="inout" />
      <xs:enumeration value="none" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="AT_start">
    <xs:restriction base="xs:string">
      <xs:enumeration value="none" />
      <xs:enumeration value="start-of-data" />
      <xs:enumeration value="all-input" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="AT_countermodes">
    <xs:restriction base="xs:string">
      <xs:enumeration value="roll" />
      <xs:enumeration value="pulse" />
      <xs:enumeration value="latch" />
    </xs:restriction>
  </xs:simpleType>
  <xs:complexType name="T_report-on-match" />
  <xs:complexType name="T_activate-on-match">
    <xs:attribute ref="macro" use="optional" default="self" />
    <xs:attribute ref="element" use="required" />
  </xs:complexType>
  <xs:complexType name="T_state-transition-element">
    <xs:sequence minOccurs="0">
      <xs:element ref="report-on-match" minOccurs="0" />
      <xs:element ref="activate-on-match" minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
    <xs:attribute name="id" type="xs:string" use="required" />
    <xs:attribute name="symbol-set" type="xs:string" use="required" />
    <xs:attribute ref="start" default="none" />
    <xs:attribute ref="latch" default="false" />
```

```
    <xs:attribute ref="case-insensitive" default="false" />
  </xs:complexType>
  <xs:complexType name="T_report-on-target" />
- <xs:complexType name="T_activate-on-target">
    <xs:attribute ref="macro" use="optional" default="self" />
    <xs:attribute ref="element" use="required" />
  </xs:complexType>
- <xs:complexType name="T_counter">
  - <xs:sequence>
      <xs:element ref="report-on-target" minOccurs="0" maxOccurs="1" />
      <xs:element ref="activate-on-target" minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
    <xs:attribute ref="countone" use="required" />
    <xs:attribute ref="reset" use="required" />
    <xs:attribute ref="target" use="required" />
    <xs:attribute ref="at_target" default="pulse" />
  </xs:complexType>
- <xs:complexType name="T_activate-on-high">
    <xs:attribute ref="macro" use="optional" default="self" />
    <xs:attribute ref="element" use="required" />
  </xs:complexType>
  <xs:complexType name="T_report-on-high" />
- <xs:complexType name="T_or_and">
  - <xs:sequence>
      <xs:element ref="report-on-high" minOccurs="0" maxOccurs="1" />
      <xs:element ref="activate-on-high" minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
    <xs:attribute name="id" type="xs:string" use="required" />
    <xs:attribute ref="high-only-on-eod" default="false" />
    <xs:attribute ref="negate-inputs" default="false" />
  </xs:complexType>
- <xs:complexType name="T_nor_nand_inverter">
  - <xs:sequence>
      <xs:element ref="report-on-high" minOccurs="0" maxOccurs="1" />
      <xs:element ref="activate-on-high" minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
    <xs:attribute name="id" type="xs:string" use="required" />
    <xs:attribute ref="high-only-on-eod" default="false" />
  </xs:complexType>
- <xs:complexType name="T_product-term">
    <xs:attribute name="id" type="xs:string" use="required" />
  </xs:complexType>
- <xs:complexType name="T_sum-of-products">
  - <xs:sequence>
      <xs:element ref="product-term" maxOccurs="unbounded" />
      <xs:element ref="report-on-high" minOccurs="0" maxOccurs="1" />
      <xs:element ref="activate-on-high" minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
    <xs:attribute name="id" type="xs:string" use="required" />
    <xs:attribute ref="high-only-on-eod" default="false" />
  </xs:complexType>
- <xs:complexType name="T_sum-term">
    <xs:attribute name="id" type="xs:string" use="required" />
  </xs:complexType>
- <xs:complexType name="T_product-of-sums">
  - <xs:sequence>
      <xs:element ref="sum-term" maxOccurs="unbounded" />
      <xs:element ref="report-on-high" minOccurs="0" maxOccurs="1" />
      <xs:element ref="activate-on-high" minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
    <xs:attribute name="id" type="xs:string" use="required" />
    <xs:attribute ref="high-only-on-eod" default="false" />
  </xs:complexType>
- <xs:complexType name="T_regex">
  - <xs:simpleContent>
    - <xs:extension base="xs:string">
        <xs:attribute name="id" type="xs:string" use="required" />
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
- <xs:complexType name="T_element-reference">
    <xs:attribute ref="macro" use="optional" />
    <xs:attribute ref="element" use="required" />
    <xs:attribute ref="type" use="required" />
```

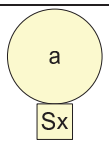
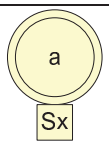
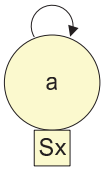
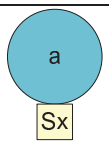
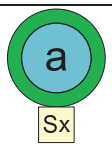
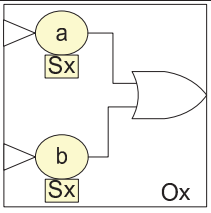
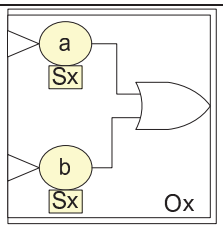
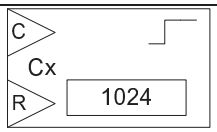
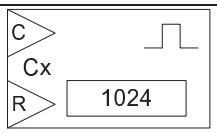
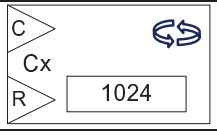
```
    <xs:attribute ref="activation" use="optional" />
    <xs:attribute ref="reporting" use="optional" />
    <xs:attribute ref="start" use="optional" />
  </xs:complexType>
- <xs:complexType name="T_port-definition">
  - <xs:sequence>
      <xs:element maxOccurs="unbounded" ref="element-reference" />
    </xs:sequence>
  </xs:complexType>
- <xs:complexType name="T_macro">
  - <xs:sequence maxOccurs="1">
      <xs:element minOccurs="0" ref="port-definition" />
    - <xs:choice maxOccurs="unbounded">
        <xs:element ref="macro" minOccurs="0" maxOccurs="unbounded" />
        <xs:element ref="regex" minOccurs="0" maxOccurs="unbounded" />
        <xs:element ref="state-transition-element" minOccurs="0" maxOccurs="unbounded" />
        <xs:element ref="counter" minOccurs="0" maxOccurs="unbounded" />
        <xs:element ref="inverter" minOccurs="0" maxOccurs="unbounded" />
        <xs:element ref="and" minOccurs="0" maxOccurs="unbounded" />
        <xs:element ref="nand" minOccurs="0" maxOccurs="unbounded" />
        <xs:element ref="or" minOccurs="0" maxOccurs="unbounded" />
        <xs:element ref="nor" minOccurs="0" maxOccurs="unbounded" />
        <xs:element ref="sum-of-products" minOccurs="0" maxOccurs="unbounded" />
        <xs:element ref="product-of-sums" minOccurs="0" maxOccurs="unbounded" />
      </xs:choice>
    </xs:sequence>
    <xs:attribute name="id" type="xs:string" use="required" />
    <xs:attribute name="name" type="xs:string" use="required" />
    <xs:attribute ref="power-region" />
  </xs:complexType>
- <xs:complexType name="T_automata-network">
  - <xs:choice maxOccurs="unbounded">
      <xs:element ref="macro" minOccurs="0" maxOccurs="unbounded" />
      <xs:element ref="state-transition-element" minOccurs="0" maxOccurs="unbounded" />
      <xs:element ref="counter" minOccurs="0" maxOccurs="unbounded" />
      <xs:element ref="inverter" minOccurs="0" maxOccurs="unbounded" />
      <xs:element ref="and" minOccurs="0" maxOccurs="unbounded" />
      <xs:element ref="nand" minOccurs="0" maxOccurs="unbounded" />
      <xs:element ref="or" minOccurs="0" maxOccurs="unbounded" />
      <xs:element ref="nor" minOccurs="0" maxOccurs="unbounded" />
      <xs:element ref="sum-of-products" minOccurs="0" maxOccurs="unbounded" />
      <xs:element ref="product-of-sums" minOccurs="0" maxOccurs="unbounded" />
    </xs:choice>
    <xs:attribute name="id" type="xs:string" use="required" />
    <xs:attribute name="name" type="xs:string" use="required" />
  </xs:complexType>
  <xs:attribute name="type" type="AT_type" />
  <xs:attribute name="reporting" type="xs:boolean" default="false" />
  <xs:attribute name="power-region" type="xs:boolean" default="false" />
  <xs:attribute name="target" type="xs:integer" />
  <xs:attribute name="start" type="AT_start" default="none" />
  <xs:attribute name="activation" type="AT_activation" default="none" />
  <xs:attribute name="reset" type="xs:string" />
  <xs:attribute name="output-on-high" type="xs:boolean" />
  <xs:attribute name="negate-inputs" type="xs:boolean" />
  <xs:attribute name="latch" type="xs:boolean" />
  <xs:attribute name="case-insensitive" type="xs:boolean" />
  <xs:attribute name="high-only-on-eod" type="xs:boolean" />
  <xs:attribute name="countone" type="xs:string" />
  <xs:attribute name="macro" type="xs:string" />
  <xs:attribute name="element" type="xs:string" />
  <xs:attribute name="at_target" type="AT_countermodes" />
  <xs:element name="report-on-match" type="T_report-on-match" />
  <xs:element name="activate-on-match" type="T_activate-on-match" />
  <xs:element name="state-transition-element" type="T_state-transition-element" />
  <xs:element name="report-on-target" type="T_report-on-target" />
  <xs:element name="activate-on-target" type="T_activate-on-target" />
  <xs:element name="counter" type="T_counter" />
  <xs:element name="activate-on-high" type="T_activate-on-high" />
  <xs:element name="report-on-high" type="T_report-on-high" />
  <xs:element name="and" type="T_or_and" />
  <xs:element name="or" type="T_or_and" />
  <xs:element name="inverter" type="T_nor_nand_inverter" />
  <xs:element name="nand" type="T_nor_nand_inverter" />
```

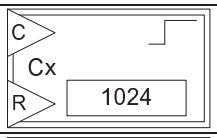
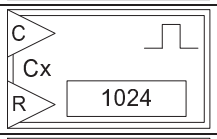
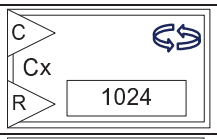
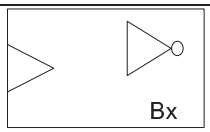
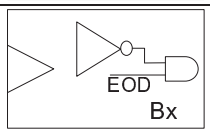
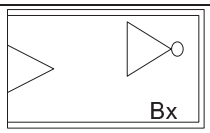
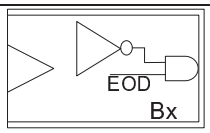
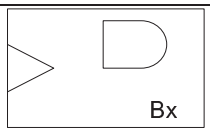
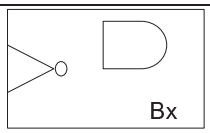
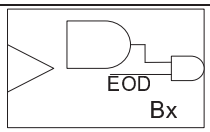
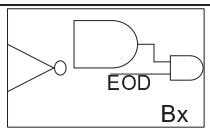
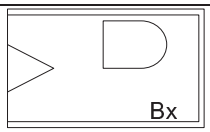
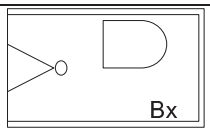
```
  <xs:element name="nor" type="T_nor_nand_inverter" />
  <xs:element name="product-term" type="T_product-term" />
  <xs:element name="sum-of-products" type="T_sum-of-products" />
  <xs:element name="sum-term" type="T_sum-term" />
  <xs:element name="product-of-sums" type="T_product-of-sums" />
  <xs:element name="regex" type="T_regex" />
  <xs:element name="element-reference" type="T_element-reference" />
  <xs:element name="port-definition" type="T_port-definition" />
  <xs:element name="macro" type="T_macro" />
  <xs:element name="automata-network" type="T_automata-network" />
</xs:schema>
```

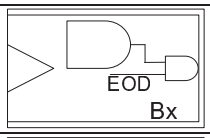
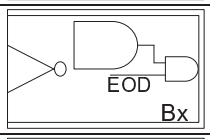
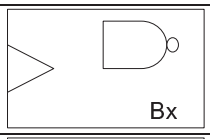
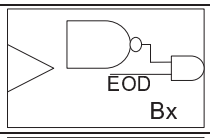
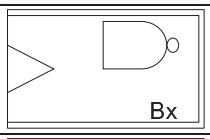
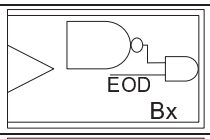
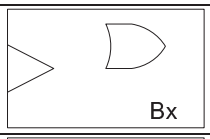
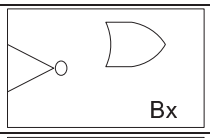
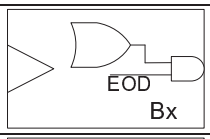
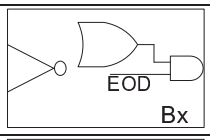
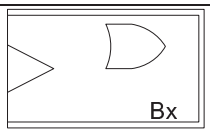
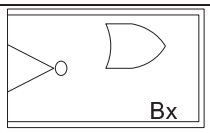
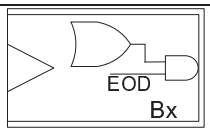
Appendix 3 ANML-G Elements Catalog

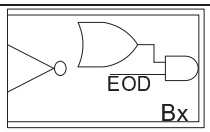
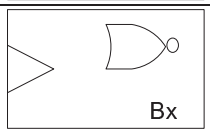
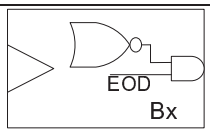
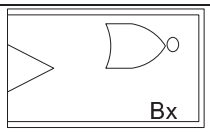
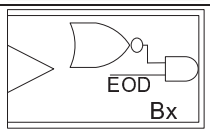
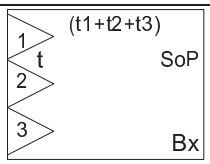
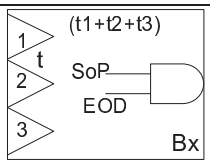
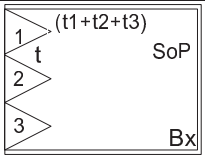
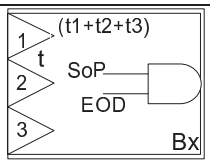
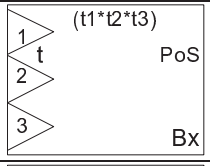
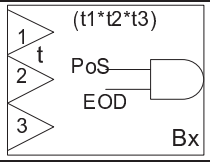
Appendix 3 ANML-G Element Catalog

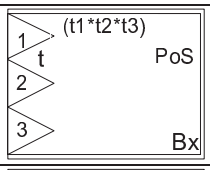
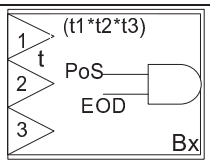
Consolidated Symbols List:

STE	Activate-on-match	
	Report output	
	Self-Activating	
	Latched activate-on-match	
	Latched report output	
	Latched report output and activate-on-match	
R2	Activate-on-match	
	Report-on-match	
Counter	Activate-on-match Latch	
	Activate-on-match Pulse	
	Activate-on-match Roll	

	Report-on-match Latch	
	Report-on-match Pulse	
	Report-on-match Roll	
Combinatorial (Inverter)	Activate-on-high	
	Activate-on-high with EoD	
	Report-on-high	
	Report-on-high with EoD	
Combinatorial (AND)	Activate-on-high	
	Activate-on-high with negate-inputs	
	Activate-on-high with high-only-on-eod	
	Activate-on-high with negate-inputs & high-only-on-eod	
	Report-on-high	
	Report-on-high with negate-inputs	

	Report-on-high with high-only-on-eod	
	Report-on-high with negate-inputs & high-only-on-eod	
Combinatorial (NAND)	activate-on-high	
	activate-on-high with high-only-on-eod	
	Report-on-high	
	Report-on-high with high-only-on-eod	
Combinatorial (OR)	activate-on-high	
	activate-on-high with negate-inputs	
	activate-on-high with high-only-on-eod	
	activate-on-high with negate-inputs & high-only-on-eod	
	Report-on-high	
	Report-on-high with negate-inputs	
	Report-on-high with high-only-on-eod	

	Report-on-high with negate-inputs & high-only-on-eod	
Combinatorial (NOR)	activate-on-high	
	activate-on-high with high-only-on-eod	
	Report-on-high	
	Report-on-high with high-only-on-eod	
Combinatorial (SoP)	activate-on-high	
	activate-on-high with high-only-on-eod	
	Report-on-high	
	Report-on-high with high-only-on-eod	
Combinatorial (PoS)	activate-on-high	
	activate-on-high with high-only-on-eod	

	Report-on-high	
	Report-on-high with high-only-on-eod	

Appendix4 Micron Automata Processor Implementation Elements Summary

Appendix 4 Micron Automata Processor Implementation Elements Summary

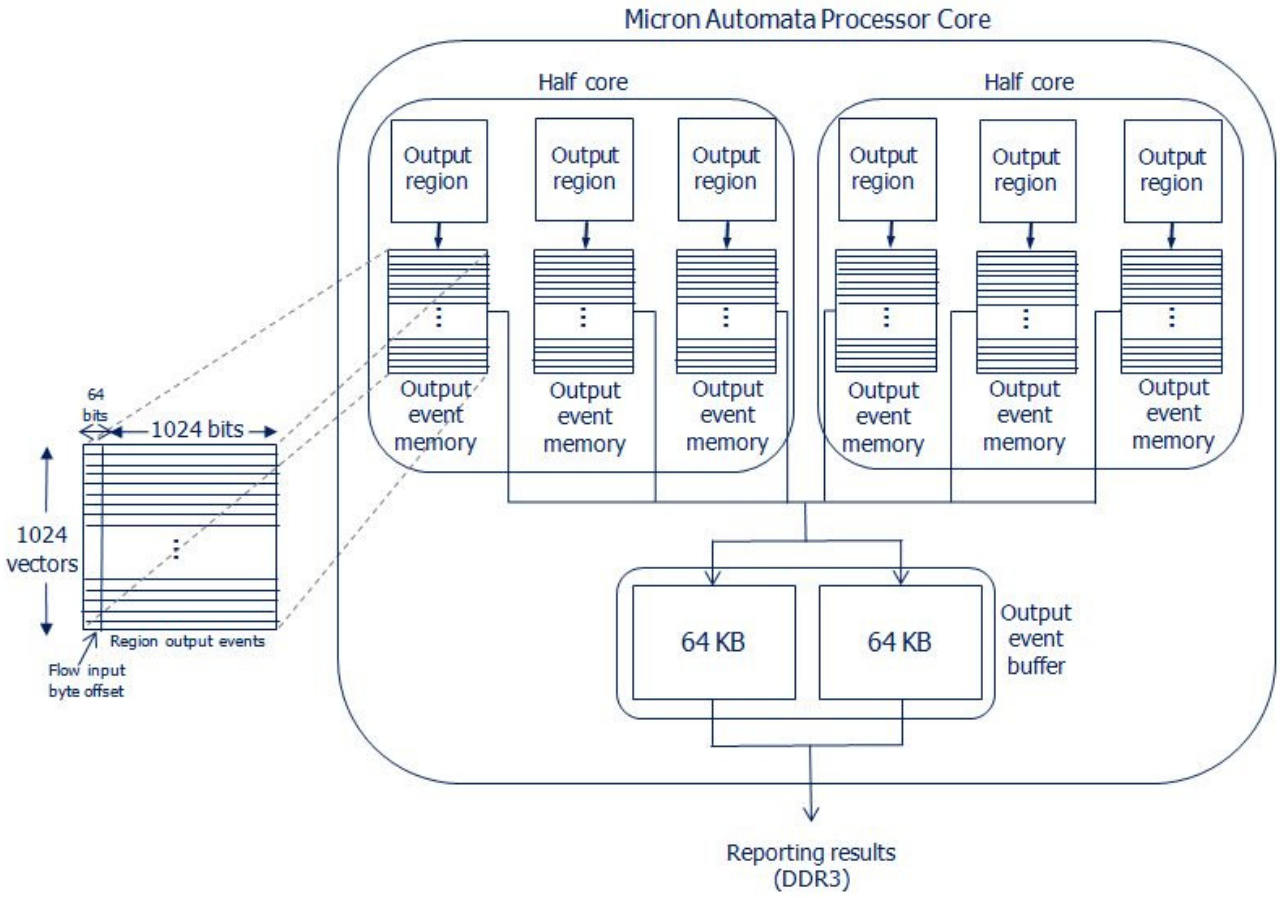
The table below describes details specific to implementation of ANML by the Micron Automata Processor.

Element	Input options	Output options	Output type	Match output option	Availability	ID Proposal	Comments (NOT applicable to ANML users)
STE	Start-of-data, All-input, STE, Counter, Combinatorial	STE, Counter, Combinatorial	Latched or Unlatched	Yes	49152 (256 per block) in two cores (24576 per core). 6144 can generate output. (32 per block, 2 per row)	S*x*	STE in GOT 3, 6 or 7 can output. Two outputs per ROW, 2-to-1 MUX for GOT 3 & BC/CNT + 2-to-1 MUX for GOT 6 & GOT 7. 16 rows per block. 192 blocks. STEs cannot be activate other STEs across cores, each core operates synchronously on the input but independently.
Counter	STE	STE	Pulsed or Latched or Roll	Yes	768 (4 per block) in two cores.	C*x*	Can connect to elements within block only. Counter can't feed into another counter or combinatorial element with exception that one counter can reset another counter.
Combinatorial	STE	STE	Unlatched	Yes	2304 (12 per block)	B*x*	Can connect to elements within block only. Combinatorial element can't feed into another counter or combinatorial element.

Appendix 5 Micron Automata Processor Implementation Output Processing and Performance

Appendix 5 Micron Automata Processor Implementation: Output Processing and Performance

Micron Automata Processor Output Processing Architecture



Output Event Processing

There are 6 output (match) regions of the chip, each containing 1024 output lines capable of reporting output events from automata elements on a single symbol cycle for a total of 6144 on the entire chip. Each output region will produce a output event vector with at least 64 bits to as much as of 1024 bits (plus 64 bits of metadata containing the byte offset in the flow where the output event occured) on each symbol cycle on which there is output in that region. The reduction of the size of the event vector is known as event vector division. The event vector size can be reduced by a fixed divisor with possible divisor values of: 1 (no reduction), 1.33, 2, 4, 8 and 16. The event vector divisor will be the same for all regions.

There is a direct relationship between the size of the event vector and the number of symbol cycles needed to transfer it between the chip core and event buffer. When the output rate is high much better performance will be obtained with smaller event vectors. The size of the event vector is set at compilation time and is based on the number of automata elements which have been configured for output and the level of the success of the place and route algorithm in positioning output elements on the chip such that the smallest possible reduced event vector size may be used. It may occur that even though the number of output elements is less than a possible event vector size the output elements cannot be positioned within the physical constraints of a smaller event vector and a larger vector must be used to ease placement. The automata processor developer can improve the overall situation simply by creating ANML designs with as few outputting elements as possible. Through experience, the designer may learn that some designs "route better" than other designs and result in greater reduction of the event vector.

If there is a single outputting automata element in a region on a symbol cycle the entire vector, with just a single bit set will be written to the output event memory. If the width of the event vector is 1024, 1023 extraneous bits are written, if it 64 only 63 are written. If there are multiple outputting automata elements in a region on a single cycle still just one vector will be written to the output event memory but more output event bits in that vector will be set. The ANML designer can improve the efficiency of output operations by getting more output information into the event vector with higher utilization of the available bits. If there is no output event in an output region an output event vector is not written to output event memory. Each output region can hold up to 1024 vectors. Although capacity exists for 1024 vectors if compression is not enabled the number of vectors that should actually be stored in the output region memory is 481, the limit of the output buffer to which vectors are transferred for output off the chip.

To report output events to the user the output event vectors must be transferred to an event buffer before they can be read off chip. The transfer time for each uncompressed output event vector is between 40 symbol cycles for a 1024-bit vector to 2.5 symbol cycles for 64-bit vector. Reading the first output event vector involves start-up overhead and takes an additional 15 symbol cycles. Determining that an output region has no output event vectors when a request to transfer the region has been made takes 2 symbol cycles. The instruction set allows any combination of output regions to be selected for a transfer, including a single region, so it is possible to avoid the 2 symbol cycle overhead for transfer of empty regions if supported by the runtime software layer. At the present time the API does not enable the user to specify that output should be restricted to a specific region and the user cannot know if all regions or some combination of regions are specified in the output request.

The compiler (place and route and loading) determines where in the six possible regions automata elements which are output-enabled will be placed. Significant differences in performance may be obtained depending on where the output automata elements are placed, not with respect to event vector dvision but also in region placement. For example, if there are six output events at a single symbol cycle and the automata element associated with those output events are placed into the six different regions transferring an event vector of 1024-bits will take 255 cycles (6 * 40 + 15). If those six automata elements were in the same region and the event vector were only 64-bits that time could potentially be reduced to 17.5 cycles, 2.5 for the cost of transfering the one region with matches and 15 cycles overhead. When many event vectors are buffered and transferred in a single operation to the event memory the overhead is amortized over many vectors and the ratio between best and worst cases becomes about 100 to 1 - 240 cycles per set of 6 vectors versus 2.5 cycles for a single region 64-bit event vector.

At this time there is no way for the ANML developer to specify where output automata elements should be placed and therefore no way to explictly attempt to improve performance by limiting the number of output regions. In the future the compiler may try, automatically, to reduce the number of output regions and an ANML parameter may be added to tell the compiler that a set of output automata elements should be grouped into a single region if possible.

The Automata Processor is divided into two half-cores which operate synchronously on the input but independently. Automata elements in one half-core cannot activate automata elements in the other half-core. With respect to output processing this means that it is not possible to reduce the number of output regions to 1 and use both half-cores unless it possible to have independent processing on one half-core without generating any output. The more common situation would be that the number of output regions would be limited to two, with each independent circuit on each half-core having output automata elements in one region each. In the least optimized case the minimum output processing cost should be calculated using 2 ouput regions. Further optimizations, however, are possible. The output over a range of input symbol cycles may be limited to one region in one core. Output events may be triggered in one region and not in the other region in the other half-core. If the software enables such an operation the populated region in this case might be the only region for which output is requested. If the software does not enable specification of the output region the cost for transfer of an unpopulated region would only be 2 symbol cycles so two regions in two half-cores could be transferred in 42 symbol cycles for a 1024-bit vector or 4.5 symbol cycles for a 64-bit vector. The key thing here is to have control over when output is transferred so that at any transfer only one region contains data. (The API functions critical to this are AP_ScanFlows and AP_GetMatches).

All of the output vectors in match memory for whatever regions are specified are transferred in one burst. The 15 symbol cycle overhead cost is incurred for each burst. The tables below shows the number of output elements

available by number of regions for each possible value of the event vector divisor and the transfer times in symbol cycles by number of regions for each possible value of the event vector divisor.

Output Vector Number of Elements and Transfer Time in Symbol Cycles by Number of Regions and EV Divisor

Regions	Maximum Output Elements (by EV Divisor)						Vector Transfer Cycles (by EV Divisor)						Overhead
	1	1.33	2	4	8	16	1	1.33	2	4	8	16	
1	1024	768	512	256	128	64	40	30	20	10	5	2.5	15
2	2048	1536	1024	512	256	128	80	60	40	20	10	5.0	15
3	3072	2304	1536	768	384	192	120	90	60	30	15	7.5	15
4	4096	3072	2048	1024	512	256	160	120	80	40	20	15.0	15
5	5120	3840	2560	1280	640	320	200	150	100	50	25	17.5	15
6	6144	4608	3072	1792	768	384	240	180	120	60	30	20.0	15

Minimum Output Vector Transfer Time (1 64-bit vector in one region) in Symbol Cycles Without Region Selection (all regions output including empty ones)

Populated Regions	Empty Regions	Populated Vectors	Overhead	Vector Transfer	Empty Region Processing	Total
1	5	1	15	2.5	2*5=10	27.5

Maximum Output Vector Transfer Time (1024 1024-bit output vectors per region) for Full Event Memory

Regions	Total Vectors	Overhead	Vector Transfer	Total Symbol Cycles	Total Time (@7.45ns per symbol cycle)
1	1024	15	40960	40975	0.3 ms
2	2048	15	81920	81935	0.6 ms
3	3072	15	122880	122895	0.9 ms
4	4096	15	163840	163855	1.2 ms
5	5120	15	204800	204815	1.5 ms
6	6144	15	245760	245775	1.8 ms

Examples

Output of all six regions is requested.

1) Region 0 has 1 output event vector
2) Region 1 has no output event vectors
3) Region 2 has no output event vectors
4) Region 3 has no output event vectors
5) Region 4 has no output event vectors
6) Region 5 has no output event vectors

Event Vector Divisor	Transfer Time in Symbol Cycles
1	15 (overhead) + 40 (region 0: transfer 1 output ev) + 2*5 (region 1,2,3,4,5: NULL transfer) = 65
1.33	15 (overhead) + 30 (region 0: transfer 1 output ev) + 2*5 (region 1,2,3,4,5: NULL transfer) = 55
2	15 (overhead) + 20 (region 0: transfer 1 output ev) + 2*5 (region 1,2,3,4,5: NULL transfer) = 45
4	15 (overhead) + 10 (region 0: transfer 1 output ev) + 2*5 (region 1,2,3,4,5: NULL transfer) = 35
8	15 (overhead) + 5 (region 0: transfer 1 output ev) + 2*5 (region 1,2,3,4,5: NULL transfer) = 30
16	15 (overhead) + 2.5 (region 0: transfer 1 output ev) + 2*5 (region 1,2,3,4,5: NULL transfer) = 27.5

Another example:

Output of all six regions is requested.

1) Region 0 has 1 output event vector
2) Region 1 has no output event vectors
3) Region 2 has 4 output event vectors
4) Region 3 has no output event vectors
5) Region 4 has no output event vectors
6) Region 5 has no output event vectors

The transfer time would be 15 (overhead) + 40 (region 0: transfer 1 output event) + 2 (region 1: NULL transfer) + 4*40 (region 2: transfer 4 output events) + 2*3 (region 3,4,5: NULL transfer) = 223 cycles.

Event Vector Divisor	Transfer Time in Symbol Cycles
1	15 (overhead) + 40 (region 0: transfer 1 output ev) + 2 (region 1: NULL transfer) + 4*40 (region 2: transfer 4 output ev) + 2*3 (region 3,4,5: NULL transfer) = 223
1.33	15 (overhead) + 30 (region 0: transfer 1 output ev) + 2 (region 1: NULL transfer) + 4*30 (region 2: transfer 4 output ev) + 2*3 (region 3,4,5: NULL transfer) = 173
2	15 (overhead) + 20 (region 0: transfer 1 output ev) + 2 (region 1: NULL transfer) + 4*20 (region 2: transfer 4 output ev) + 2*3 (region 3,4,5: NULL transfer) = 123
4	15 (overhead) + 10 (region 0: transfer 1 output ev) + 2 (region 1: NULL transfer) + 4*10 (region 2: transfer 4 output ev) + 2*3 (region 3,4,5: NULL transfer) = 73
8	15 (overhead) + 5 (region 0: transfer 1 output ev) + 2 (region 1: NULL transfer) + 4*5 (region 2: transfer 4 output ev) + 2*3 (region 3,4,5: NULL transfer) = 48
16	15 (overhead) + 2.5 (region 0: transfer 1 output event) + 2 (region 1: NULL transfer) + 4*2.5 (region 2: transfer 4 output events) + 2*3 (region 3,4,5: NULL transfer) = 35.5

Output event vectors can be compressed. It has not yet been determined what the timing would be for compressed output event vector transfers from output event memory to the output event buffer.

Transfers from the output event memory to the user-accessible output event buffer are concurrent with other chip operations. This may "hide" some of the cost of the transfer from event memory to the event buffer but, in any case, the overall time will not be less than the total time consumed by event vector transfer.

Performance and Output Processing

Chip performance will be throttled by transfer time between output event memory and the output event buffer if more than one output event vector is generated every 40/event-vector-divisor symbol cycles (that is, 40, 30, 20, 10, 5 or 2.5, depending on what divisor the compiler is able to use). Because there are six regions it is possible to generate as much as six output vectors per input symbol cycle, giving a worst case degradation of performance of 240/event-vector-divisor times the input rate.

The only way to mitigate this problem in high output scenarios is to aggregate output events - that is, to reduce the number of output vectors by combining events over many symbol cycles into fewer vectors. If we have 1 output event per input symbol in a region we will write a 1088 bit vector, which can take as much as 40 symbol cycles, depending on the event vector divisor, to transfer on every symbol just to convey one bit of information. If we can aggregate events of 40 symbol cycles, writing still just one vector but using 40 out the 1024 available bits we can run at the input symbol cycle rate. The ANML Cookbook shows many examples of output aggregration with techniques using timing state-transition-elements, counters, and the end-of-data signal enabling a combinatorial gate.

Compression

The preceding discussion assumed that the output vector is not compressed. Data is not yet available for the performance consequences of adding compression to the vector transfer time. A reasonable assumption is that it will, on average, significantly increase it. In deciding whether or not to use compression analysis must be made of the expected compression rate, size of the output buffer, and transfer time of data from the output buffer to the host processor. The transfer time of data from the output buffer to the host processor is also a potential bottleneck. The application architecture must balance both the output memory transfer and the output buffer transfer to maximize performance.

Output Vector Format

The Automata Processor API interprets the output buffer containing output vectors and will report an id that can be mapped to the ANML id associated with each output event and the byte offset in the input *flow* which triggered the output event. There may be instances where it could be more efficient for the application to handle the output buffer directly. At present, however, it may not be possible for the user application to detect region boundaries, although this may addressed in the future with the addition of a region header. Each region section consists of populated output vectors for that region. The output vector has a 64-bit metadata field consisting of a 32-bit byte offset in the flow to the symbol that caused the output event and 1024 bits representing the output state of each possible output event in the region. The position of each event bit in the output vector is associated with a physical address on the chip. It is necessary to have results from compilation of the ANML description giving the correlation between these physical addresses on the chip at the ANML elements associated with output events to interpret the event settings in the output vector. Additional functionality in the Automata Processor SDK may be necessary to enable the application

developer to obtain this information from the compilation step. It is also possible for multiple flows to be represented in the output buffer but there is no information in the output vector about the identify of the source flow. This information is added to match results by the Automata Processor software.

Uncompressed the size of a NULL region is 64 bits and a populated region is (64 + 1024 bits) x # output vectors. In the first example above with 1 vector in one region and 5 empty regions the total buffer size would be 1088 (region 0) + 64*5 (region 1,2,3,4,5) = 1408 bits or 176 Bytes. The second example with 1 vector in one region, 4 vectors in another region, and 4 empty regions would have a total buffer size of 1088 (region 0) + 64 (region 1) + 4352 (region 2) + 64*3 (region 3,4,5) = 5696 bits or 712 Bytes. The output buffer consists of two ping-pong half-buffers of 64K each. Uncompressed each half-buffer can hold 481 output vectors. Without using compression the number of state vectors that can reside in a region's match memory is effectively reduced to 481, less than the match memory capacity of 1024 event vectors.

The output buffer may also be compressed, depending on the configuration, potentially controllable by the user through a setting in the Automata Processor Runtime API. The output buffer will be automatically uncompressed by the Automata Processor API. If the application designer does not use the API to interpret the output buffer it will be necessary to manually uncompress it. This functionality may not be available as an independant operation in the API.

State Vector

The Automata Processor state vector contains the current state of the AP elements at a given time. The Automata Processor on-chip state vector cache allows storage of up to 512 state vectors. If there is a need to save more than 512 then the state vectors can be moved to system memory and retrieved when required. Every flow being processed has an associated state vector. A single state vector constitutes of 59936 bits [(256 enable bits per block + 56 counter bits per block) * 192 blocks + 32 count] and it takes 1668 symbol cycles to transfer state vector from the state vector cache to the save buffer. Even though the State vector and Event vector are independent of each other, AP uses the same internal bus and compressor (if enabled) for transferring the state vector and the event vector to the respective buffers. i.e. only of one of them can be transferred at a time.

Appendix 6 Micron Automata Processor Implementation SDK Notes for ANML Designers

Appendix 6 Micron Automata Processor Implementation: SDK for ANML Designers

Runtime Functions and Data Structures

device	
A device is a logical Automata Processor. A logical Automata Processor may correspond to one or more physical chips. Multiple chips are organized on ranks and rank information is part of the configuration of the device. A device must be set up before automata can be loaded to it and input (flows) can be run on it.	
main data structure	*ap_device_t*
AP_GetVersionString	device/driver/chip version
AP_CreateDevice	sets up device (*ap_device_config,* setting the number of ranks, number of physical devices per rank, and the partitioning of logical cores over physical devices in a rank, and other device parameters)
AP_EnableCompression	Configures device operation to compress state vectors (for off-chip storage of flow states) and device automata images (program). Output event vector compression settings are not currently supported.
AP_OpenDevice	makes device ready to process input
AP_GetMatches	returns output event data (*ap_match_result*, compiler-generated output ID (not the ANML ID) and byte offset in the flow for each output event). The flow is not identified in the output event data. If there are multiple flows associated with the device it is necessary to run *AP_ScanFlows* followed by *AP_Wait* followed by *AP_GetMatches* to ensure that the user knows which flow's matches are being returned from *AP_GetMatches*.
AP_QueryDeviceCount	returns number of devices on the system
AP_QueryDeviceMetrics	returns *ap_device_metrics* datastructure (hw ID, #ranks, rank configuration (*ap_device_config*), memory allocation, max # flows, /dev name)
AP_CloseDevice	shuts down device

automaton/runtime object	
An automaton is created in the compilation phase and describes the physical circuit which will can be run on a device (a logical Automata Processor). Once the automaton is loaded to the device it becomes a runtime object. Input data (flows) can be processed by a runtime object.	
main data structures	*ap_automaton_t* an automaton before it is loaded to a device (compilation output) *ap_rto* runtime object, a runnable automaton after it is loaded to a device
AP_GetInfo	Returns *ap_automation_info* (#subgraphs, block info, info on STEs, counters and combinatoral elements)
AP_Duplicate	Creates a duplicate copy of an automaton.
AP_Save	Saves an automaton to a file.
AP_Restore	Restores an automaton from a file.
AP_Destroy	Destroys and automaton and frees associated resources.
AP_Load	Loads an automaton onto a device, making it a runnable runtime object.
AP_Find	Returns a runtime object searching by a character name identifier provided by the user when the runtime object is loaded.
AP_Unload	Unloads a runtime object from a device.

flow	
A flow is a data stream run against a runtime object.	
main data structure	*ap_flow_t*
AP_OpenFlow	Associates a flow to a runtime object on a device.
AP_CloseFlow	Closes a flow on a device.
AP_ScanFlows	Processes data chunks(s) against a runtime object. Data is defined in *ap_flow_data* (flow object, length, number of chunks, chunks, eod marker (s)). Runs asynchronously but can be followed by *AP_Wait* to suspend further processing until AP_ScanFlows finishes.
AP_Wait	Waits for a designated operation to complete. *AP_Wait* operates on a device, not a flow, but the only operation which can create an operation parameter for *AP_Wait* is *AP_ScanFlows*, linking *AP_Wait* to the synchronization of processing of flows. Matches are not identified by the flow so when there are multiple flows run on a device it is necessary to run *AP_ScanFlows* followed by *AP_Wait* followed by *AP_GetMatches* to ensure that the user knows which flow's matches are being returned from *AP_GetMatches*.