ON AUTOMATION OF CTL* VERIFICATION FOR INFINITE-STATE SYSTEMS

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

[1] University College London
[2] University of Leicester

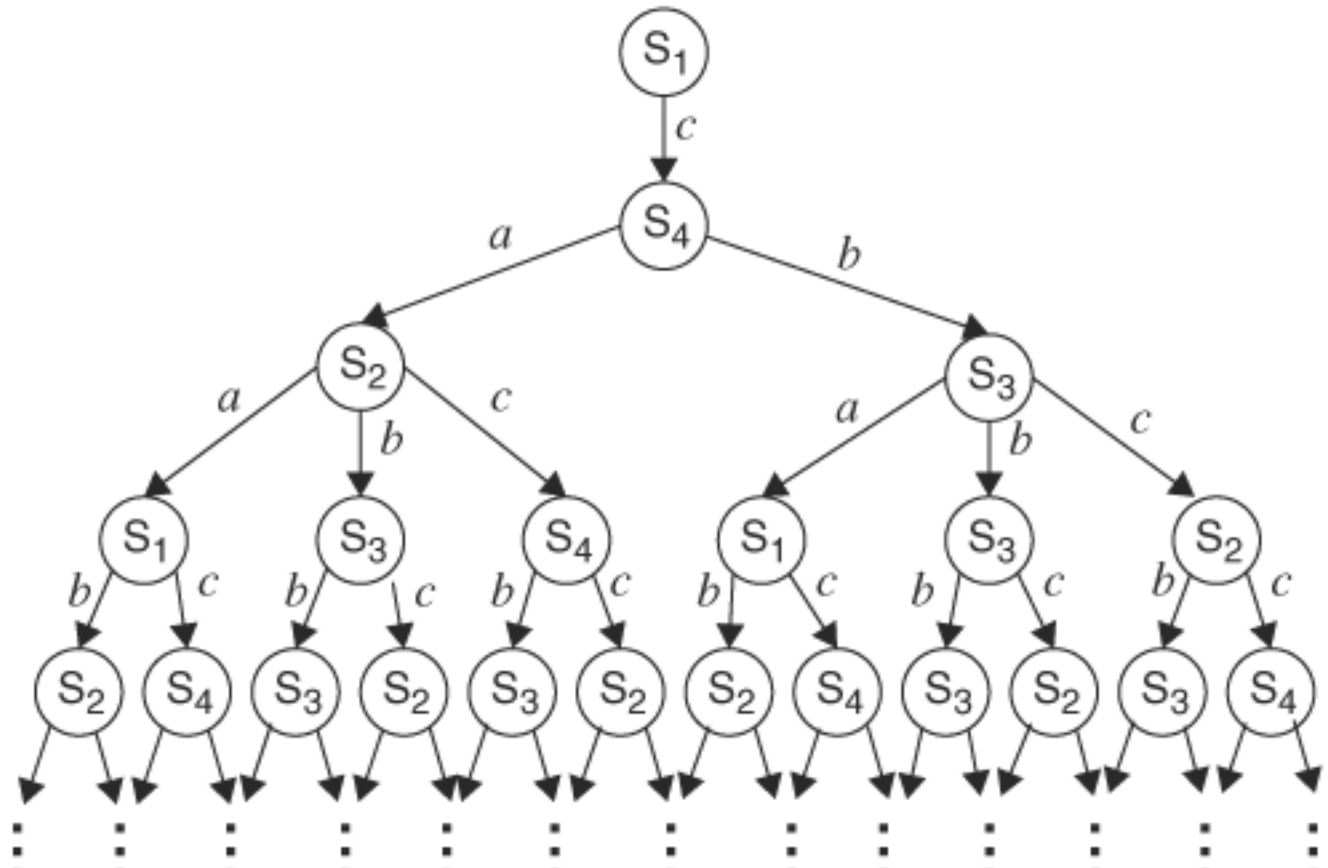
AUTOMATED CTL* VERIFICATION

- First known tool for *automatically* proving CTL* properties of infinite-state programs.
- Solution based precondition synthesis over prophecy variables which determine nondeterministic decisions regarding which paths are taken.
  - Prophecies: Variables that summarize the future of the program execution.

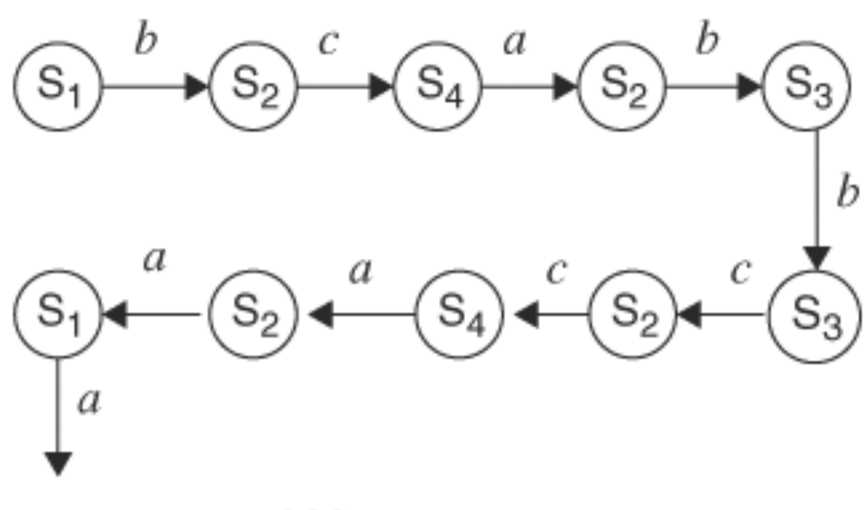
TEMPORAL LOGIC

- Logic reasoning about propositions qualified in terms of time.
- Used as a specification language as it encompasses safety, liveness, fairness, etc.
- Most commonly used sub-logics are CTL*, CTL (state based), and LTL (trace based).

CTL VS LTL

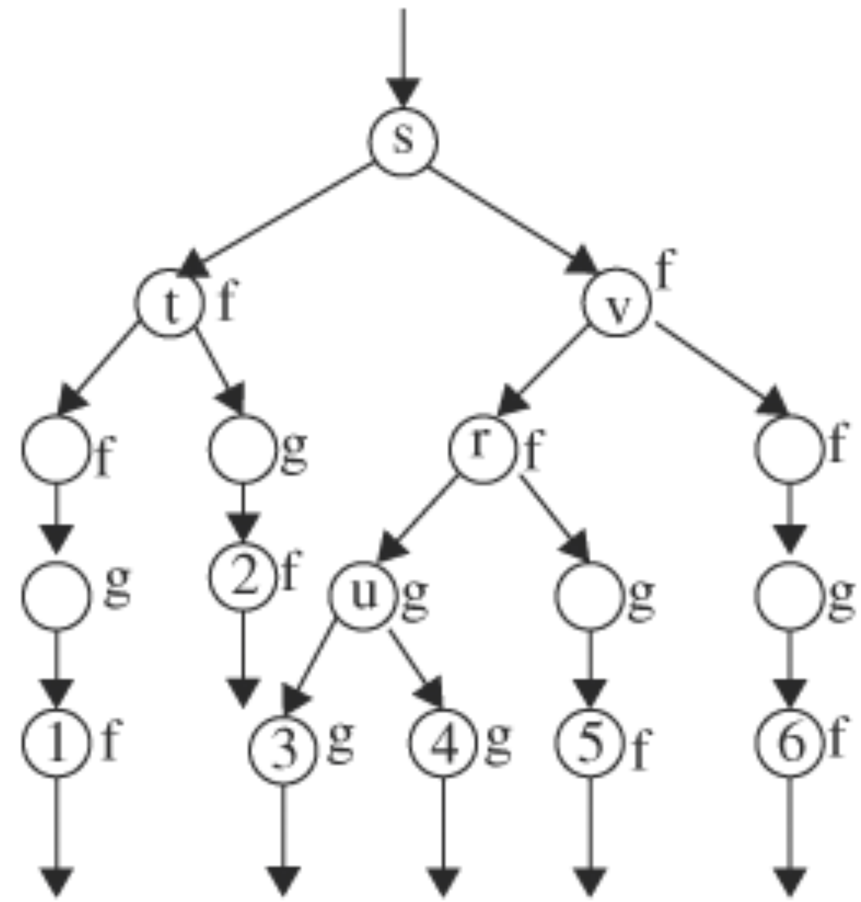


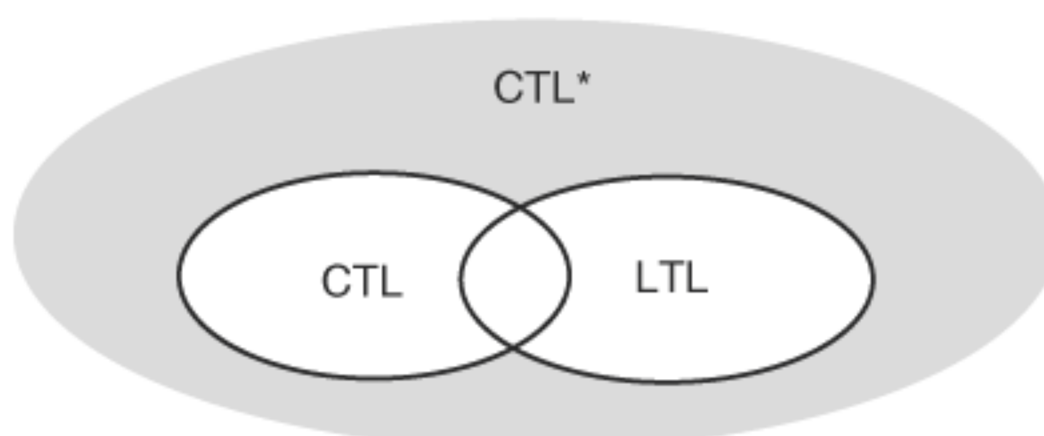
CTL



LTL

CTL*





CTL

- Reasoning about sets of states.
- Reasoning about non-deterministic (branching) programs.
- $\varphi ::= \alpha \mid \neg\alpha \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid AX\varphi \mid AF\varphi \mid A[\varphi W \varphi] \mid EX\varphi \mid EG\varphi \mid E[\varphi U \varphi]$
- 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.

CTL

- X φ – Next: φ has to hold at the next state.
- G φ – Globally: φ has to hold on the all states along a path.
- F φ – Finally: φ eventually has to hold.
- φ_1 U φ_2 – Until: φ_1 has to hold at least until at some position φ_2 holds. φ_2 must be verified in the future.
- φ_1 W φ_2 – Weak until: φ_1 has to hold until φ_2 holds.

LTL

- Reasoning about sets of paths.
- Reasoning about concurrent programs.
- $\psi ::= \alpha \mid \psi \wedge \psi \mid \psi \vee \psi \mid G\psi \mid F\psi \mid [\psi W \psi] \mid [\psi U \psi]$.

CTL*

- CTL* can express both CTL, LTL, and properties requiring path and state based interplay.
- $\phi ::= \alpha \mid \neg\alpha \mid \phi \wedge \phi \mid \phi \vee \phi \mid A\psi \mid E\psi$
- $\psi ::= \phi \mid \psi \wedge \psi \mid \psi \vee \psi \mid G\psi \mid F\psi \mid [\psi W \psi] \mid [\psi U \psi]$

CTL*

- LTL: Can naturally express fairness: $GF\ p \Rightarrow GF\ q$.
- CTL: Can express *existential* properties.
- CTL* allows the interplay between LTL and CTL properties:
  - "Along some future an event occurs infinitely often" (EGF)
  - $EFG(\neg x \wedge (EGF\ x))$
  - $AG(EG\ \neg x) \vee (EFG\ y)$

VERIFYING CTL* (OVERVIEW)

- Recurse over a CTL* formula, and for each sub-formula θ produce a satisfying precondition.
  - Deconstruction allows us to identify the interplay of path and state formulae.
- State formulae preconditions acquired via existing CTL techniques.
- **How to acquire sufficient path formulae preconditions that admit a sound interaction with state formulae?**

VERIFYING CTL* (OVERVIEW)

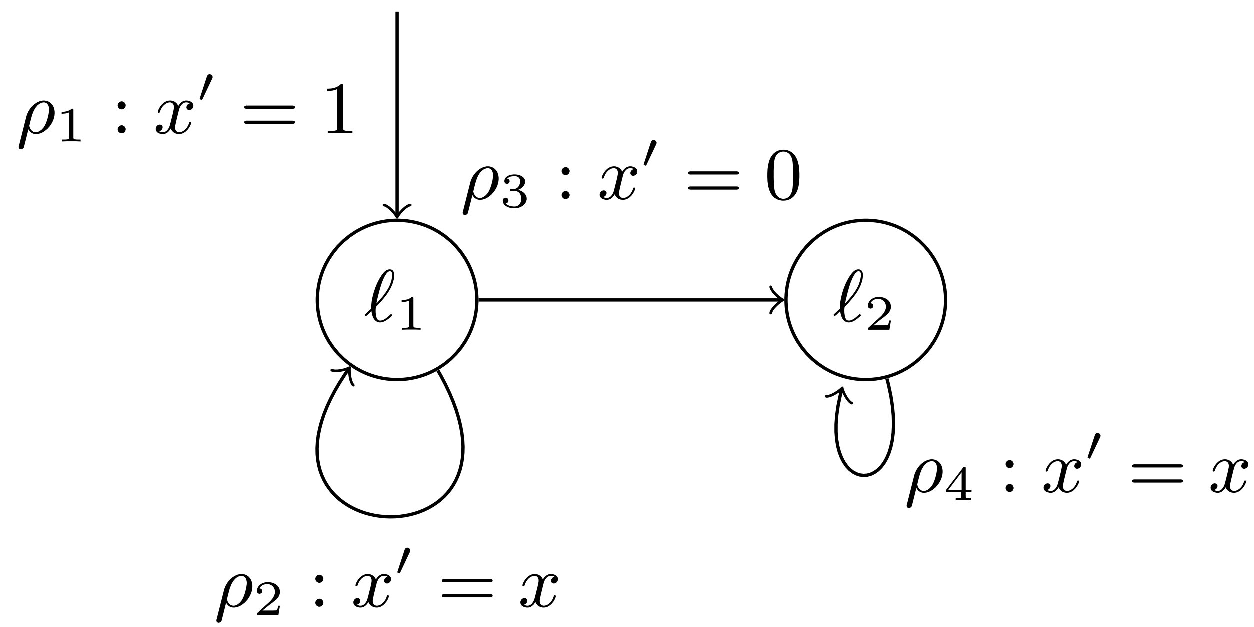
1. **Formula:** Over-approximate a path sub-formula to a universal CTL formula (ACTL).

2. **TS:** Nondeterministic decisions regarding which paths are taken are determined by prophecy variables.

3. Use an existing CTL model-checker.

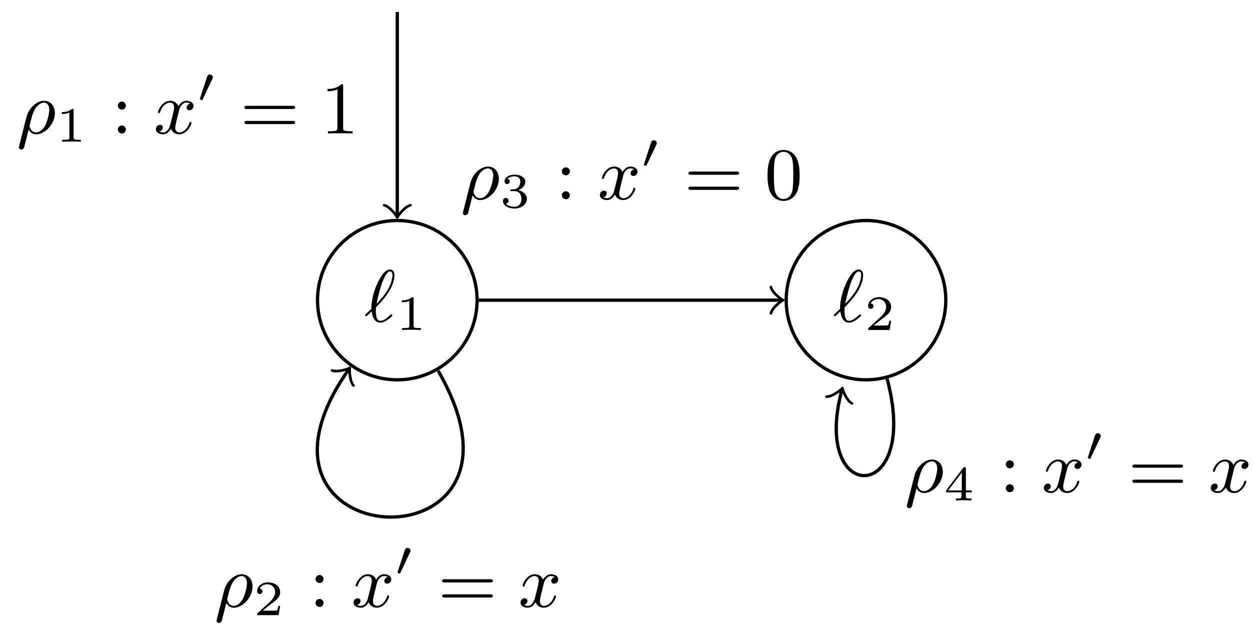
4. Apply QE over prophecies to acquire sound precondition.

EXAMPLE



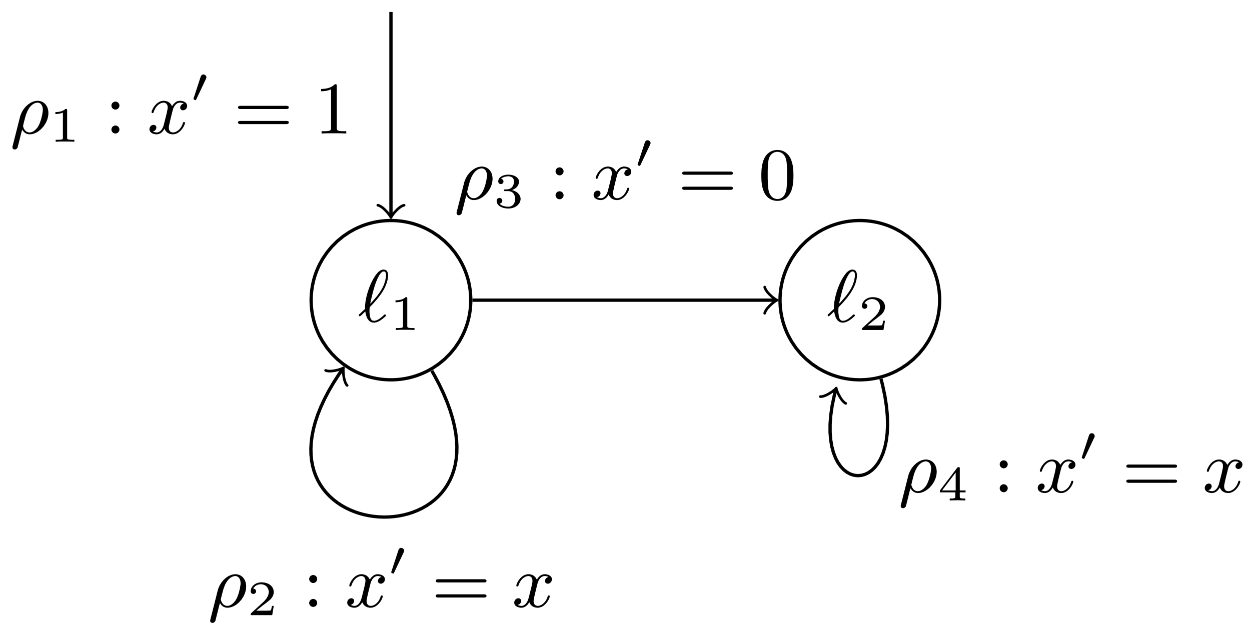
- Prove the CTL* property EFG x = 1.

APPROXIMATE



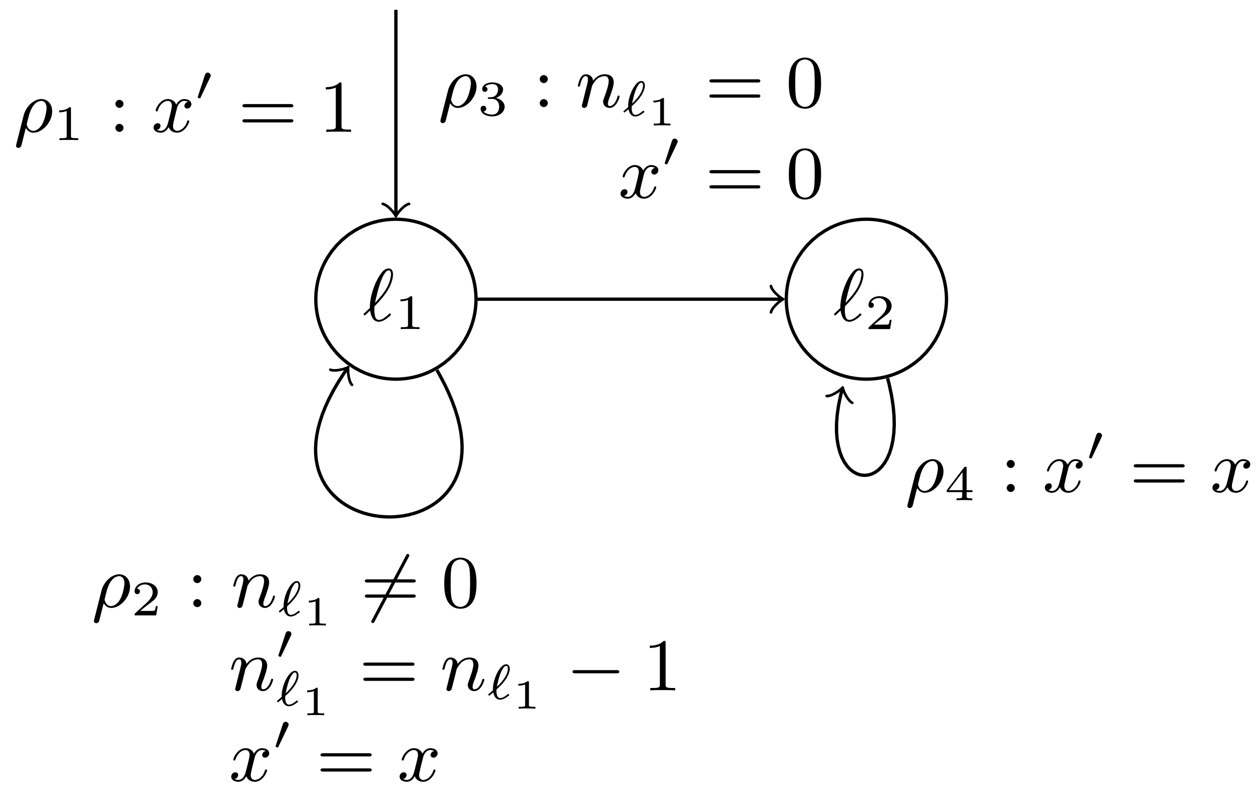
- Prove the CTL* sub-property G x = 1.
  - Over-approximate to **A**G x = 1.
  - No set of states exemplify the infinite possibilities of leaving **ρ**2 to possibly reaching **ρ**3 or remaining in **ρ**2 forever.

DETERMINIZE



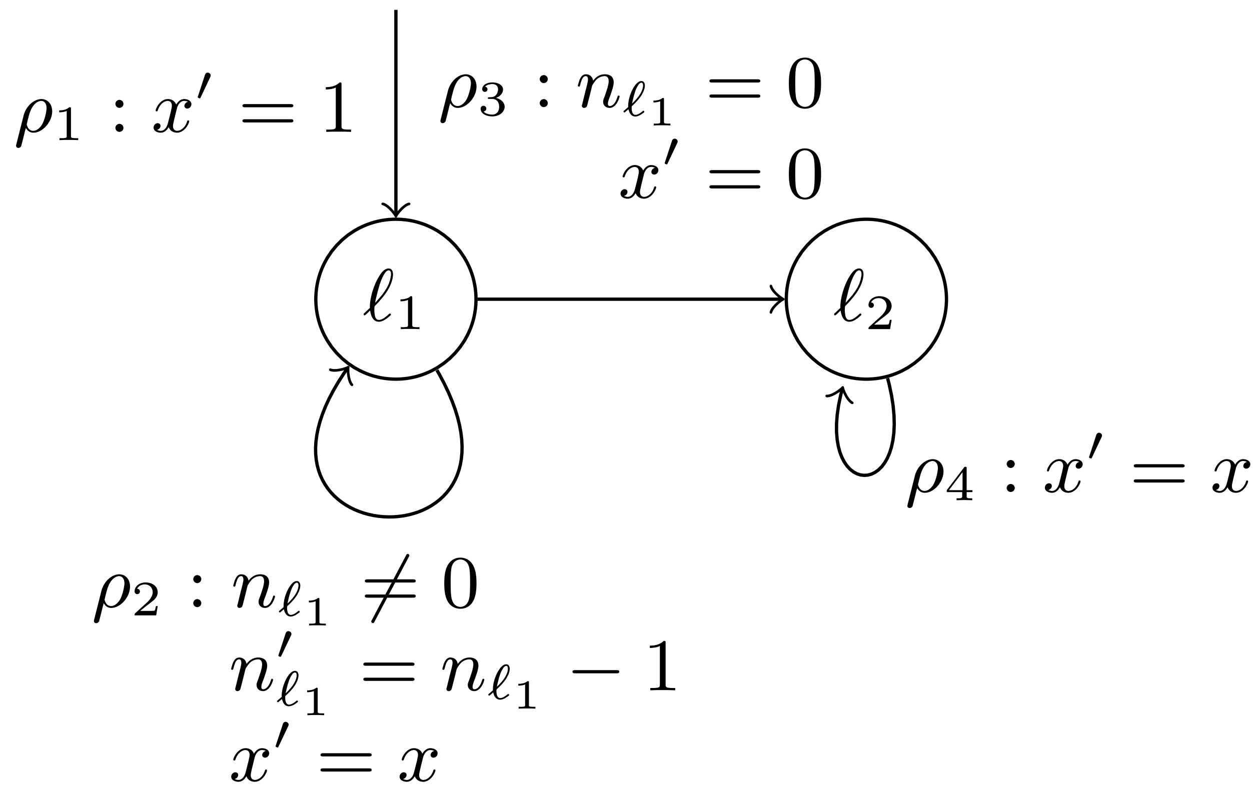
- Construct a *partially* determinized program over **relation pairs.**
  - Transitions stemming from same location, but are not part of the same strongly connected subgraph.
  - We identify ($\boldsymbol{\rho}_2$, $\boldsymbol{\rho}_3$) as a relation pair.

DETERMINIZE



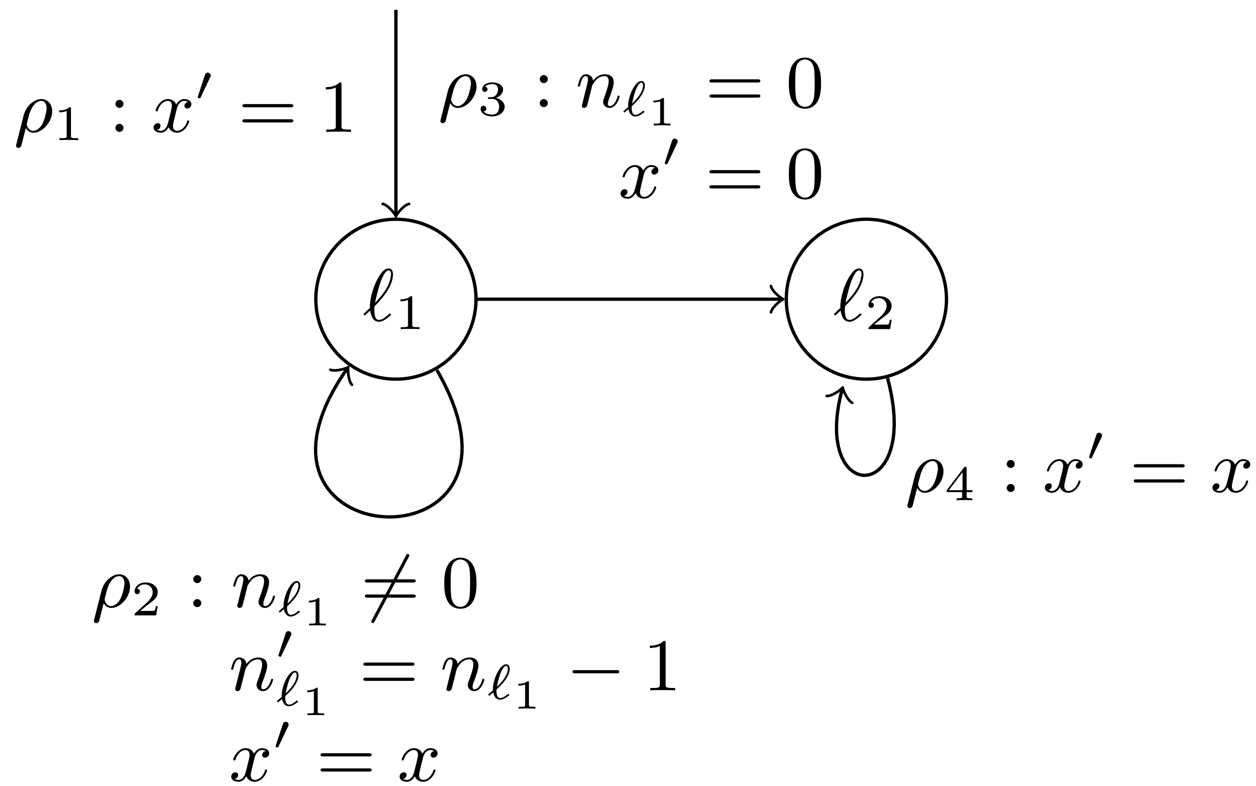
- Introduce prophecy variable (n_{L1}) associated with the relation pair (ρ_2, ρ_3).
  - Used to make predictions about the path taken.

DETERMINIZE



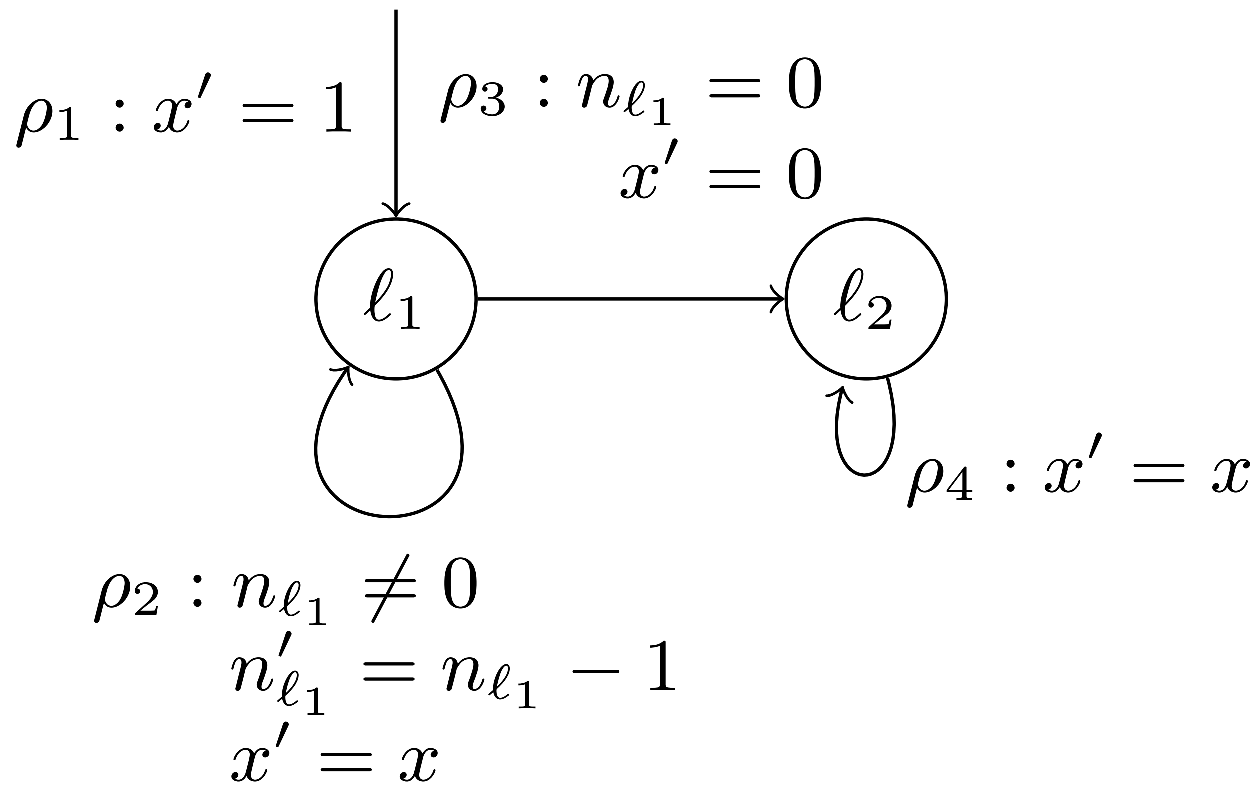
- A positive number chosen predicts the number of instances that transition ρ_2 is visited before transitioning to ρ_3.
    - We remain in ρ_2 until $n_{L1} = 0$, with n_{L1} being decremented each time.
- A negative assignment to n_{L1} denotes remaining in ρ_2 forever, or non-termination.

PRECONDITION SYNTHESIS



- We can now use an existing CTL model-checker!
- Returns an assertion characterizing the states in which AG x = 1.
- $a_G = (l_1 \wedge n_{L1} < 0)$ is returned.

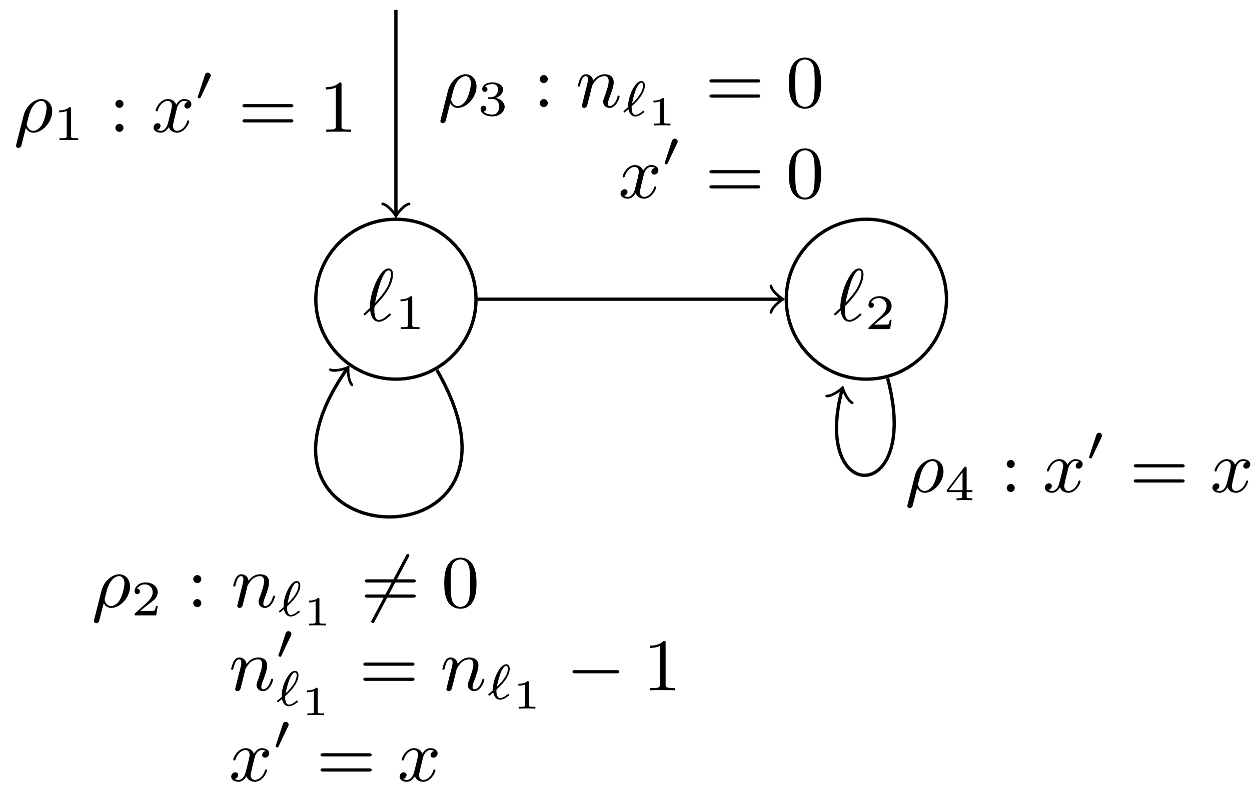
PRECONDITION SYNTHESIS



- $a_G = (l_I \wedge n_{LI} < 0)$.

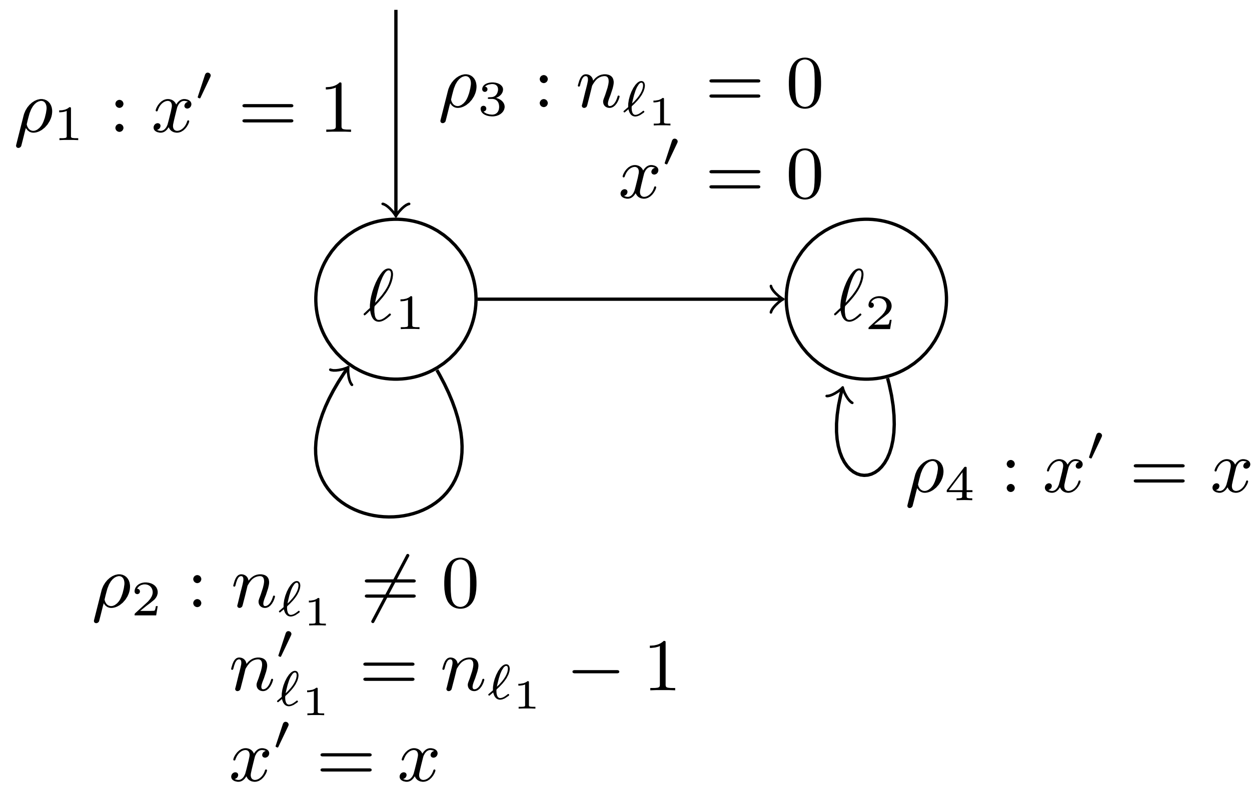
- Replace the sub-formula with its assertion in the original CTL* formula: EFa_G.

QUANTIFIER ELIMINATION



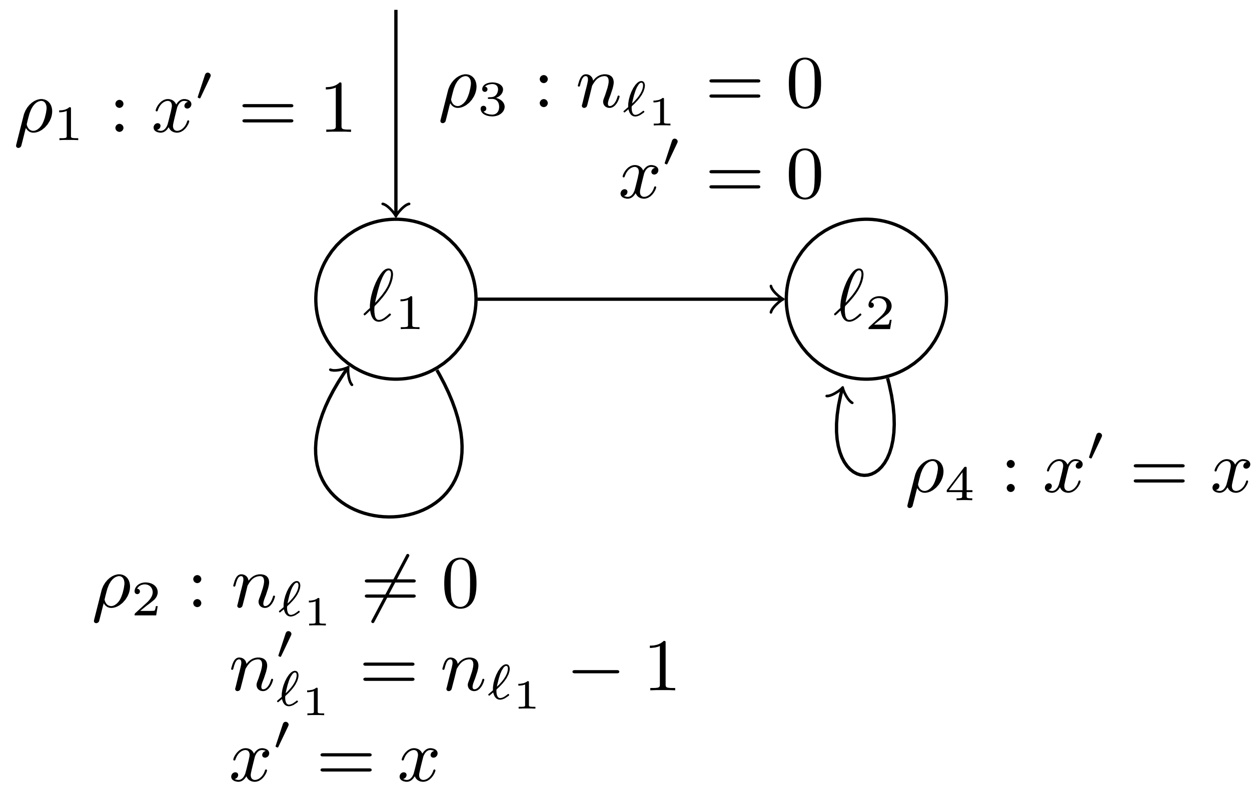
- EFa_G is a readily acceptable CTL formula.
- E exists within a larger context reasoning about paths (inner formula FG).
- To interchange between path and state formulae, we collapse determinized relations to incorporate path quantifiers via **QE**.

QUANTIFIER ELIMINATION



- Verify EFa$_G$ over the same determinized program above.
- Precondition $(l_l \wedge n_{Ll} < 0)$ is returned (again).
- Use QE to existentially quantify out introduced prophecy variables.

QUANTIFIER ELIMINATION



- Existential quantification corresponds to searching for some path (or paths) that satisfy the path formula.
- EFG x = 1 holds.

VERIFYING CTL*

1. **Approximate:** Over-approximate a path sub-formula to a universal CTL formula (ACTL).

2. **Determinize:** Nondeterministic decisions regarding which paths are taken are determined by prophecy variables.

3. **Precondition Synthesis:** Through an existing CTL model-checker.

4. **Quantifier Elimination:** Allow path formulae preconditions to admit a sound interaction with state formulae.

EXPERIMENTS

Program	LoC	Property	Time(s)	Res.
OS frag. 1	393	$\mathsf{AG}((\mathsf{EG}(\texttt{phi\_io\_compl} \leq 0)) \vee (\mathsf{EFG}(\texttt{phi\_nSUC\_ret} > 0))))$	32.0	$\times$
OS frag. 1	393	$\mathsf{EF}((\mathsf{AF}(\texttt{phi\_io\_compl} > 0)) \wedge (\mathsf{AGF}(\texttt{phi\_nSUC\_ret} \leq 0))))$	13.2	$\checkmark$
OS frag. 2	380	$\mathsf{EFG}((\texttt{keA} \leq 0 \wedge (\mathsf{AG}\ \texttt{keR} = 0)))$	28.3	$\checkmark$
OS frag. 2	380	$\mathsf{EFG}((\texttt{keA} \leq 0 \vee (\mathsf{EF}\ \texttt{keR} = 1)))$	16.5	$\checkmark$
OS frag. 3	50	$\mathsf{EF}(\texttt{PPBlockInits} > 0 \wedge (((\mathsf{EFG}\ \texttt{IoCreateDevice} = 0)$ $\vee (\mathsf{AGF}\ \texttt{status} = 1)) \wedge (\mathsf{EG}\ \texttt{PPBunlockInits} \leq 0)))$	10.4	$\checkmark$
PgSQL arch 1	106	$\mathsf{EFG}(\texttt{tt} > 0 \vee (\mathsf{AF}\ \texttt{wakend} = 0))$	1.5	$\times$
PgSQL arch 1	106	$\mathsf{AGF}(\texttt{tt} \leq 0 \wedge (\mathsf{EG}\ \texttt{wakend} \neq 0))$	3.8	$\checkmark$
PgSQL arch 1	106	$\mathsf{EFG}(\texttt{wakend} = 1 \wedge (\mathsf{EGF}\ \texttt{wakend} = 0))$	18.3	$\checkmark$
PgSQL arch 1	106	$\mathsf{EGF}(\mathsf{AG}\ \texttt{wakend} = 1)$	10.3	$\checkmark$
PgSQL arch 1	106	$\mathsf{AFG}(\mathsf{EF}\ \texttt{wakend} = 0)$	1.5	$\times$
PgSQL arch 2	100	$\mathsf{AGF}\ \texttt{wakend} = 1$	1.4	$\checkmark$
PgSQL arch 2	100	$\mathsf{EFG}\ \texttt{wakend} = 0$	0.5	$\times$
Bench 1	12	$\mathsf{EFG}(\texttt{x} = 1 \wedge (\mathsf{EG}\ \texttt{y} = 0))$	1.0	$\checkmark$
Bench 2	12	$\mathsf{EGF}\ \texttt{x} > 0$	0.1	$\checkmark$
Bench 3	12	$\mathsf{AFG}\ \texttt{x} = 1$	0.1	$\checkmark$
Bench 4	10	$\mathsf{AG}((\mathsf{EFG}\ \texttt{y} = 1) \wedge (\mathsf{EF}\ \texttt{x} \geq \texttt{t}))$	0.5	$\times$
Bench 5	10	$\mathsf{AG}(\texttt{x} = 0\ \mathsf{U}\ \texttt{b} = 0)$	T/O	–
Bench 6	8	$\mathsf{AG}((\mathsf{EFG}\ \texttt{x} = 0) \wedge (\mathsf{EF}\ \texttt{x} = 20))$	0.1	$\checkmark$
Bench 7	6	$(\mathsf{EFG}\texttt{x} = 0) \wedge (\mathsf{EFG}\texttt{y} = 1)$	0.5	$\times$
Bench 8	6	$\mathsf{AG}((\mathsf{AFG}\ \texttt{x} = 0) \vee (\mathsf{AFG}\texttt{x} = 1))$	0.5	$\checkmark$

RECAP

- The first known method for symbolically and automatically proving CTL* properties of (infinite-state) integer programs.
- Solution based on program transformation which trades nondeterminism in the transition relation for nondeterminism explicit in prophecy variables.
- Implemented as an extension to **T2:** https://github.com/hkhlaaf/T2/tree/T2Star

Eleventh Haifa Verification Conference

HVC2015

November 17 – 19   Haifa, Israel

Submission deadline: July 24, 2015