Yes, Matilda! Abstraction can Replace Deduction, even for Computational Models which are BAD (Büchi Automata with Data)

A. Pnueli
Weizmann Institute of Sciences and
Universite Joseph Fourier

VHS Meeting
Grenoble 28.1.99

Joint work (in progress) with

Y. Kesten
Ben Gurion University

M. Vardi
Rice University

Research carried out within:
the Minerva Center for the Verification of Reactive Systems, Weizmann Institute

A Question and It's Answer

In [KIT'98], we presented the LTL version of Verification by Finitary Abstraction (VFA) as follows:

To prove $\mathcal{K} \models \psi$,

- Abstract $\mathcal{K}$ into a finite-state system $\mathcal{K}^\alpha$ and the specification ψ into a propositional LTL formula ψ^α.
- Model check $\mathcal{K}^\alpha \models \psi^\alpha$.
- Conclude that $\mathcal{K} \models \psi$.

In that talk, we presented a recipe for defining $\mathcal{K}^\alpha$ and ψ^α, such that the method is sound for an arbitrary state-mapping α and conjectured that it is complete, asking the question:

Can Abstraction Replace Deduction?

In a [LICS'99] submission, we confirmed this conjecture, establishing that

Whenever $\mathcal{K} \models \psi$, there exists a state mapping α such that $\mathcal{K}^\alpha \models \psi^\alpha$.

Another Question

Moshe Vardi raised the question:

> Can we do the same within an automata-theoretic framework?

To answer this question, we had first to invent an automata-theoretic deductive proof system so that we will have something to replace.

The answer to this question is presented in this talk.

The BAD Computational Model

For our computational model, we take a Büchi Automaton with Data (BAD) $\mathcal{A} = \langle V, Q, \Theta, q^0, \rho, \mathcal{F} \rangle$ consisting of:

- V – A finite set of data variables. A data state s is an interpretation of V. Σ – the set of all data states.
- Q – A set of locations (control states).
- Θ – An initial condition. A satisfiable assertion that characterizes the initial data states.
- $q^0 \in Q$ – An initial location.
- ρ – A transition relation. For each pair of locations q_i and q_j, $\rho_{i,j}(V, V')$, identifies the condition under which the automaton can move from q_i to q_j, and the new values assumed by the data variables in the next state.
- $\mathcal{F} \subseteq Q$ – The set of accepting locations.

BAD Runs and Computations

A (complete) state of a BAD $\mathcal{A} = \langle V, Q, \Theta, q^0, \rho, \mathcal{F}\rangle$ is a pair $\langle q, s\rangle$ consisting of a location $q \in Q$ and a data state $s \in \Sigma_{\mathcal{A}}$. We say that the state $\langle q_j, s_j\rangle$ is an $\mathcal{A}$-successor of the state $\langle q_i, s_i\rangle$ if $\langle s_i, s_j\rangle \models \rho_{i,j}(V, V')$.

A computation of $\mathcal{A}$ is an infinite sequence of states

$$\sigma : \langle q_0, s_0\rangle, \langle q_1, s_1\rangle, \langle q_2, s_2\rangle, ...,$$

satisfying the following requirements:

- Initiality: $\langle q_0, s_0\rangle$ is initial, i.e., $q_0 = q^0$ and $s_0 \models \Theta_{\mathcal{A}}$.
- Consecution: For each $j = 0, 1, ...,$ $\langle q_{j+1}, s_{j+1}\rangle$ is an $\mathcal{A}$-successor of $\langle q_j, s_j\rangle$.
- Acceptance: The sequence σ contains infinitely many occurrences of accepting locations.

An infinite state-sequence satisfying only the two first requirements is called a run of $\mathcal{A}$.

Property Verification in the BAD Framework

To establish that LTL property ψ is valid over system S ($S \models \psi$), one

- Constructs a BAD $\mathcal{A}_S$, representing system S,
- Constructs a BAD $\mathcal{A}_{\neg\psi}$, characterizing all sequences violating property ψ,
- Forms the (synchronous) parallel composition $\mathcal{C} = \mathcal{A}_S \parallel\mid \mathcal{A}_{\neg\psi}$.
- Establishes that $\mathcal{C}$ is empty, i.e., has no computations.

This confirms that S has no computations violating ψ. Hence all S-computations satisfy ψ — ψ is S-valid.

I. Constructing $\mathcal{A}_S$ for System S

Consider the following program ANY-Y:

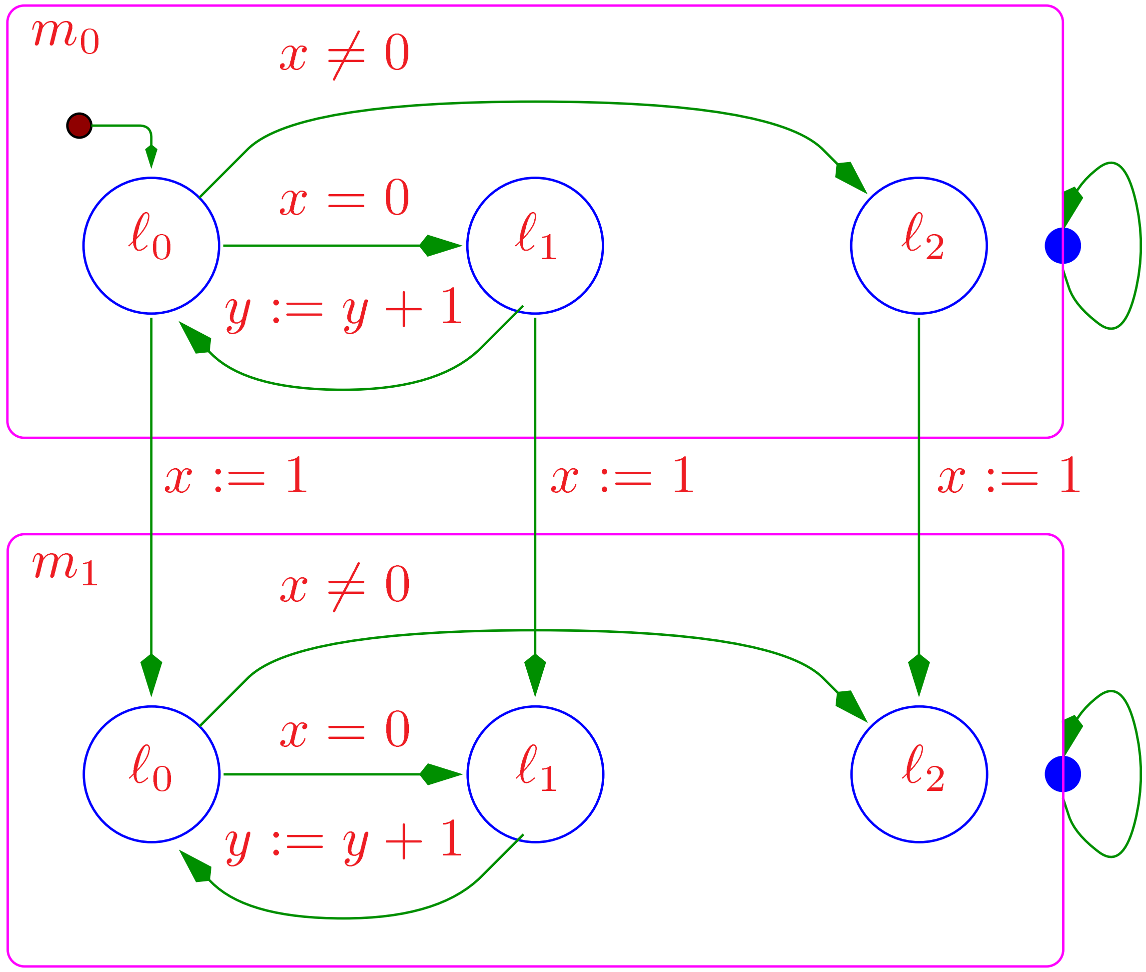
$$x,\ y: \textbf{ integer initially } x = y = 0$$

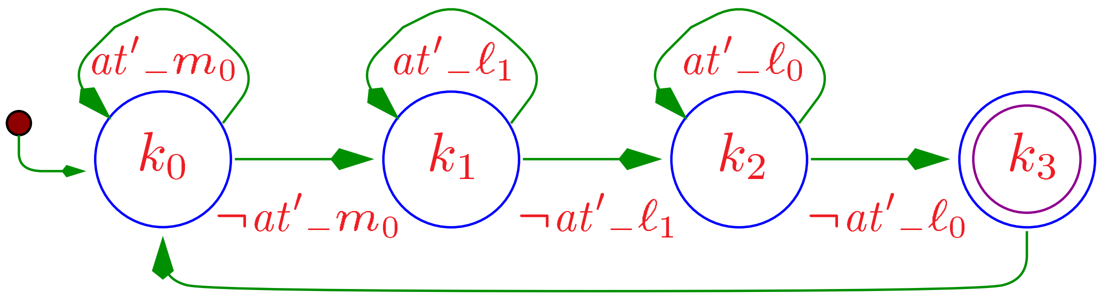
$$\left[\begin{array}{ll} \ell_0: & \textbf{while } x = 0 \textbf{ do} \\ & \quad [\ell_1: \ y := y + 1] \\ \ell_2: & \end{array}\right] \quad \| \quad \left[\begin{array}{ll} m_0: & x := 1 \\ m_1: & \end{array}\right]$$

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

The Corresponding BAD $\mathcal{A}_{\text{ANY-Y}}$

initially $x = y = 0$

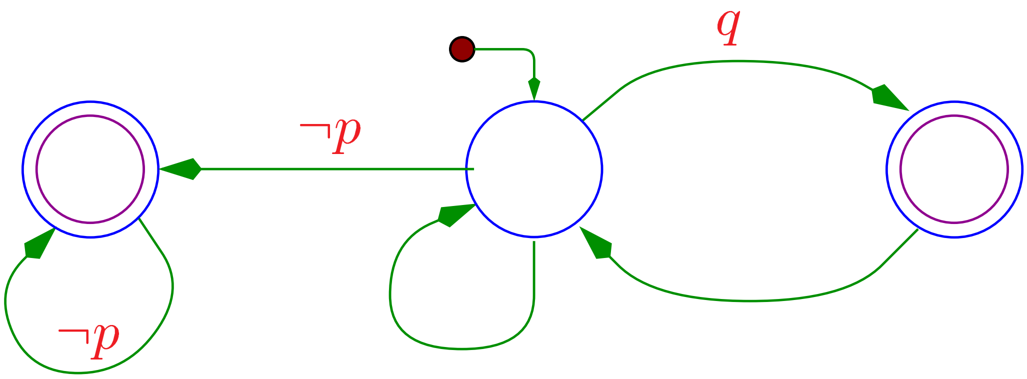




II. Constructing $\mathcal{A}_\psi$ for Property ψ

Illustrate on property:

$$\square \diamond p \quad \rightarrow \quad \square \diamond q$$



III. Forming $\mathcal{C} = \mathcal{A}_S \mathbin{|||} \mathcal{A}_{\neg\psi}$

The (synchronous) parallel composition of BADs
$\mathcal{A}_1 = \langle V_1, Q_1, \Theta_1, q_1^0, \rho_1, \mathcal{F}_1 \rangle$ and
$\mathcal{A}_2 = \langle V_2, Q_2, \Theta_2, q_2^0, \rho_2, \mathcal{F}_2 \rangle$, denoted $\mathcal{A}_1 \mathbin{|||} \mathcal{A}_2$,
is given by $\mathcal{A} = \langle V, Q, \Theta, q^0, \rho, \{\mathcal{G}_1, \mathcal{G}_2\} \rangle$, where

$$
\begin{array}{lcccc}
V & = & V_1 & \cup & V_2 \\
Q & = & Q_1 & \times & Q_2 \\
\Theta & = & \Theta_1 & \wedge & \Theta_2 \\
q^0 & = & & (q_1^0, q_2^0) & \\
\rho & = & \rho_1 & \wedge & \rho_2 \\
\mathcal{G}_1 & = & \mathcal{F}_1 & \times & Q_2 \\
\mathcal{G}_2 & = & Q_1 & \times & \mathcal{F}_2
\end{array}
$$

Note that the resulting automaton is a multi-Büchi automaton. However, a standard transformation brings it back to the BAD form.

IV. Establishing that $\mathcal{C} = \mathcal{A}_S \,|||\, \mathcal{A}_{\neg\psi}$ is Empty

Let $\mathcal{A} = \langle V, Q, \Theta, q^0, \rho, \mathcal{F}\rangle$ be a BAD such that $Q = \{q^0{=}q_1, \ldots, q_n\}$. The following single proof rule is sound and complete for establishing that $\mathcal{A}$ is empty.

Rule WELL

For assertions $\varphi_1, \ldots, \varphi_n$,
a well-founded domain $(\mathcal{W}, \succ)$,
and ranking functions $\delta_1, \ldots, \delta_n : \Sigma \mapsto \mathcal{W}$

W1. $\Theta \quad \rightarrow \quad \varphi_1$

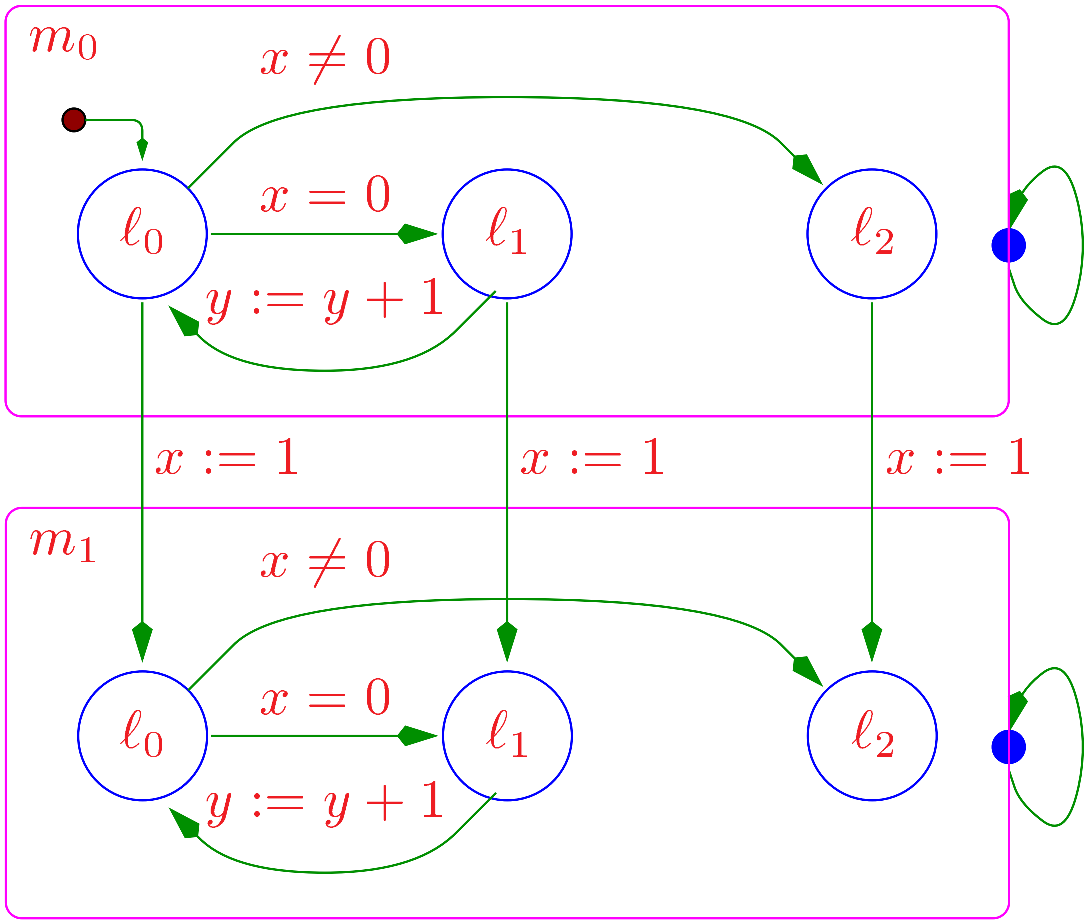
For $i, j \in \{1, \ldots, n\}$,
W2. $\rho_{i,j} \wedge \varphi_i \quad \rightarrow \quad {\varphi_j}' \wedge \delta_i \succeq {\delta_j}'$

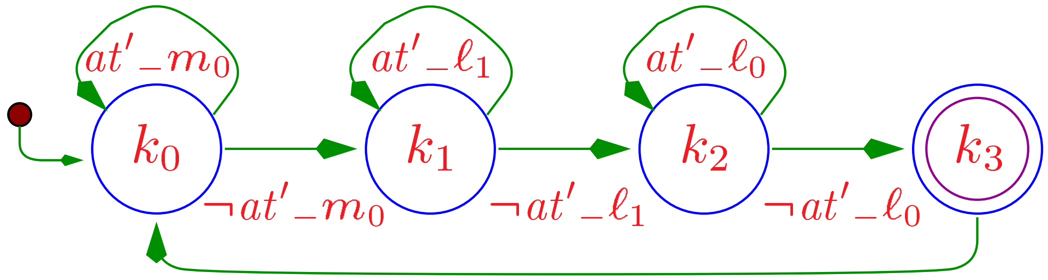
In addition, if $q_j \in \mathcal{F}$, then also,
W3. $\rho_{i,j} \wedge \varphi_i \quad \rightarrow \quad {\varphi_j}' \wedge \delta_i \succ {\delta_j}'$

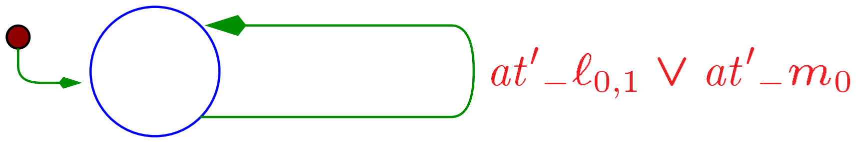
$\mathcal{A}$ is empty

Example: Verifying Termination of ANY-Y

initially $x = y = 0$







Invoking Rule WELL

To establish that $\mathcal{A}_{\text{ANY-Y}} \,|||\, \mathcal{A}_{\square\,(at\_\ell_{0,1} \vee at\_m_0)}$ is empty, we take:

$$(\mathcal{W}, \succ) \quad = \quad (\{0, \ldots, 3\},\ >)$$

and

$$\begin{array}{lcl@{\qquad\qquad}lcl}
\varphi_{k_0} & : & at_\ell_{0..2}\ \wedge\ at_m_0\ \wedge\ x = 0 & \delta_{k_0} & : & 3 \\
\varphi_{k_1} & : & at_\ell_{0..2}\ \wedge\ at_m_1\ \wedge\ x = 1 & \delta_{k_1} & : & 2 \\
\varphi_{k_2} & : & at_\ell_{0,2}\ \wedge\ at_m_1\ \wedge\ x = 1 & \delta_{k_2} & : & 1 \\
\varphi_{k_3} & : & at_\ell_{2}\ \wedge\ at_m_1\ \wedge\ x = 1 & \delta_{k_2} & : & 0
\end{array}$$

Part II: Replacing by Abstraction

No sooner have we constructed an automata theoretic deductive proof system, that we proceed to deconstruct it.

Verification by Finitary Abstraction

Based on the notion of abstract interpretation [CC77].

Let Σ denote the set of concrete states. Let $\alpha : \Sigma \mapsto \Sigma_A$ be a mapping of concrete states into abstract states, syntactically specified by equations of the form $V_A = \alpha(V)$.

Lifting the mapping α from states to assertions (sets of states), we define for assertion p,

$$\alpha^+(p): \quad \exists V \; (V_A = \alpha(V) \; \wedge \; p(V))$$

Note that an abstract state S_A satisfies $\alpha^+(p)$ iff some concrete state s satisfying p is α-mapped to S_A.

In a similar way, we define for a transition relation ρ,

$$\alpha^{++}(\rho) \;=\; \exists V, V' \left(\begin{array}{cc} & V_A = \alpha(V) \;\wedge\; V'_A = a(V') \\ \wedge & \rho(V, V') \end{array} \right)$$

Abstracting a BAD Automaton

For a BAD $\mathcal{A} = \langle V, Q, \Theta, q^0, \rho, \mathcal{F}\rangle$ and an abstraction mapping $V_A = \alpha(V)$, we define the α-abstraction of $\mathcal{A}$ as:

$$\mathcal{A}^\alpha = \langle V_A, Q, \alpha^+(\Theta), q^0, \alpha^{++}(\rho), \mathcal{F}\rangle$$

Claim 1. *If $\sigma \ : \ \langle q_0, s_0\rangle, \langle q_1, s_1\rangle, \langle q_2, s_2\rangle, \ldots,$ is a computation of $\mathcal{A}$, then*

$$\alpha(\sigma) : \langle q_0, \alpha(s_0)\rangle, \langle q_1, \alpha(s_1)\rangle, \langle q_2, \alpha(s_2)\rangle, \ldots,$$

is a computation of $\mathcal{A}^\alpha$.

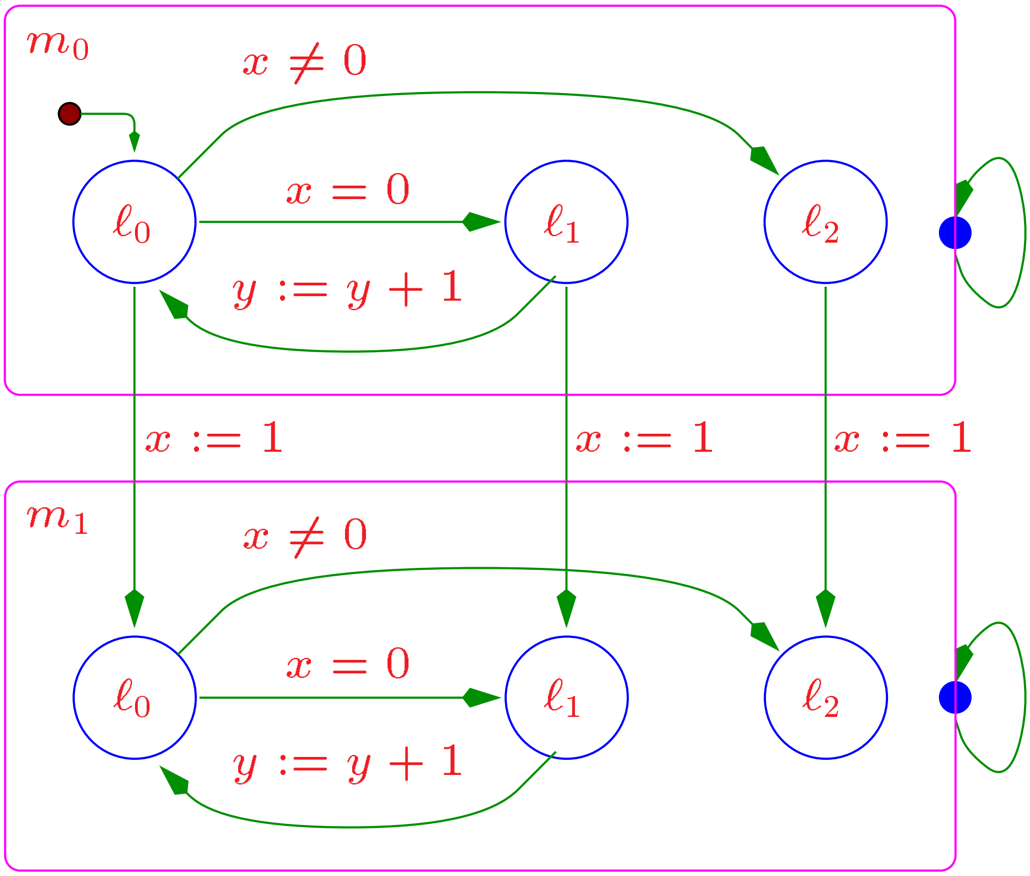
That is, every computation of $\mathcal{A}$ is α-mapped into a computation of $\mathcal{A}^\alpha$ but, usually, $\mathcal{A}^\alpha$ has additional computations which are not the α-image of any computation of $\mathcal{A}$.

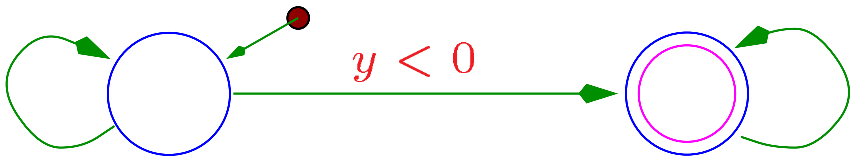
Corollary 1. *If $\mathcal{A}^\alpha$ is empty then so is $\mathcal{A}$.*

This establishes that the VFA method is sound. Soundness means that the abstraction preserves counter examples.

Example: ANY-Y satisfies $\square\,(y \geq 0)$

initially $x = y = 0$





Abstract by

$$Y\colon \{neg, zero, pos\} \quad = \quad \alpha(y) \quad \text{is}$$

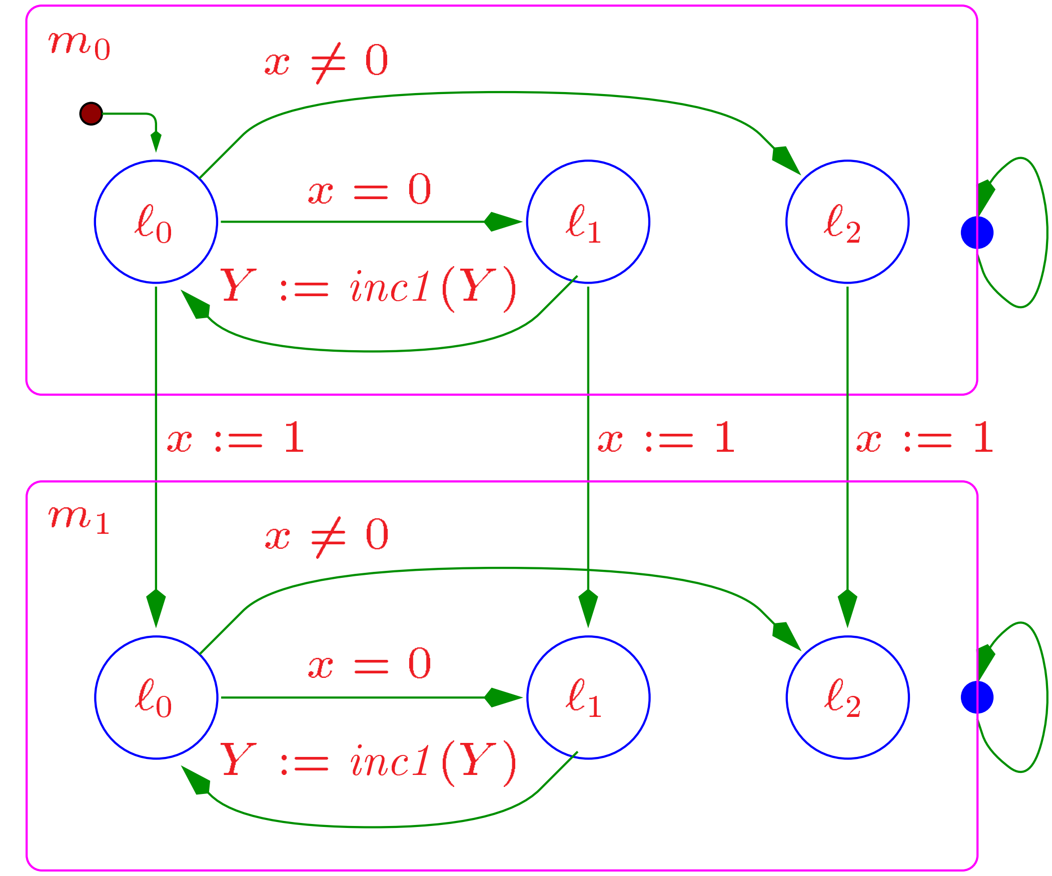
case

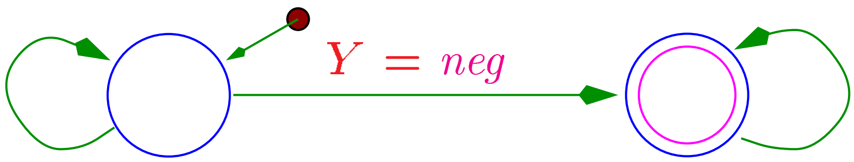
$$\begin{array}{lcl} y < 0 & : & neg \\ y = 0 & : & zero \\ y > 0 & : & pos \end{array}$$

esac

The Abstracted Version

x: **integer**; **initially** $x = 0$;
Y: $\{neg,\ zero,\ pos\}$; **initially** $Y = zero$





where

$$inc1(Y) = \left(\begin{array}{lll} \textbf{case} & & \\ \quad Y = neg & : & \{neg, zero\} \\ \quad \textbf{otherwise} & : & pos \\ \textbf{esac} & & \end{array} \right)$$

Systematic Construction of Abstractions

Let $p_1, p_2, \ldots, p_k$ be the set of all atomic formulas referring to the data (non-control) variables appearing within conditions in the program P and within the temporal formula ψ.

Following [BBM95], define abstract boolean variables $B_{p_1}, B_{p_2}, \ldots, B_{p_k}$, one for each atomic data formula. The abstraction mapping α is defined by

$$\alpha: \quad \{B_{p_1} = p_1, B_{p_2} = p_2, \ldots, B_{p_k} = p_k\}$$

Example: Program BAKERY-2

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

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

$$\|$$

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

The temporal properties for program BAKERY-2 are

$$\begin{array}{lll} \psi_{exc} & : & \Box \neg(at\_\ell_4 \ \wedge \ at\_m_4) \\ \psi_{acc} & : & \Box \ (at\_\ell_2 \ \rightarrow \ \Diamond \ at\_\ell_4), \end{array}$$

Abstracting Program BAKERY-2

Define abstract variables $B_{y_1=0}$, $B_{y_2=0}$, and $B_{y_1<y_2}$.

$$\textbf{local}\quad B_{y_1=0}, B_{y_2=0}, B_{y_1<y_2} \;:\; \textbf{boolean}$$
$$\textbf{initially}\ B_{y_1=0} = B_{y_2=0} = 1, B_{y_1<y_2} = 0$$

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

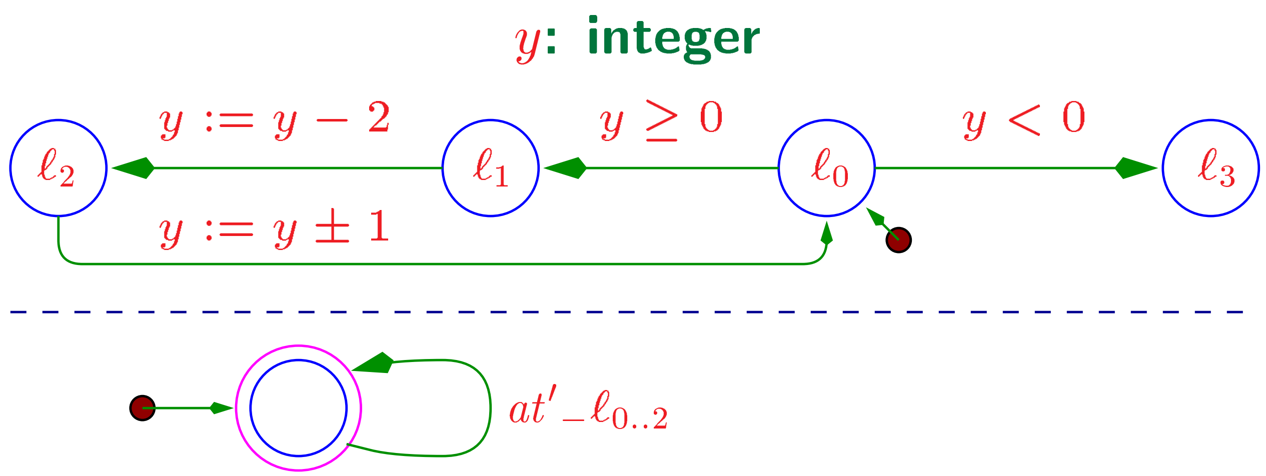
$$\|$$

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

The abstracted properties can now be model-checked.

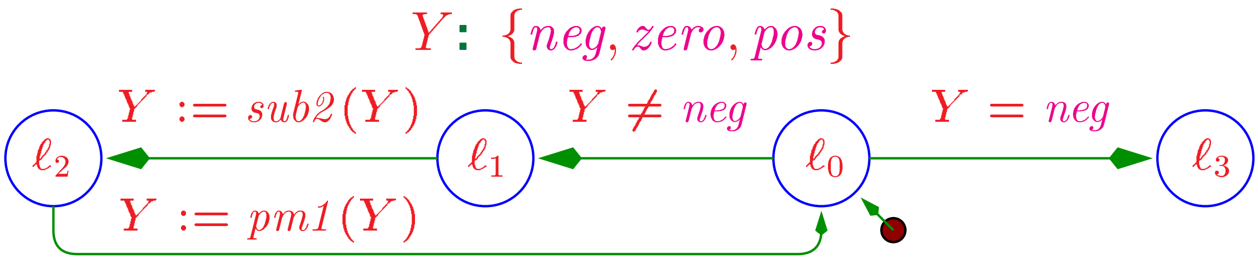
Completeness

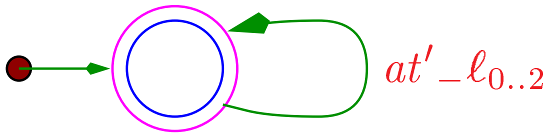
In general, it is not sufficient to apply an abstraction mapping. Typically, liveness properties are not preserved. Consider, for example, the following program TERM with a tester for termination:



The Abstracted Version

Applying the abstraction α : **integer** $\mapsto \{neg, zero, pos\}$ to program TERM, we obtain:





where,

$$sub2(Y) \quad = \quad \left(\begin{array}{ll} \textbf{if } Y = pos & \textbf{then } \{neg, zero, pos\} \\ & \textbf{else} \quad neg \end{array} \right)$$

$$pm1(Y) \quad = \quad \left(\begin{array}{l} \textbf{case} \\ \quad \begin{array}{lcl} Y = pos & : & \{zero, pos\} \\ Y = zero & : & \{neg, pos\} \\ Y = neg & : & \{neg, zero\} \end{array} \\ \textbf{esac} \end{array} \right)$$

Unfortunately, this program does not terminate, as shown by the counter-example computation

$$(\langle \ell_0, pos\rangle, \langle \ell_1, pos\rangle, \langle \ell_2, pos\rangle)^\omega$$

Augmentation Saves the Day

To preserve all properties, we may need augmentation of the original system.

Let us be guided by the deductive proof of termination for program TERM. This is based on the auxiliary assertion

$$\varphi: \quad y + 3 \geq 3 \cdot at\_\ell_1 + at\_\ell_2,$$

and the ranking function

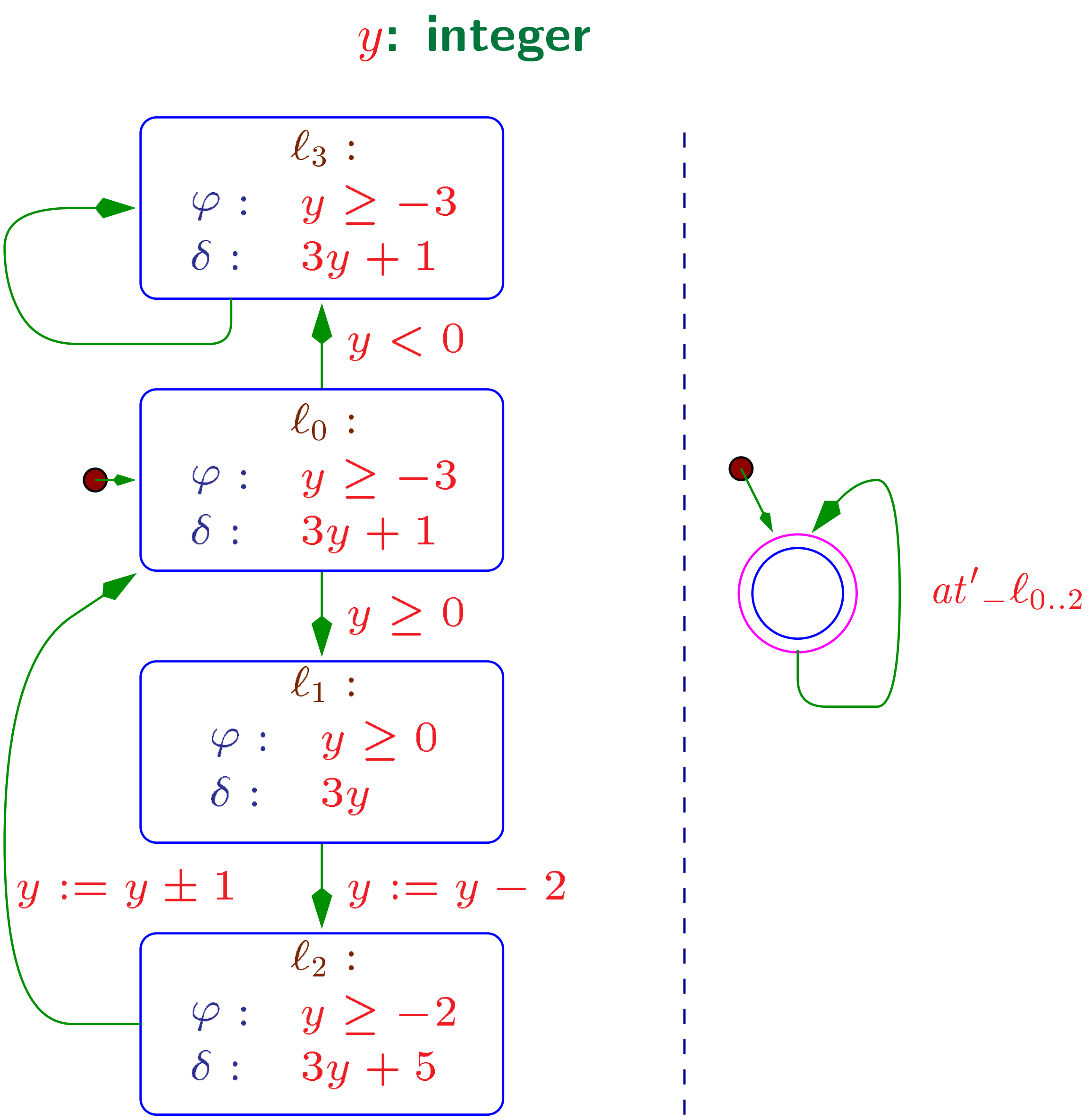
$$\delta: \quad \left(\begin{array}{lll} \textbf{case} & & \\ \quad at\_\ell_0 & : & 3y+1 \\ \quad at\_\ell_2 & : & 3y+5 \\ \quad \textbf{otherwise} & : & 3y \\ \textbf{esac} & & \end{array}\right)$$

also expressible as

$$\delta: \quad 3y + at\_\ell_0 + 5 \cdot at\_\ell_2$$

Graphically

Annotating the locations with their versions of φ and δ, we obtain

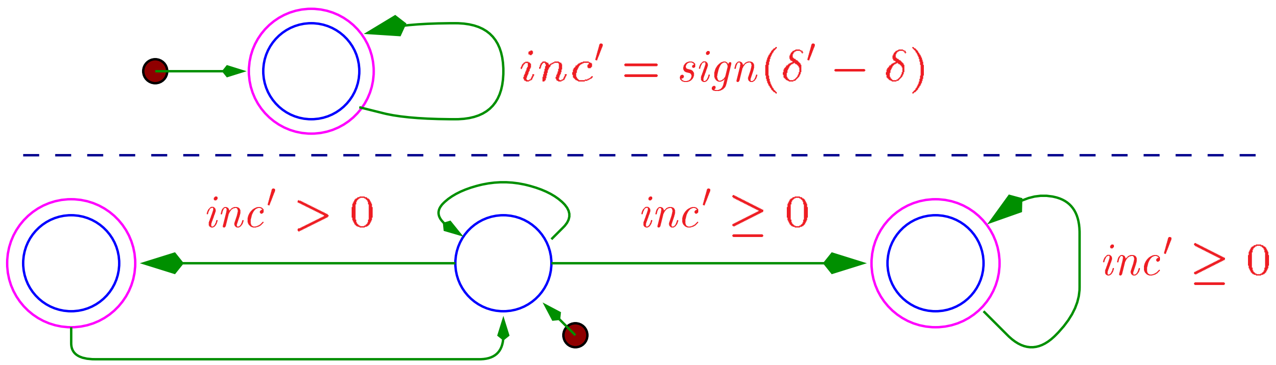


Safe Augmentation by a Ranking Monitor

A BAD M is said to be a safe augmentation of a BAD $\mathcal{A}$ if $\mathcal{A}$ and $\mathcal{A} \mathrel{|||} M$ have identical sets of $V_{\mathcal{A}}$-observations. Obviously, if M is a safe augmentation of $\mathcal{A}$ and ψ is an alleged property of $\mathcal{A}$, then ψ is valid over $\mathcal{A}$ iff ψ is valid over $\mathcal{A} \mathrel{|||} M$.

Claim 2. *If δ is a function mapping into a well-founded domain $(\mathcal{W}, \succ)$, then the following ranking monitor M_δ is a safe augmentation of any BAD $\mathcal{A}$:*

inc: $\{-1, 0, +1\}$



Note that the expression $sign(\delta' \succ \delta)$ is equivalent to

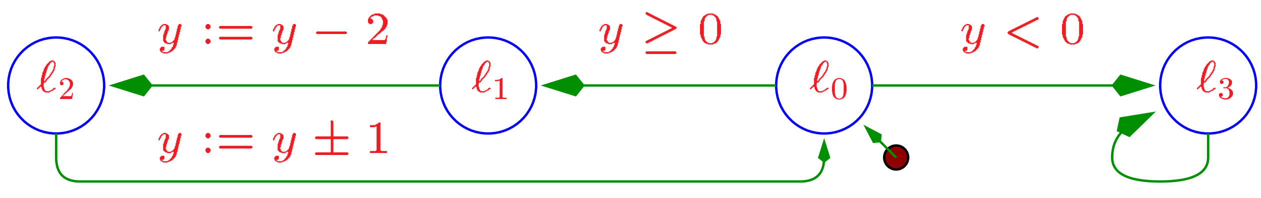
if $\delta' \succ \delta$ **then** $+1$ **else if** $\delta' \prec \delta$ **then** -1 **else** 0,

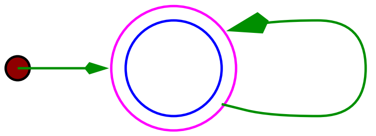
and that the second component of M_δ requires

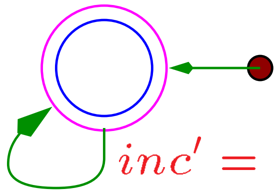
$$\Box \Diamond (inc < 0) \quad \rightarrow \quad \Box \Diamond (inc > 0).$$

Augmenting Program TERM

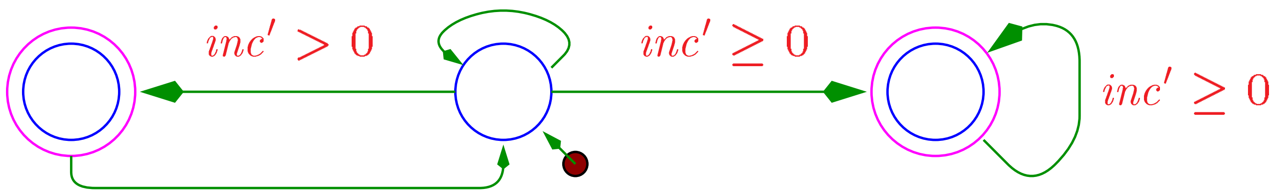
inc: $\{-1, 0, +1\}$







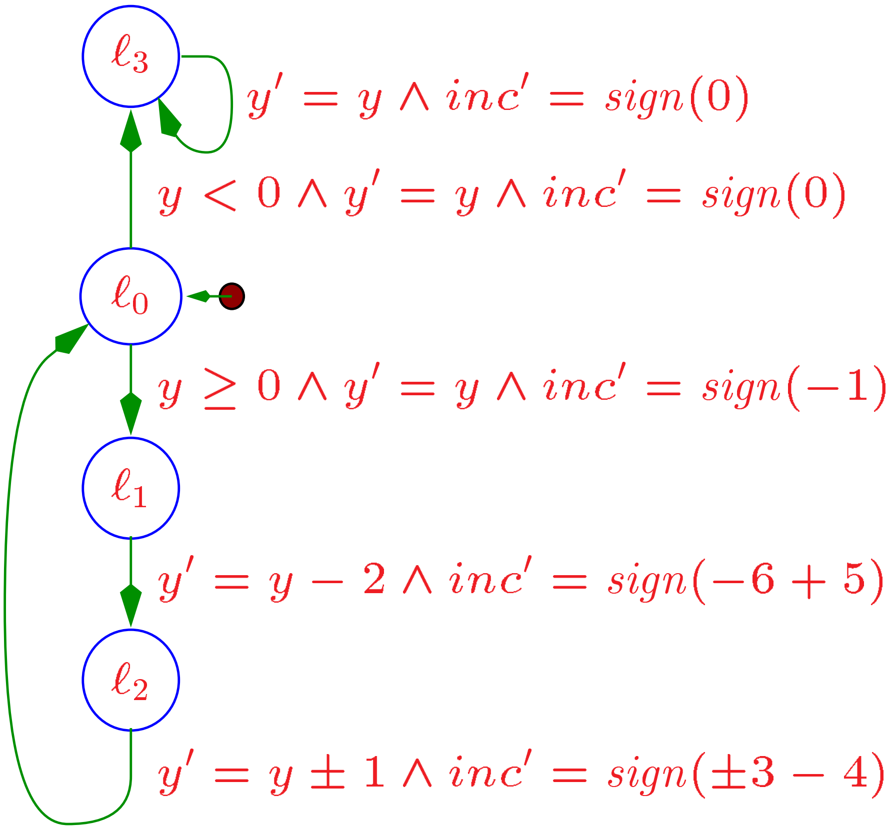
$$inc' = sign(\underbrace{3y' + at'\_\ell_0 + 5 \cdot at'\_\ell_2}_{\delta'} - \underbrace{3y + at\_\ell_0 + 5 \cdot at\_\ell_2}_{\delta})$$

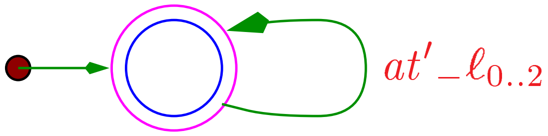


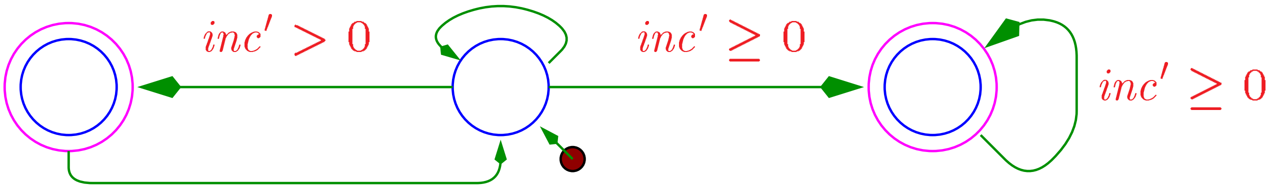
Multiplying the 3rd component into the first one, we obtain:

Multiplied Augmented TERM

inc: $\{-1, 0, +1\}$



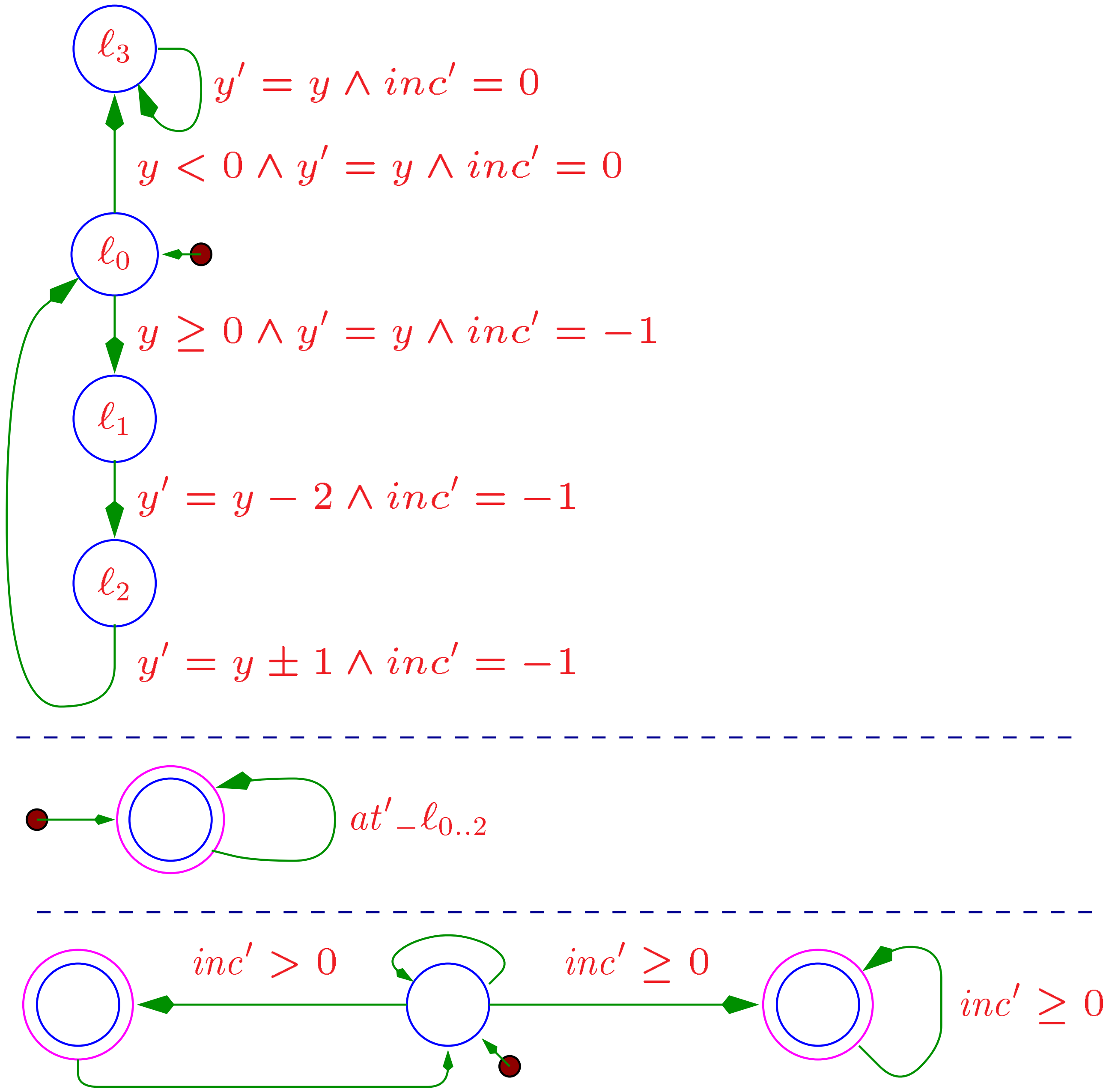




This can be simplified to:

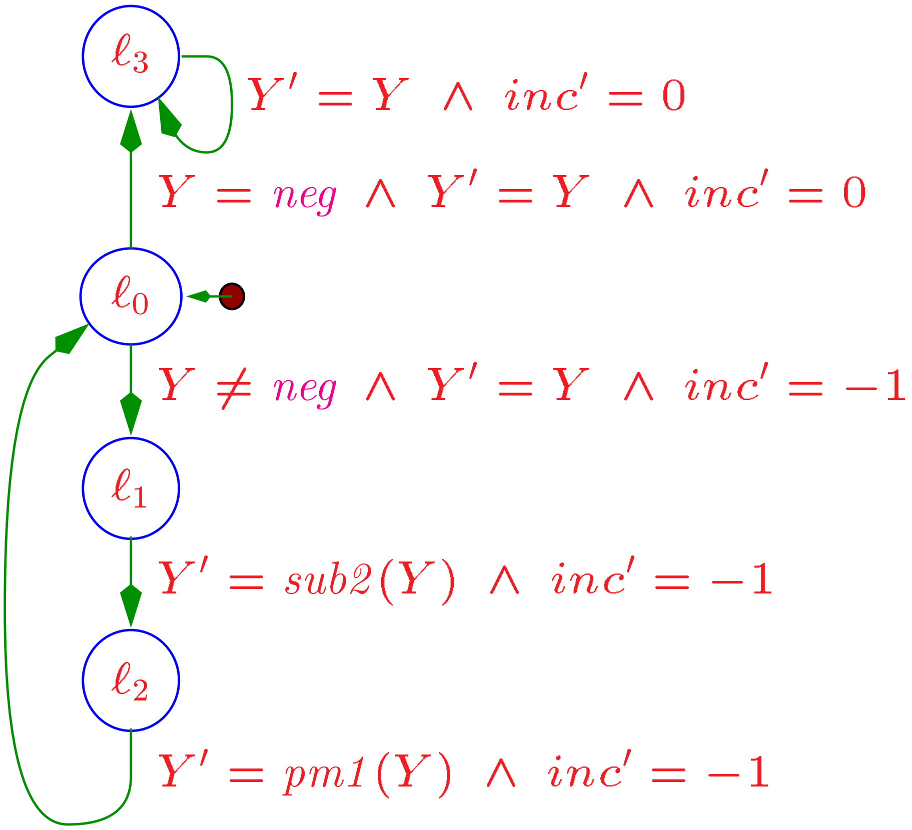
Simplified Augmented TERM

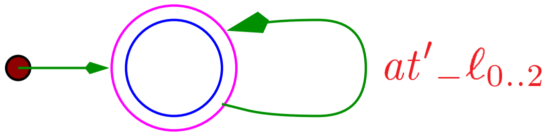
$$inc\colon \{-1, 0, +1\}$$

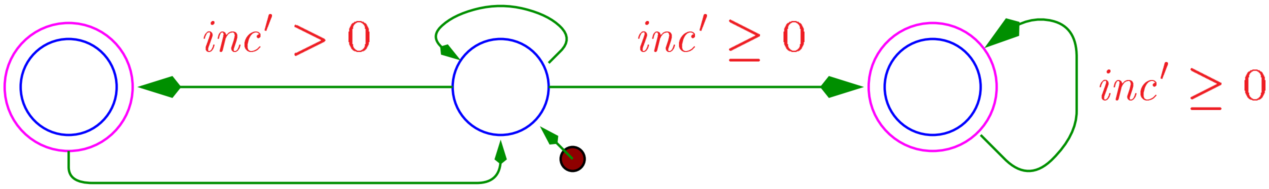


Abstracted Augmented TERM

$$\begin{array}{ll} Y\textbf{:} & \{neg, zero, pos\} \\ inc\textbf{:} & \{-1, 0, +1\} \end{array}$$







which has no computations!!!

Completeness Achieved by Augmentation

A dual way to abstract assertions is given by

$$\alpha^{-}(p): \quad \forall V \ (V_A = \alpha(V) \rightarrow p(V))$$

An abstract state S_A satisfies $\alpha^{-}(p)$ iff every concrete state α-mapped to S_A satisfies p.

A mapping α is said to be precise with respect to an assertion p if $\alpha^{+}(p) \sim \alpha^{-}(p)$. A sufficient condition for precision is that V_A contains a boolean variable B with the α-mapping $B = p$.

Claim 3. *If the emptiness of $\mathcal{A}$ is provable by rule* WELL *using assertions $\varphi_1, \ldots, \varphi_n$ and ranking function δ, and the mapping α is precise with respect to $\varphi_1, \ldots, \varphi_n$, then $(\mathcal{A} \parallel\!| M_\delta)^\alpha$ is also empty.*

Corollary 2. *The method of verification by finitary abstraction is (relatively) complete, provided we are allowed to use safe augmentations.*