FASTER TEMPORAL REASONING FOR INFINITE-STATE PROGRAMS

Heidy Khlaaf[1] Byron Cook[1,2] Nir Piterman[3]

[1]University College London
[2]Microsoft Research
[3]University of Leicester

BRIEF OVERVIEW

- A new symbolic model checking procedure for CTL verification of infinite-state programs.
- Counterexample-guided precondition synthesis strategy to compute location(PC) specific preconditions.
- Existing tools not scalable. We propose:
  - Compositional strategy via exploiting the natural decomposition of the program's state space.
  - Performance improvement and scalability!

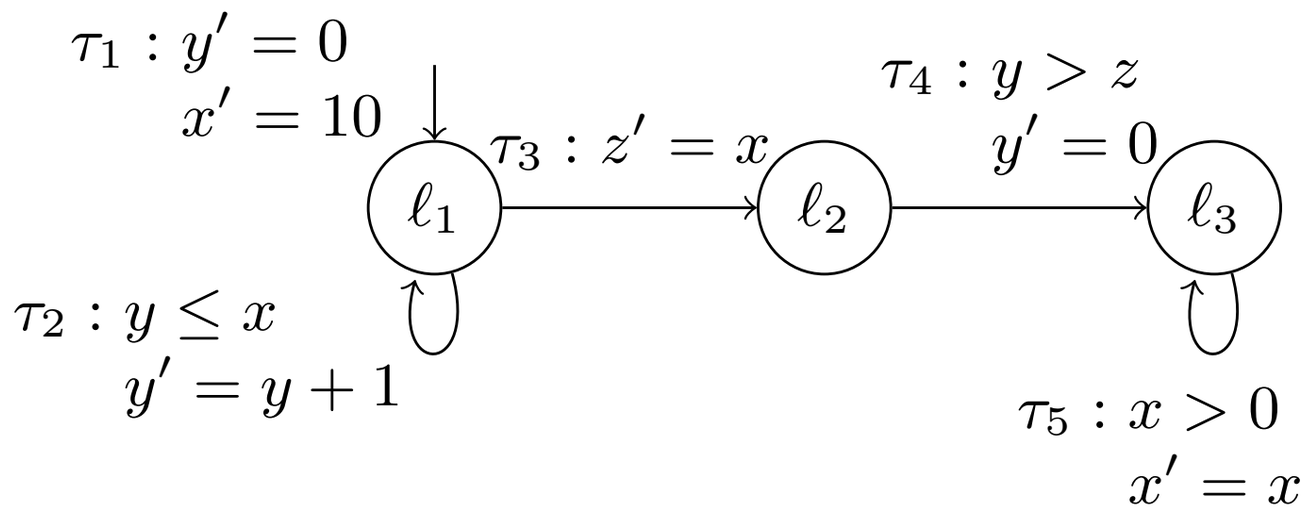
CTL - A REFRESHER

- Temporal logic reasoning about sets of states.
- Implemented via formula structure.
- Reasoning about non-deterministic (branching) programs.
- Termination - AFAX false, Non-Termination - EGEX true, AG - Safety
- These operators can be nested to generate complex liveness and safety properties.
- Used to uncover bugs in device drivers, operating systems, servers, etc.

CTL - A REFRESHER

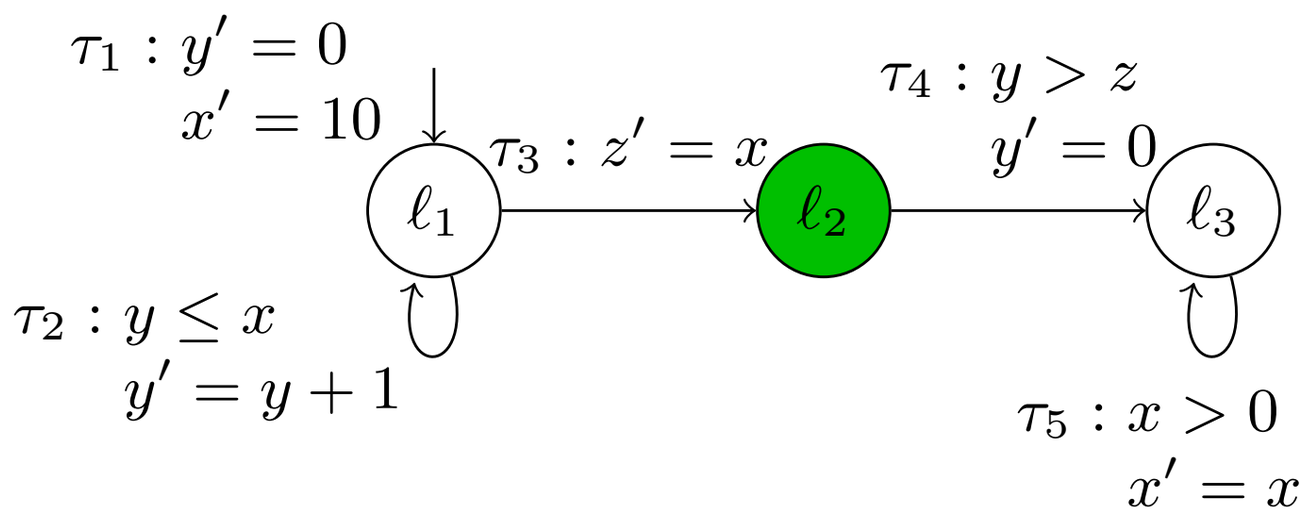
- A φ – All: φ has to hold on all paths starting from all initial states.
- E φ – Exists: there exists at least one path starting from all initial states where φ holds.
- G φ – Globally: φ has to hold on the all states along a path.
- F φ – Finally: φ eventually has to hold.
- Other operators include X φ – Next, φ_1 U φ_2 – Until, φ_1 W φ_2 – Weak until.

INTUITION



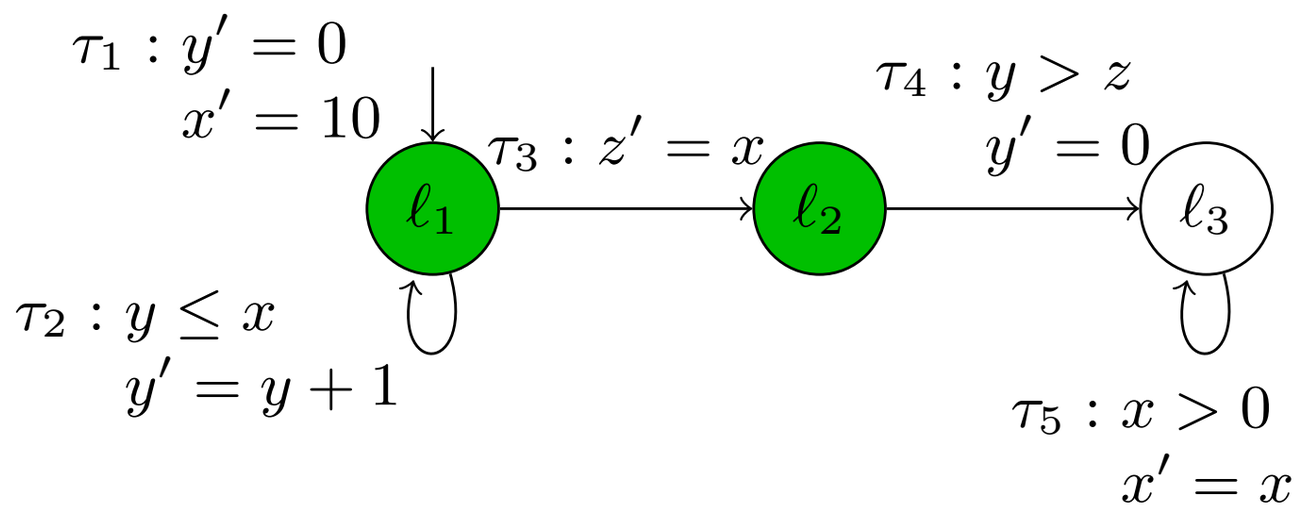
- Verify EF y > z

INTUITION



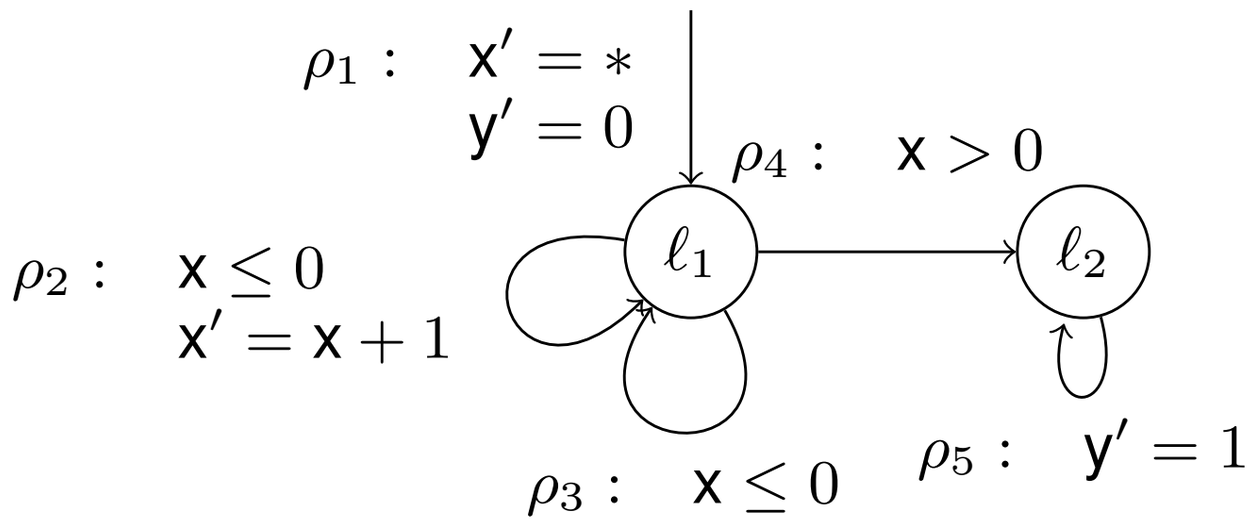
- The set of states satisfying EF y > z *before* a program command is very often the same as the set of states respecting EF y > z *after* the command.

INTUITION



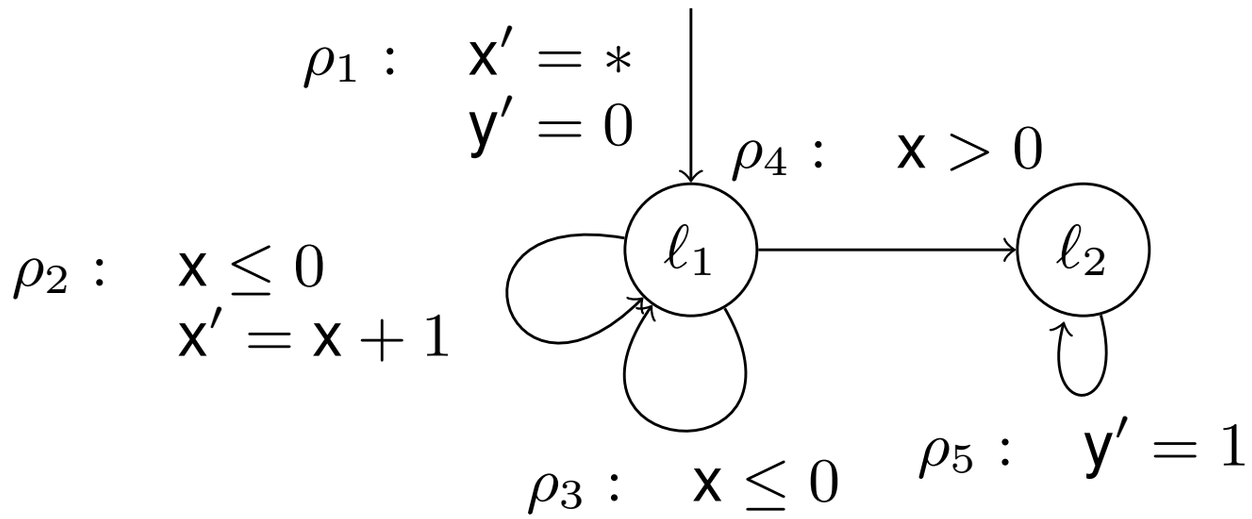
- We can infer whether a command is likely to affect the truth of EF y > z.
  - From one counterexample, we use precondition synthesis to infer the pre-image of all locations within a counterexample.

EXAMPLE (PT. 1)



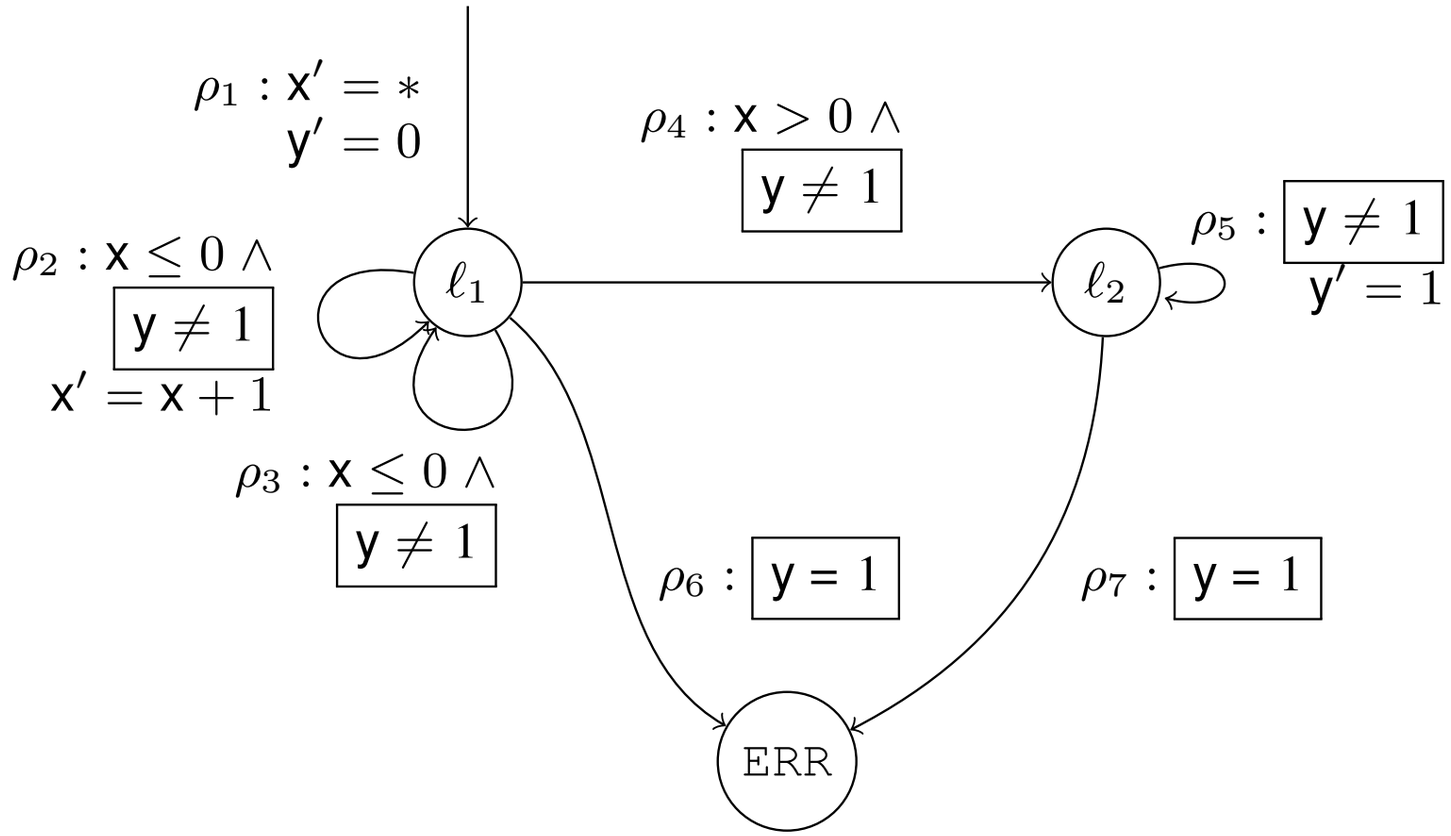
- Prove the CTL property AGEF $y = 1$.

EXAMPLE (PT. I)



- Recurse over the structure of the CTL formula.
- Find $\wp$ such that AG$\wp$ holds, and $\wp \models$ EF y = 1.
- $\wp\langle\varphi\rangle$ takes the form $\bigwedge_{l_i}$ (pc = $l_i \Rightarrow \wp\langle l_i, \varphi\rangle$).

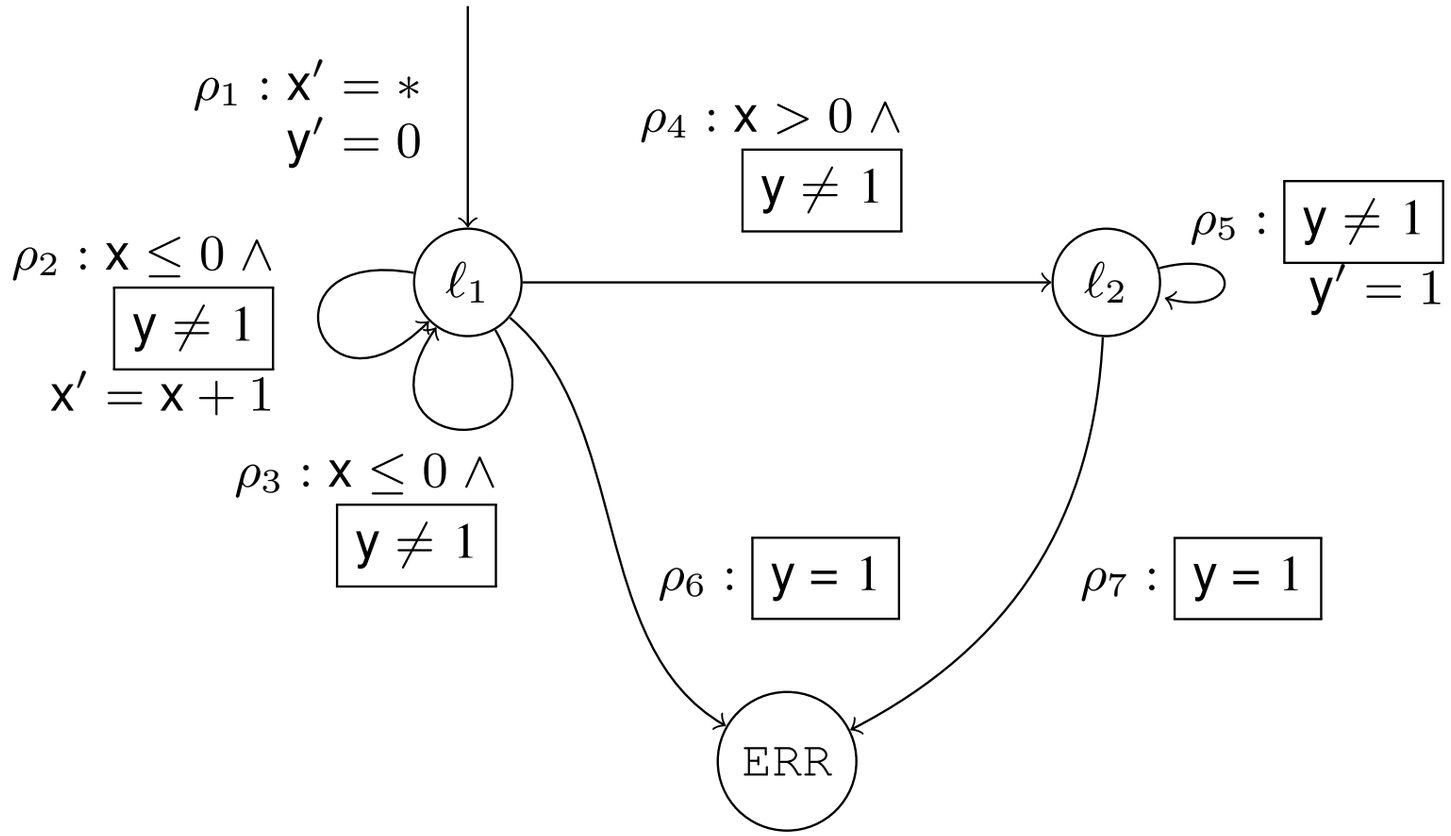
EXAMPLE (PT. I)



- Verify the universal dual of the existential property and seek a set of counterexamples to serve as witnesses.

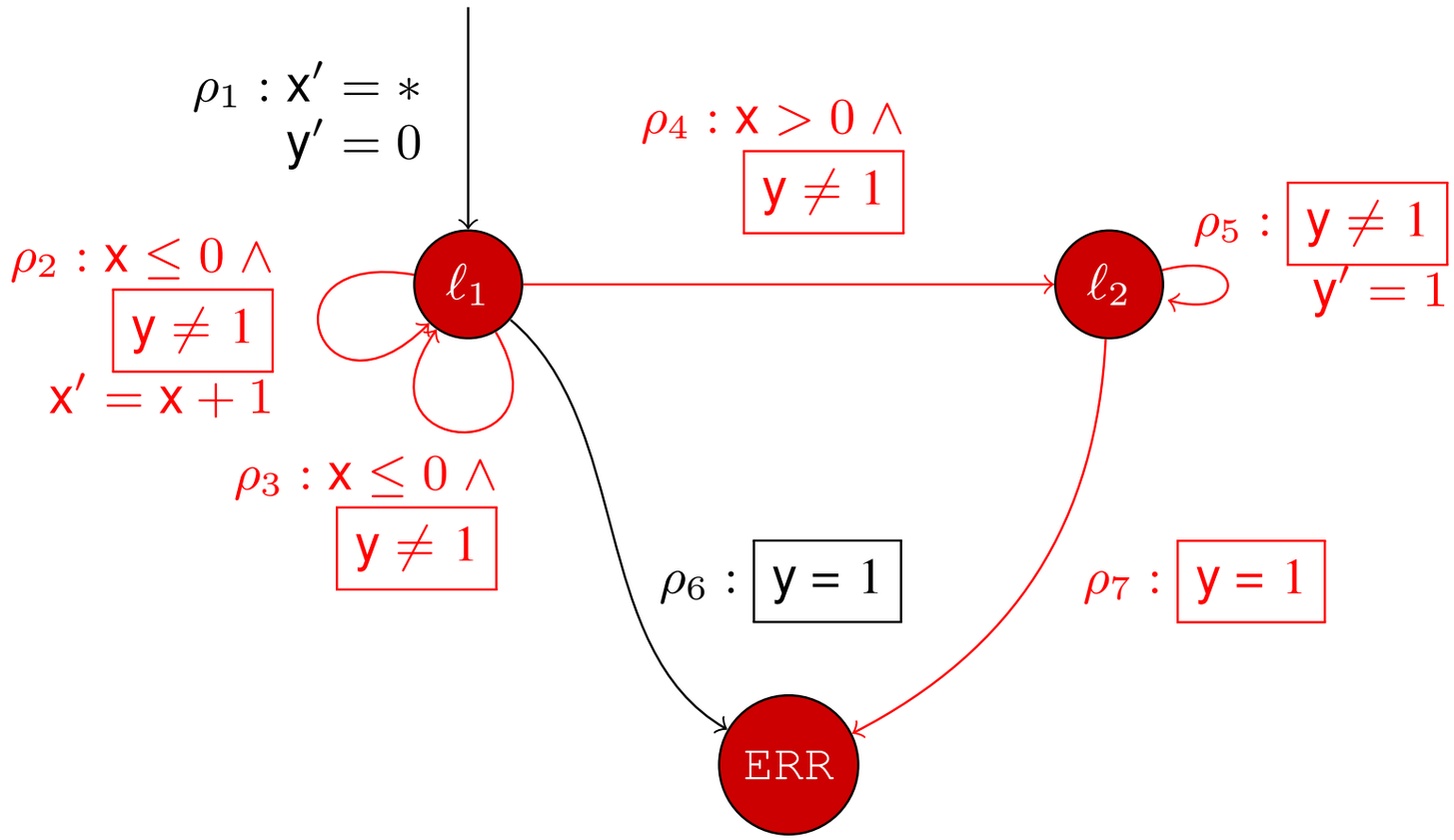
- Transform the program for the property φ = EF y = 1 using its dual AG y ≠ 1.

EXAMPLE (PT. I)



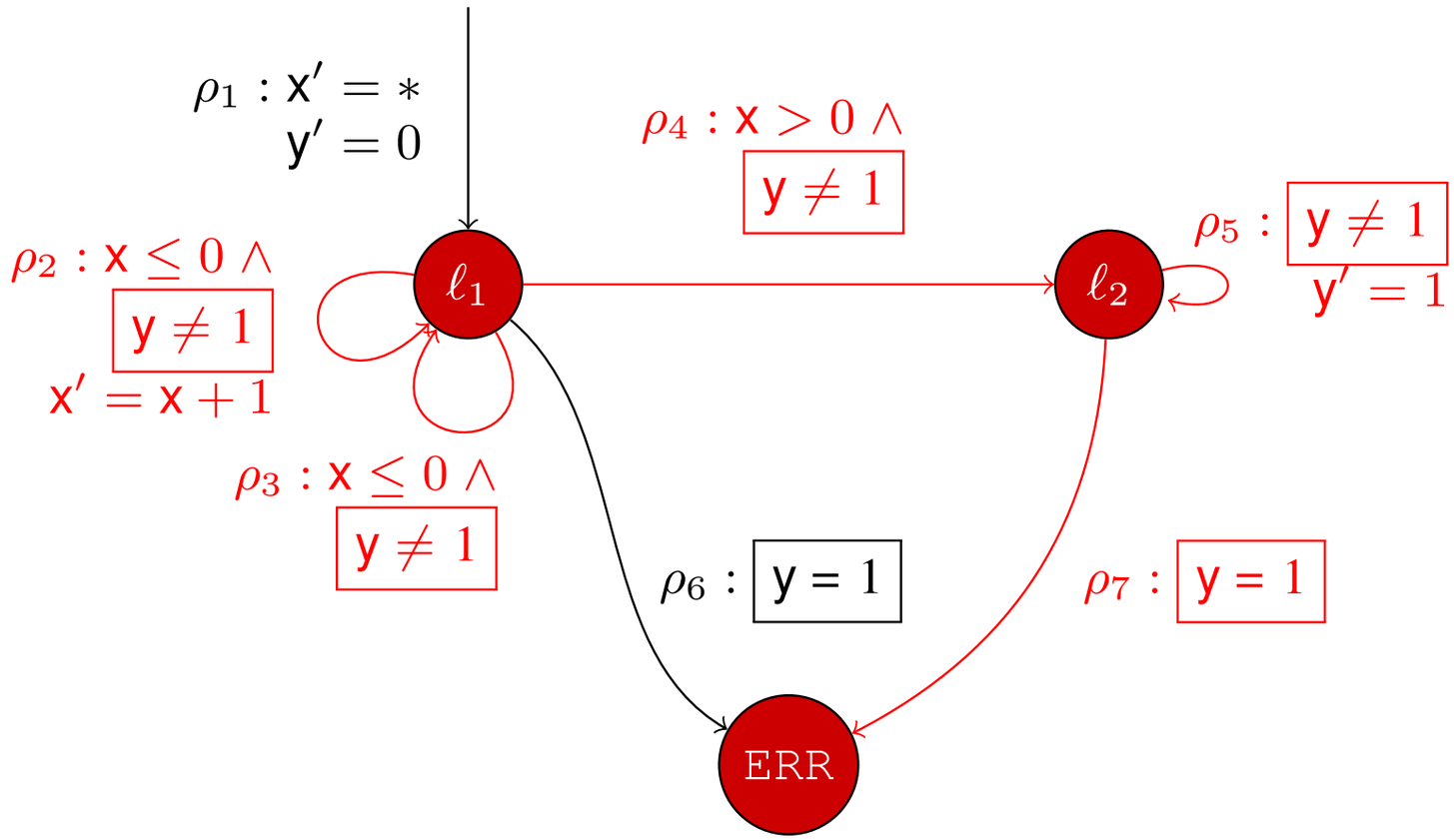
- Initially $\wp\langle\varphi\rangle$ = false as only failures to proving AG y ≠ 1 imply that there exists a witness such that EF y = 1.

EXAMPLE (PT. I)



- Use a safety prover to check reachability of ERR, and begin from I_1.
- For every reachable location $l \in L$ in CEX_1, we compute a pre-image using the suffix of CEX_1 from l onwards.

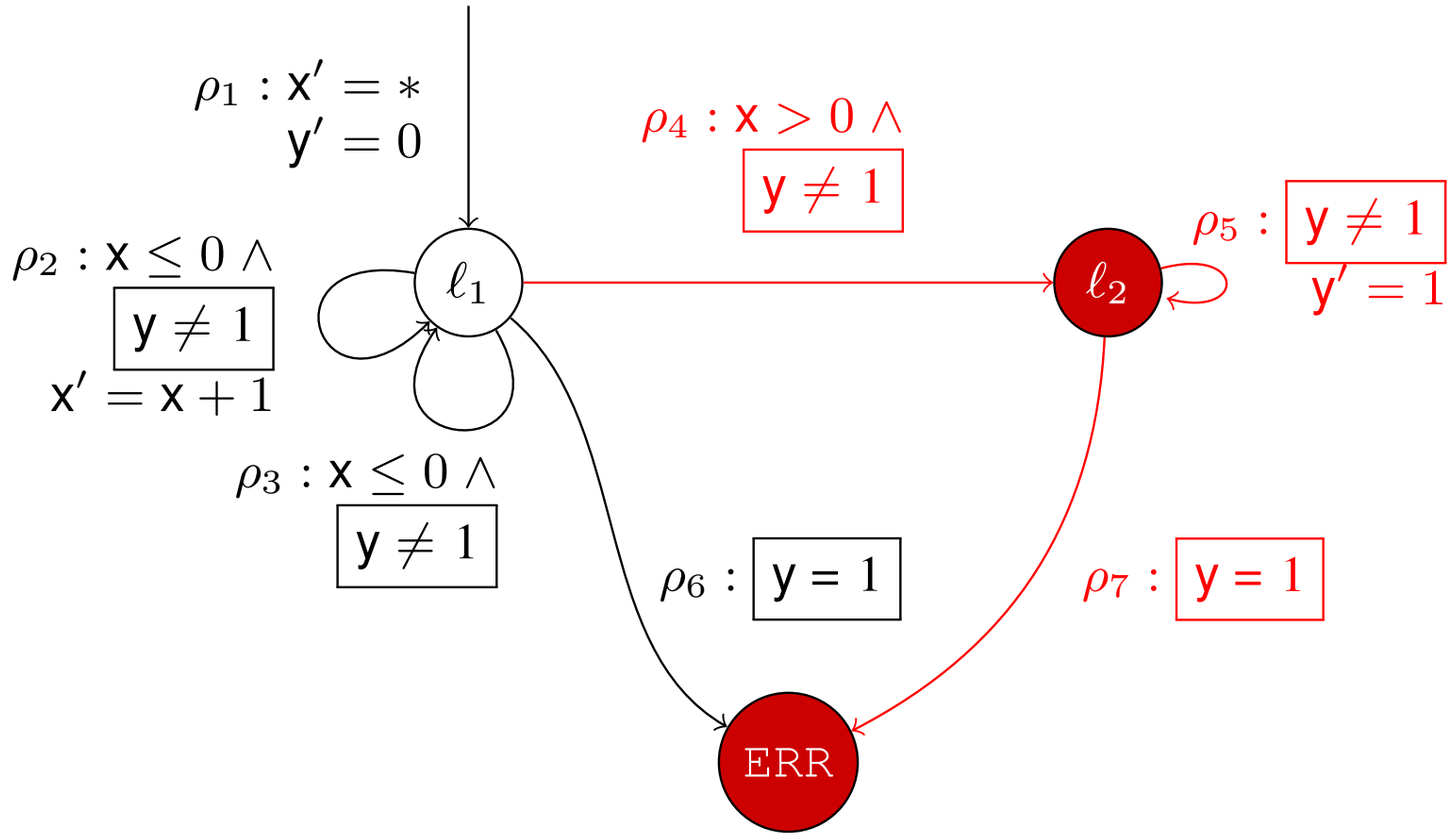
EXAMPLE (PT. I)



- $pre(CEX_l) = y = 0$.
- When a new counterexample is discovered, we refine

  $\wp\langle l, \varphi\rangle$ resulting in $\wp\langle l, \varphi\rangle = \bigvee_{n \in N} pre(CEX_n)$

EXAMPLE (PT. I)



- pre($CEX_{I'}$) = $x > 0$.

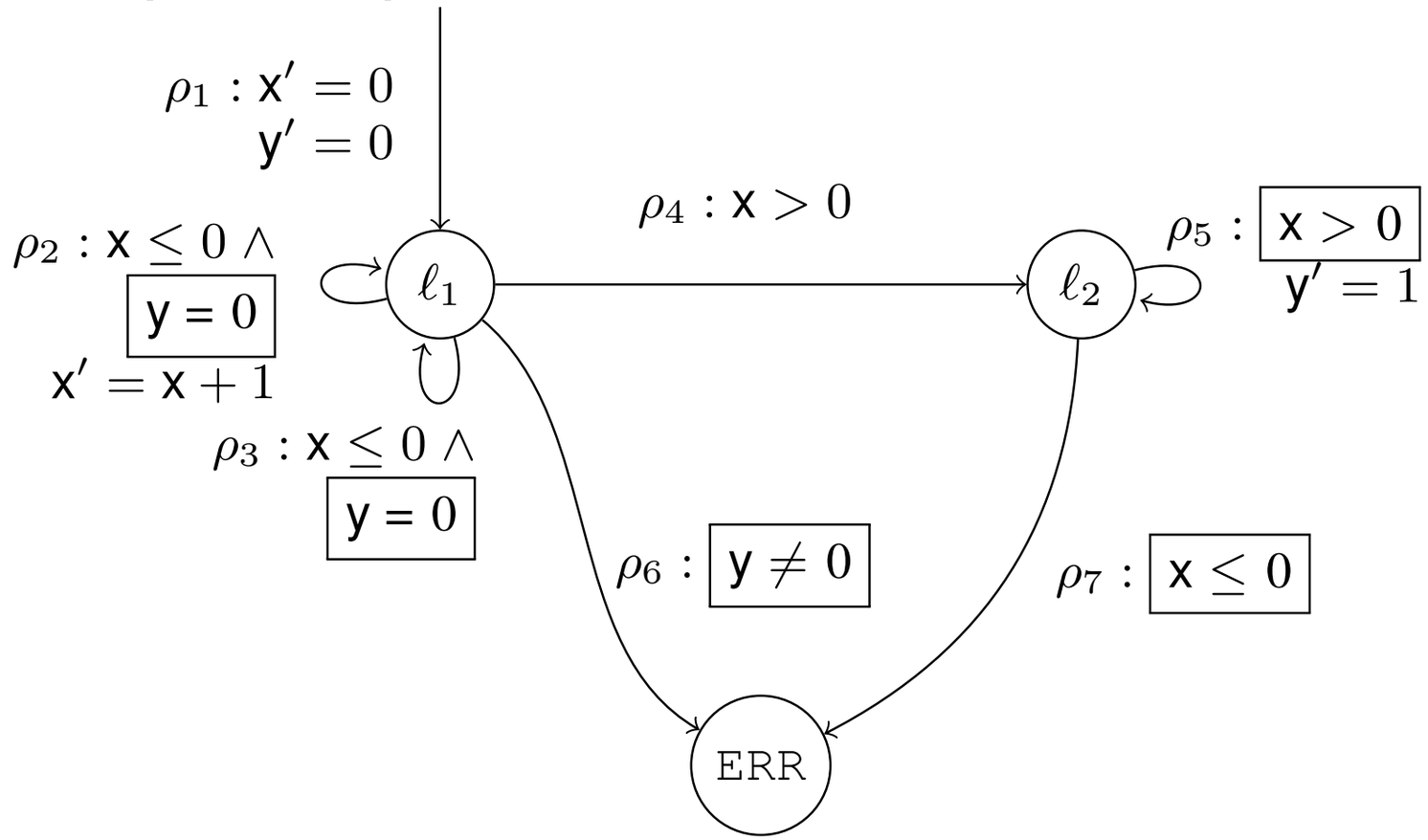
- Computed a refinement for I_2 from a counterexample generated for I_1. No need to verify I_2 independently!

EXAMPLE (PT. 1)

- Ensure EF y = 1 satisfies all initial states: Rule out CEX_1 by adding ¬pre(CEX_1) to each transition from l_1 to the error state.
- Re-run the safety checker.
- No more counterexamples are generated and all locations covered:

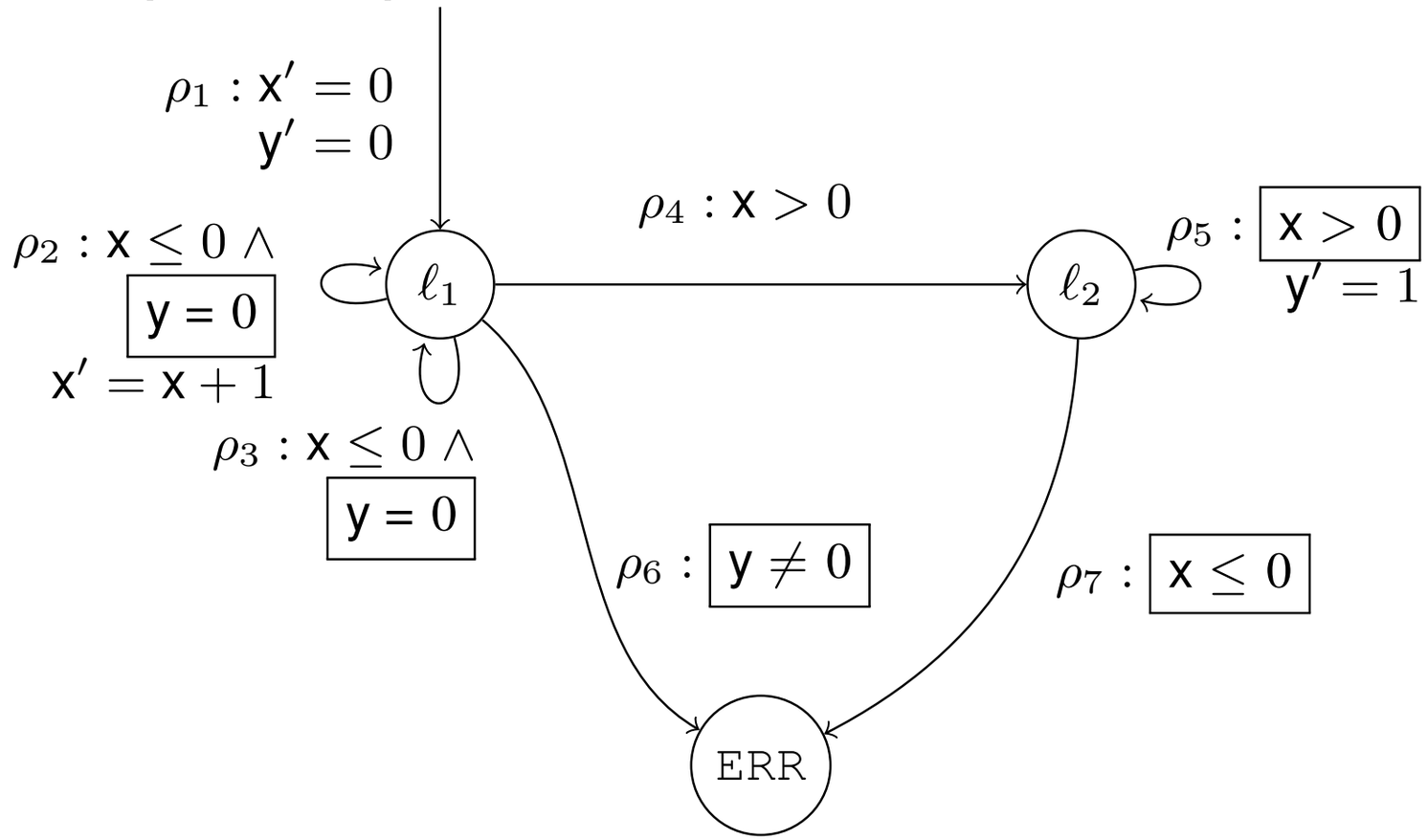
  $\wp\langle EF\ y = 1\rangle = (pc{=}l_1 \Rightarrow y{=}0) \wedge (pc{=}l_2 \Rightarrow x{>}0)$.

EXAMPLE (PT. 2)



- Modify φ = AGEF y = 1 by using $\wp\langle$EF y = 1$\rangle$:
  - φ' = AG ((pc = $l_1 \Rightarrow$y=0)$\wedge$(pc=$l_2 \Rightarrow$x>0)).

EXAMPLE (PT. 2)



- $\wp\langle \mathsf{I}, \varphi \rangle$ resulting in $\wp\langle \mathsf{I}, \varphi \rangle = \bigwedge_{n \in N} \neg \mathsf{pre}(\mathsf{CEX}_n)$

- Universal: the initial precondition $\wp\langle \varphi \rangle$ = true. No counterexamples are generated thus $\wp\langle$ AGEF y = 1$\rangle$= true!

RECAP

- Partition CTL formula preconditions by program location:
  - $\wp\langle\varphi\rangle$ takes the form $\bigwedge_{l_i}(pc = l_i \Rightarrow \wp\langle l_i, \varphi\rangle)$.
- Universal location preconditions:
  - $\wp\langle l, \varphi\rangle = \bigwedge_{n\in N} \neg pre(CEX_n)$
- Existential location preconditions:
  - $\wp\langle l, \varphi\rangle = \bigvee_{n\in N} pre(CEX_n)$

EXPERIMENTS

- Built as an extension to the open source project T2
  - Source Code: http://research.microsoft.com/en-us/projects/t2/.
- Input: C files converted to t2 file format + CTL specification.
- Compared our tool to :

[1] T. A. Beyene, C. Popeea, and A. Rybalchenko, "Solving existentially quantified horn clauses," in *CAV'13*. Springer, 2013

[2] B. Cook and E. Koskinen, "Reasoning about nondeterminism in programs," in *PLDI'13*. ACM, 2013.

EXPERIMENTS

LoC	Property	T2	[1]	[2]
1050	AG(b = 1 —> AF(u = 0))	67.3	T/O	T/O
1050	EG(b = 1 —> EF(u = 0))	36.2	T/O	T/O
370	AG(a = 1 —> EF(r = 1))	6.8	35.45	90.0
370	EF(a = 1 && AG (r ≠1))	4.7	T/O	T/O
370	EG(io ≠ 1) && EG(ret ≠ 1)	13.5	T/O	7.6
370	AG(io ≠ 1) || AG(ret ≠ 1)	8.0	0.1	T/O
90	AGEF w = 1	2.0	0.7	T/O
90	EFAG w ≠ 1	2.0	0.1	T/O
90	EFEG w ≠ 1	0.1	0.1	35.2

LIMITATIONS

- Divergence can occur due to infinitely many counterexamples.
  - Take pre-image **α** of a PC, quantify out all variables that are updated proceeding a program location.
- Can lead to unsoundness due to over-approximation of the set of states for existential path quantifiers.
- Check that the precondition is sound *e.g.*
  that $\wp_1 \Rightarrow \text{EG } \wp_2$, we can use SMT based strategies to double check the small lemma on initial locations.

SUMMARY

- A new symbolic model checking procedure for CTL verification of infinite-state programs.

- Use a counterexample-guided precondition synthesis strategy to compute location-specific preconditions.

- Reduces the amount of irrelevant reasoning traditionally performed as several preconditions for each location can be computed simultaneously.

- Performance improvement and scalability!

BACKGROUND

- **P** = (L, E, Vars),
- Each edge $\tau = (l, \rho, l')$ in E, where $l, l' \in L$ and ρ is a condition, specifies possible transitions in the program.
- **T** = (S, R)
  - **S** = L × (Vars → Vals)
  - **R** ⊆ S × S
- A **cut-point** is a set C such that C ⊆ L and every cycle in the program's graph contains at least one cut-point.
- **Pre-images**: For a path $\pi = (l_0, \rho_0, l'_0), (l_1, \rho_1, l'_1) n \ldots, (l_n, \rho_n, l'_n)$, we compute a pre-image for every possible suffix of π:
  - $pre_{n+1} = S$ and $pre_i = pre[(l_i, \rho_i, l'_i), \ldots, (l_n, \rho_n, l'_n)]$ as the set of states such that $pre_i = \{s \mid \exists s' \in pre_{i+1} \text{ s.t. } ((l_i, s), (l'_i, s')) \models \rho_i\}$.

FINDING TEMPORAL PRECONDITIONS

- Recurse over the structure of the given CTL formula.
- For each CTL sub-formula φ we find a precondition $\wp\langle\varphi\rangle$ that ensures its satisfaction.
- Each sub-formula φ is then replaced with $\wp\langle\varphi\rangle$ within the original formula.
  - Note: It is only necessary to handle formulas of nesting depth 1.
- To utilize sequential locality of a counterexample's control-flow graph:
  - $\wp\langle\varphi\rangle$ takes the form $\bigwedge_{l_i} (pc = l_i \Rightarrow \wp\langle l_i, \varphi\rangle)$.

FINDING TEMPORAL PRECONDITIONS

- For a universal CTL sub-property:
  - A precondition $\wp\langle l, \varphi\rangle$ for a program location l is initialized to true.
  - When a new counterexample is discovered, we refine

    $\wp\langle l, \varphi\rangle$ resulting in $\wp\langle l, \varphi\rangle = \bigwedge_{n\in N} \neg pre(CEX_n)$
- For an existential CTL sub-property:
  - Verify the universal dual of the existential property and seek a set of counterexamples to serve as witnesses.
  - A precondition $\wp\langle l, \varphi\rangle$ for a program location l is initialized to false.
  - When a new counterexample is discovered, we refine

    $\wp\langle l, \varphi\rangle$ resulting in $\wp\langle l, \varphi\rangle = \bigvee_{n\in N} pre(CEX_n)$