Sticks and Stones: Making the Most of a Single Unbounded Dimension

Amir Pnueli

The Weizmann Institute of Sciences
The John von Neumann Minerva Center for the Verification of Reactive Systems

Autrans, June 2000

Joint work with:

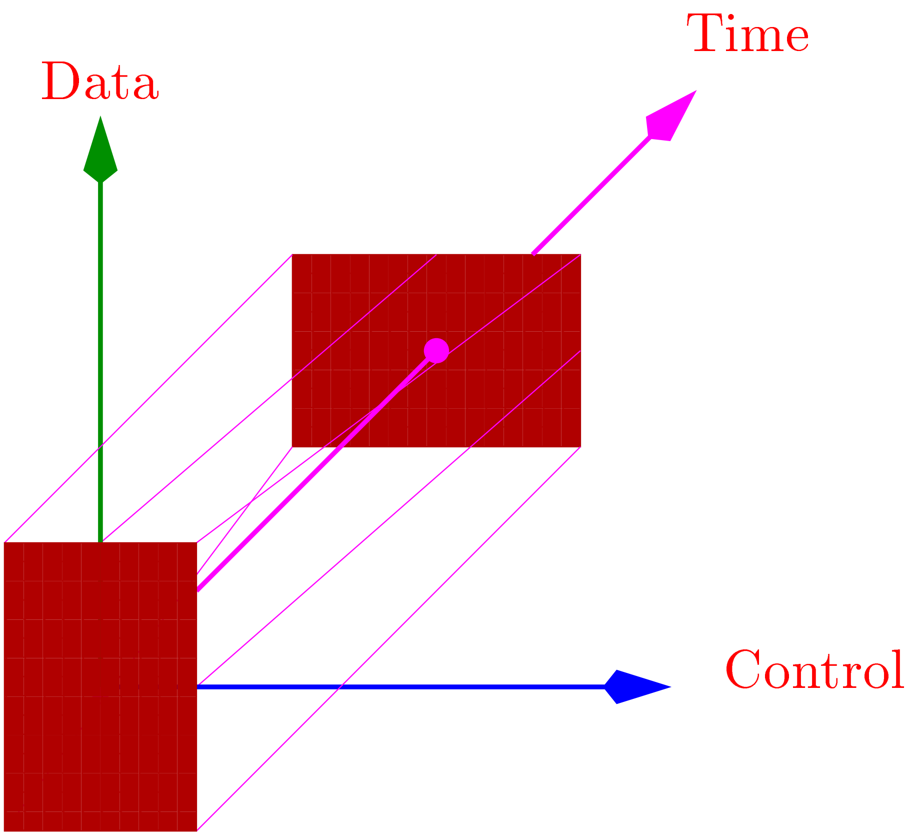
Elad Shahar and Dana Fisman

A Folk Saying

Sticks and Stones may Break my Bones, but
Words would Never Harm me.

The 3D World of Verification

Formal Verification can be viewed as taking place in a 3-dimensional world.



At any time instance $t \geq 0$, we have a set of states which are reachable at t. This defines a (possibly time-dependent) relation between the control and data system variables, such as

$$at\_\ell_{0,1} + at\_m_{0,1} + y = 1,$$

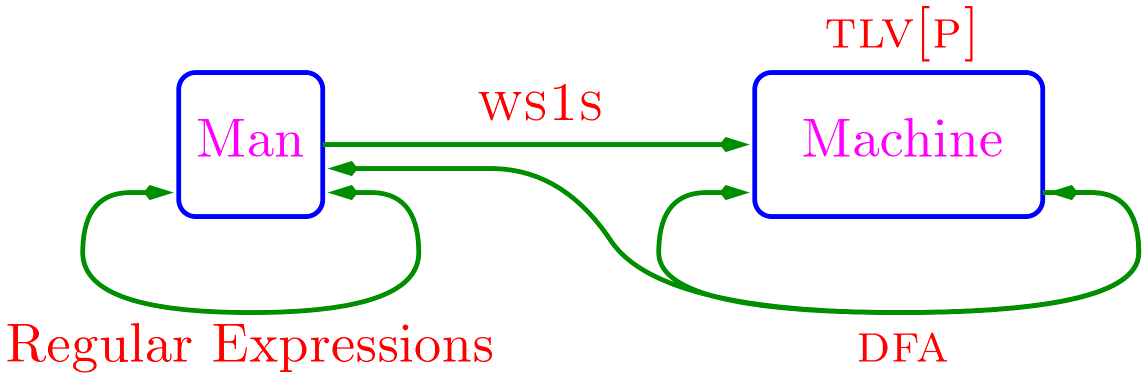
or

$$t = 1 \rightarrow at\_\ell_{0,1}$$

Regular Languages

The machinery of Finite-State Automata/Regular Expressions is one of the most powerful tools in Computer Science for dealing with infinite sets which have a finite representation. It allows us to manipulate sets of objects of unbounded sizes. With the extension to Büchi (ω-) Automata, we can also deal with infinite objects.

There are three languages of equal expressive power, which are relevant to this machinery.



The Logic WS1S

We use the logic WS1S (weak second order logic of one successor) [Th90] which is very similar to the language M2L used in MONA.

Assume a signature $\Xi : \{\Sigma_1, \ldots, \Sigma_k\}$ consisting of a finite set of finite alphabets. The vocabulary consists of position variables $p_1, p_2, \ldots$ and, for each $\Sigma_i \in \Xi$, a set of Σ_i-string variables $X_i, Y_i, Z_i, \ldots$. Variable M denotes the upper bound on the length of all strings and all position variables.

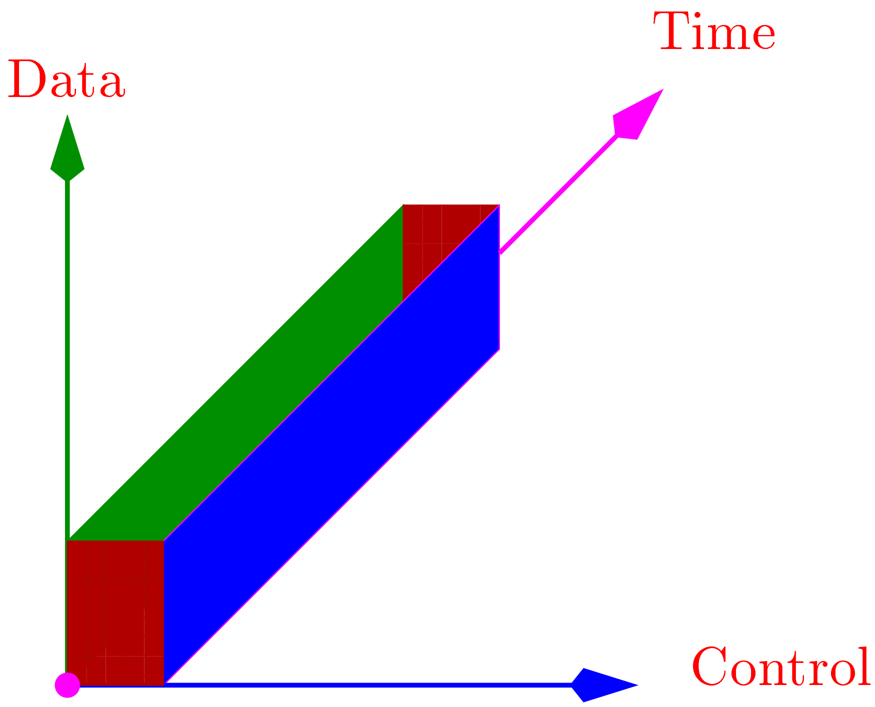
- Position (First-order) terms: The constant 1 and any position variable p_i are position terms. If t is a position term then so is $t + 1$.
- Letter terms:
  - Every $a \in \Sigma_i$ is a Σ_i-term.
  - If X is a Σ_i-string variable and t is a position term, then $X[t]$ is a Σ_i-term.
- Atomic Formulas:
  - $t_1 \sim t_2$, where t_1 and t_2 are position terms and $\sim \in \{=, <\}$.
  - $x = y$, where x and y are Σ_i-terms for some $\Sigma_i \in \Xi$.
- Formulas:
  - An atomic formula is a formula.
  - If φ, ψ are formulas, then so are $\neg\varphi$, $\varphi \vee \psi$, $\exists p : \varphi$, $\exists X : \varphi$, for position variable p and string variable X.

Unfortunately

All attempts to generalize the formalism of automata/regular expressions to true 2 dimensions have failed miserably. All problems become undecidable.

If we wish to apply the tool of regular languages to verification, this forces us to choose which of the 3 dimensions will enjoy the benefits of this tool.

Applying REG to Time



We keep control and data bounded, and apply REG to unbounded time. Assume that the state is described by the (finite-range) state variable X, the initial condition is given by $\Theta(X)$, and the transition relation is given by $\rho(X, X')$. Verifying that $\psi(X)$ is $\mathcal{D}$-valid amounts to checking the validity of the WS1S formula

$$\Theta(X[1]) \;\wedge\; (\forall i : \rho(X[i], X[i+1])) \quad \rightarrow \quad \psi$$

where, for example, we can declare X as

$$X : \textbf{string of } \langle \pi_1, \pi_2 : [0..3]; y : \textbf{boolean} \rangle$$

and ψ defined as

$$\psi : \quad \forall j \neg (X[j].\pi_1 = 3 \;\wedge\; X[j].\pi_2 = 3)$$

This Establishes Mutual Exclusion

of the following program:

$$y: \textbf{ natural initially } y = 1$$

$$\left[\begin{array}{ll} \ell_0: & \textbf{loop forever do} \\ & \left[\begin{array}{ll} \ell_1: & \textbf{Non-critical} \\ \ell_2: & \textbf{request } y \\ \ell_3: & \textbf{Critical} \\ \ell_4: & \textbf{release } y \end{array}\right] \end{array}\right] \parallel \left[\begin{array}{ll} m_0: & \textbf{loop forever do} \\ & \left[\begin{array}{ll} m_1: & \textbf{Non-critical} \\ m_2: & \textbf{request } y \\ m_3: & \textbf{Critical} \\ m_4: & \textbf{release } y \end{array}\right] \end{array}\right]$$

$$- \quad P_1 \quad - \qquad\qquad\qquad - \quad P_2 \quad -$$

If we wish to use this method for verifying liveness properties, we will have to move to S1S and ω-automata.

This approach has not been thoroughly investigated, even though it has some interesting connections with bounded model checking.

Not Wishing to Waste our Precious Resource on Time

If we do not wish to waste the precious REG resource on time, we can dedicate it to the encoding of an unbounded static state space, either through unbounded control or through unbounded data.

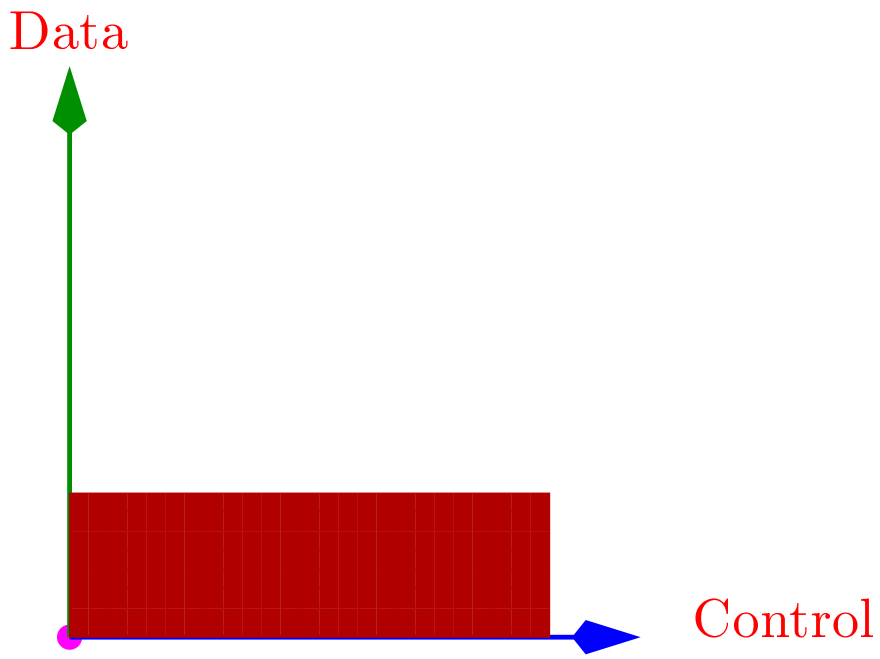
Time is dealt with through the dynamic computation of symbolic model checking, where the assertional language used for describing sets of states will have the expressive power of REG.

Procedure ***Compute Reachable***
  $\varphi_0, \varphi_1, \ldots$: **assertion**
  Let $\varphi_0 := \Theta$
  For $i = 0, 1, \ldots$ **repeat**
    Let $\varphi_{i+1} := \varphi_i \vee succ(\varphi_i)$
  until $\varphi_{i+1} = \varphi_i$
end procedure

where,

$$succ(\varphi) : \quad unprime(\exists V : (\varphi(V) \ \wedge \ \rho(V, V')))$$

Applying REG to Control



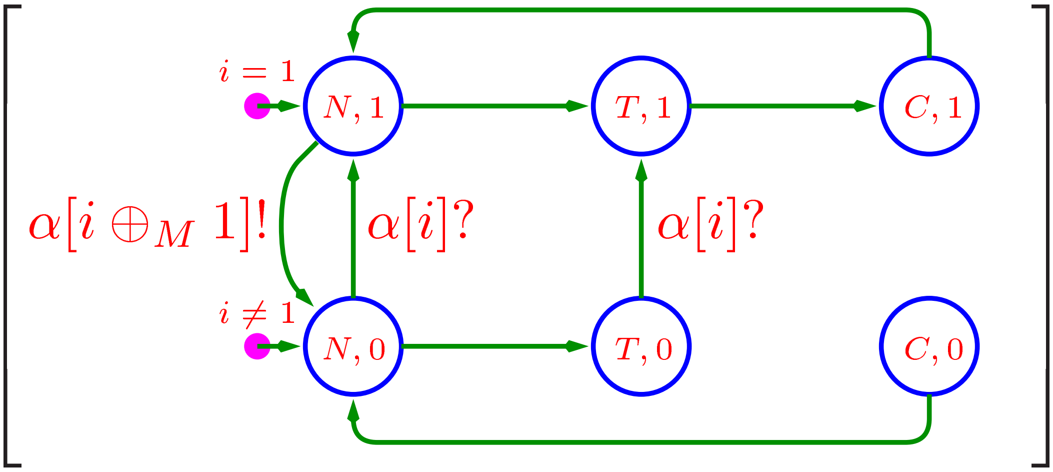
We use REG to encode unbounded control structures, e.g., an unbounded number of processes. However, we have to keep the data dimension bounded, i.e., a finite data domain for each process.

Example: Process Ring

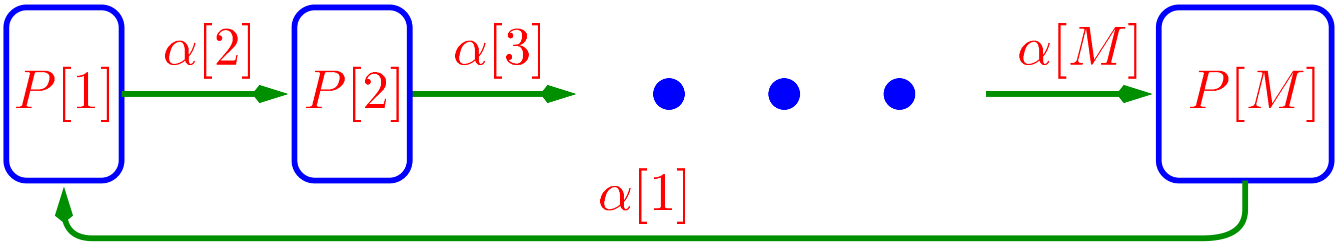
Consider the system:

in	M	:	**integer where** $M > 1$
local	α	:	**array** $[1..M]$ **of channel of boolean**

$\overset{M}{\underset{i=1}{\|}} \; P[i] ::$



with architecture:



Regular Model Checking

It is possible to apply symbolic model checking, where we use regular expressions as the (symbolic) assertional language used to describe sets of states.

Local state	Σ :	$\{N, T, C, N_t, T_t, C_t\}$
Global state (configuration):		A word in Σ^+
Initial Condition	Θ :	$N_t N^*$
Transitions		Given by rewrite rules:

$$\text{Unary: } \left\{ \begin{array}{ccc} N & \rightarrow & T \\ C & \rightarrow & N \\ N_t & \rightarrow & T_t \\ T_t & \rightarrow & C_t \\ C_t & \rightarrow & N_t \end{array} \right. \qquad \text{Binary: } \left\{ \begin{array}{ccc} N_t N & \rightarrow & N N_t \\ N_t T & \rightarrow & N T_t \end{array} \right.$$

Computing the set of reachable global states, we get

$$Reach = \quad (N+T)^*(N_t + T_t + C_t)(N+T)^*$$

The property of mutual inclusion, which is the violation of mutual exclusion can be specified by

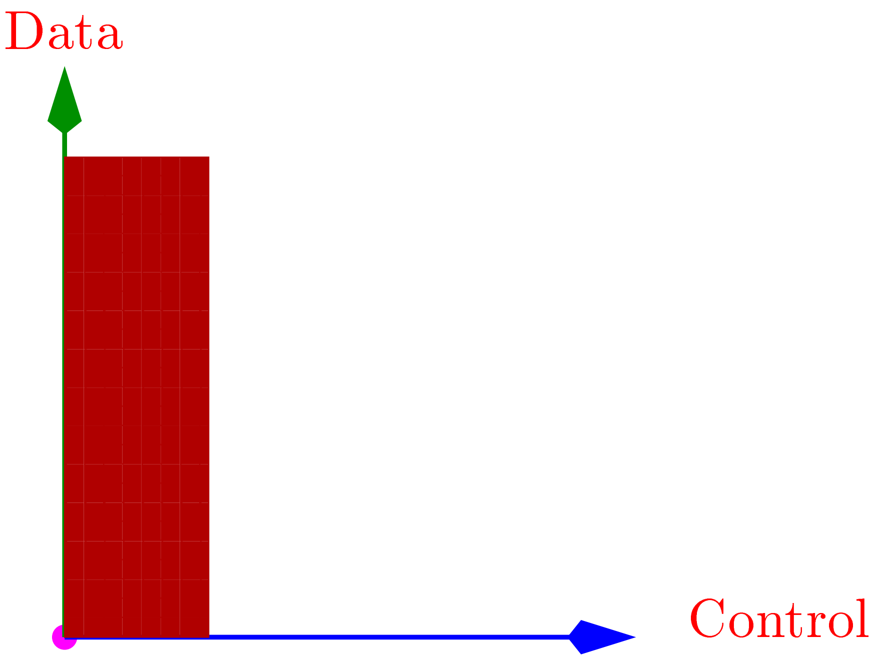
$$Inclusion = \quad \Sigma^*(C+C_t)\Sigma^*(C+C_t)\Sigma^*$$

Taking their intersection, we obtain

$$Reach \cap Inclusion = \emptyset$$

which leads to the conclusion that process ring satisfies mutual exclusion for every value of the parameter N.

Applying REG to Data



We can use REG to encode unbounded data, e.g., a finite number of integer variables. However, we have to keep the control dimension bounded, i.e., a fixed number of processes.

Example: Program BAKERY-2

local y_1, y_2 : **natural initially** $y_1 = y_2 = 0$

$$P_1 :: \left[\begin{array}{l} \ell_0 : \textbf{loop forever do} \\ \left[\begin{array}{ll} \ell_1 : & \textbf{Non-Critical} \\ \ell_2 : & y_1 := y_2 + 1 \\ \ell_3 : & \textbf{await } y_2 = 0 \ \vee \ y_1 < y_2 \\ \ell_4 : & \textbf{Critical} \\ \ell_5 : & y_1 := 0 \end{array}\right] \end{array}\right]$$

$$\|$$

$$P_2 :: \left[\begin{array}{l} m_0 : \textbf{loop forever do} \\ \left[\begin{array}{ll} m_1 : & \textbf{Non-Critical} \\ m_2 : & y_2 := y_1 + 1 \\ m_3 : & \textbf{await } y_1 = 0 \ \vee \ y_2 \leq y_1 \\ m_4 : & \textbf{Critical} \\ m_5 : & y_2 := 0 \end{array}\right] \end{array}\right]$$

REG Encoding

The state variables are defined by

$$\begin{array}{ll} \pi_1, \pi_2: & [0..5] \\ Y_1, Y_2: & \textbf{string}[1..N] \textbf{ of boolean} \end{array}$$

We use unary representation to encode an integer n as the string $Y = \underbrace{1,\ldots,1}_{n} 0,\ldots,0$.

We define the formula

$$is\_int(Y): \quad \exists i: \forall j: Y[j] \leftrightarrow (j < i) \quad \text{equivalently} \quad Y \in 1^*0^*$$

The test $y_2 = 0 \;\vee\; y_1 < y_2$ is encoded as

$$\left(\begin{array}{c} Y_2 \\ Y_1 \end{array}\right) \in \left(\begin{array}{c} 0 \\ 0+1 \end{array}\right)^* + \left(\begin{array}{c} 1 \\ 1 \end{array}\right)^* \left(\begin{array}{c} 1 \\ 0 \end{array}\right)^+ \left(\begin{array}{c} 0 \\ 0 \end{array}\right)^*$$

The transition relation for the assignment $y_1 := y_2 + 1$ is encoded as

$$\left(\begin{array}{c} Y_2 \\ Y_1' \end{array}\right) \in \left(\begin{array}{c} 1 \\ 1 \end{array}\right)^* \left(\begin{array}{c} 0 \\ 1 \end{array}\right) \left(\begin{array}{c} 0 \\ 0 \end{array}\right)^*$$

Model checking this transition system terminates and yields the set of reachable global states.

Now to the Main Question

What to do if we have both unbounded control and unbounded data?

Answer: Try to find a joint encoding which will reduce the state description into a single unbounded dimension.

Example: Bakery(N)

N : **integer where** $N > 0$
y : **array** $[1..N]$ **of integer where** $y = 0$

$$\overset{N}{\underset{i=1}{\|}} P[i] :: \left[\begin{array}{l} \ell_0\text{: } \textbf{loop forever do} \\ \quad \left[\begin{array}{ll} \ell_1\text{:} & \textbf{noncritical} \\ \ell_2\text{:} & y[i] := \max(y[1], \ldots, y[N]) + 1 \\ \ell_3\text{:} & \textbf{await } \forall j \neq i : y[j] = 0 \ \vee \ y[i] < y[j] \\ \ell_4\text{:} & \textbf{critical} \\ \ell_5\text{:} & y[i] := 0 \end{array}\right] \end{array}\right]$$

Program BAKERY: the Bakery Algorithm.

This algorithm has an unbounded number of processes and, for each process $P[i]$, an unbounded integer variable. Appears to have two unbounded dimensions.

Linear Encoding

A global state will be encoded as a word of the form

$$(\bigcirc + \bullet)^* \;\Big|\; (\bigcirc + \bullet)^*$$

In this encoding, $\bigcirc$ (a stone) represents a non-critical process, while $\bullet$ represents a critical process. $\Big|$ represents a barrier. All processes to the left of the barrier have their y variable set to 0. Processes to the right of the barrier are ordered, left-to-right according to increasing values of y.

The system is specified by

$$\Theta : \quad \bigcirc^+ \;\Big|$$

$$\rho : \quad \begin{cases} x \bigcirc y \;\Big|\; z & \rightarrow \quad x\, y \;\Big|\; z \bigcirc \\ x \;\Big|\; \bigcirc y & \rightarrow \quad x \;\Big|\; \bullet y \\ x \;\Big|\; \bullet y & \rightarrow \quad \bigcirc x \;\Big|\; y \end{cases}$$

Computing the set of reachable states, we obtain

$$\bigcirc^* \;\Big|\; \bigcirc^* \quad + \quad \bigcirc^* \;\Big|\; \bullet \bigcirc^*$$

We can conclude that Bakery(N) satisfies mutual exclusion.

Note that this encoding traded process identity for ordering imposed by the magnitude of y.

Example: Program Peterson(2)

$$\textbf{local} \quad \begin{array}{ll} y_1, y_2 & : \textbf{booelan } y_1 = y_2 = 0 \\ s & : \{1,2\} \textbf{ initially } s = 1 \end{array}$$

$$P_1 :: \left[\begin{array}{l} \ell_0 : \textbf{loop forever do} \\ \left[\begin{array}{ll} \ell_1 : & \textbf{Non-Critical} \\ \ell_2 : & (y_1, s) := (1, 1) \\ \ell_3 : & \textbf{await } y_2 = 0 \ \vee \ s \neq 1 \\ \ell_4 : & \textbf{Critical} \\ \ell_5 : & y_1 := 0 \end{array} \right] \end{array} \right]$$

$$\|$$

$$P_2 :: \left[\begin{array}{l} m_0 : \textbf{loop forever do} \\ \left[\begin{array}{ll} m_1 : & \textbf{Non-Critical} \\ m_2 : & (y_2, s) := (1, 2) \\ m_3 : & \textbf{await } y_1 = 0 \ \vee \ s \neq 2 \\ m_4 : & \textbf{Critical} \\ m_5 : & y_2 := 0 \end{array} \right] \end{array} \right]$$

This can be generalized to N processes.

Peterson(N)

$$
\begin{array}{l}
\qquad N \quad : \textbf{integer where } N > 0 \\
\qquad y \quad : \textbf{array } [1..N] \textbf{ of boolean where } y = 0 \\
\qquad s \quad : \textbf{array } [1..N] \textbf{ of } [1..N] \\
\overset{N}{\underset{i=1}{\|}} P[i] :: \left[\begin{array}{l}
t : \textbf{integer} \\
\ell_0: \ \textbf{loop forever do} \\
\quad \left[\begin{array}{l}
\ell_1: \ \textbf{noncritical} \\
\ell_2: \ \textbf{for } t := 1 \textbf{ to } N-1 \textbf{ do} \\
\quad \left[\begin{array}{l}
\ell_3: \ (y[i], s[t]) := (t, i) \\
\ell_4: \ \textbf{await } s[t] \neq i \ \vee \ \forall j \neq i : y[j] < i \\
\\
\end{array}\right] \\
\ell_5: \ \textbf{critical} \\
\ell_6: \ y[i] := 0
\end{array}\right]
\end{array}\right]
\end{array}
$$

Program Peterson(N).

Linear Encoding

$$
\begin{array}{ccccccccc}
 & & s[1] & & s[2] & & s[N{-}1] & & \\
\bigcirc\cdots\bigcirc & | & \bigcirc\cdots\bigcirc & | & \cdots & | & \bigcirc\cdots\bigcirc & | & \bigcirc\cdots\bigcirc \\
y=0 & & y=1 & & & & y=N{-}1 & & y=N{-}1
\end{array}
$$

All processes within partition $k = 0, 1, \ldots, N{-}1$ has $y = k$. The leftmost process in partition k is the one signed in $s[k]$. Processes in the rightmost partition are critical.

The system can be specified by

$$
\Theta : \quad \bigcirc^{+}\ |^{+}
$$

$$
\rho : \quad \begin{cases}
x : \{\bigcirc^{*}\}\bigcirc \ |\ y & \rightarrow \quad x\ |\ \bigcirc y \\
x\bigcirc\ |\ y : \{\ |\ ^{*}\} & \rightarrow \quad x\ |\ \bigcirc\ y \\
x\bigcirc\bigcirc\ |\ y & \rightarrow \quad x\bigcirc\ |\ \bigcirc\ y \\
x\bigcirc & \rightarrow \quad \bigcirc x
\end{cases}
$$

Verifying Mutual Exclusion for Peterson(N)

The set of reachable states is given by

$$\bigcirc^* \mid (\bigcirc^+ \mid)^* \bigcirc^* \mid^* \quad + \quad \bigcirc^* \mid (\bigcirc^+ \mid)^* \mid^+ \bigcirc \mid^*$$

To the right of any $\mid\;\mid$ there can be at most one $\bigcirc$.

This has a non-empty intersection with $x \bigcirc \bigcirc$, given by

$$\bigcirc^* \mid (\bigcirc^+ \mid)^* \bigcirc^* \bigcirc \bigcirc$$

So how is Mutual Exclusion Verified?

There are as Many Stones as There are Sticks

In the initial configuration there are $N > 0$ stones and N sticks. If we intersect the language

$$L = \{w \in (\bigcirc + \Big|)^* \quad | \quad \#(\bigcirc, w) \; = \; \#(\Big|, w) \; > \; 0\}$$

with

$$\bigcirc^* \Big| (\bigcirc^+ \Big|)^* \bigcirc^* \bigcirc \bigcirc$$

we get an empty intersection.

Conclusion: Peterson(N) satisfies the property of Mutual Exclusion.

Conclusions

Words (regular languages) can sometimes solve the problems raised by Sticks and Stones.