

Deciding Full Branching Time Logic by Program Transformation

Alberto Pettorossi (Univ. Tor Vergata, Rome, Italy),

Maurizio Proietti (IASI-CNR, Rome, Italy),

Valerio Senni (Univ. Tor Vergata, Rome, Italy)

Pescara, Italy

April 16, 2010

Our Goal

... is to establish by program transformation
the correctness of
finite state concurrent systems with infinite behaviour
(reactive systems) such as:

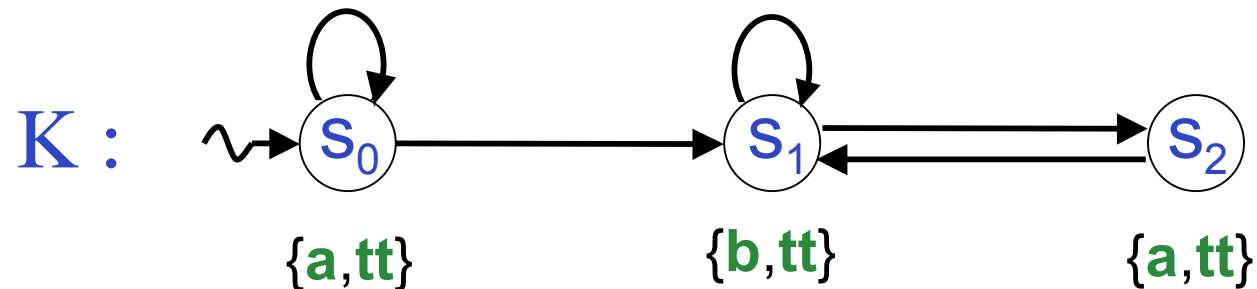
- communication protocols
- security protocols
- hardware controllers
- ...

Related Work

- Proving first order formulas via unfold/fold rules
[Kott 1982, P.P. 1999, Roychoudhury et al. 1999, P.P. 2000, ...]
- Verifying temporal properties of infinite state systems
via specialization and unfold/fold rules
[Leuschel 1999, 2000, Roychoudhury et al. 2000,
Fioravanti et al. 2001, ...]

Concurrent Systems and Properties

■ Concurrent Systems as Kripke Structures



■ Properties as Formulas in CTL*

$$\text{Elem} = \{a, b, tt\}$$

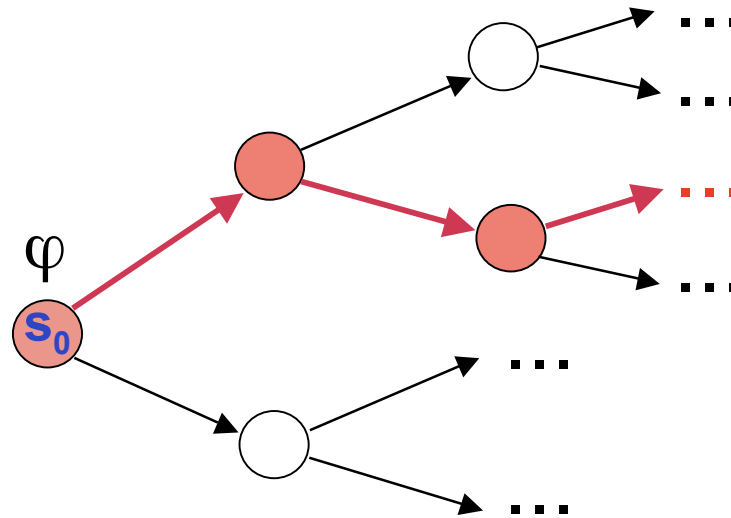
$$\varphi = E(a \text{ U } \neg E(tt \text{ U } \neg (tt \text{ U } \neg b)))$$

LTL (linear-time temporal logic) \subset CTL*

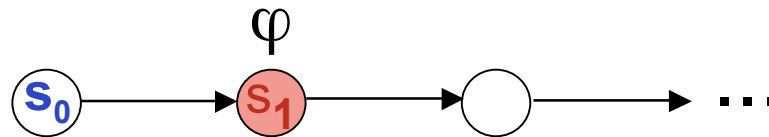
CTL (computational tree logic) \subset CTL*

exists a path - next - until

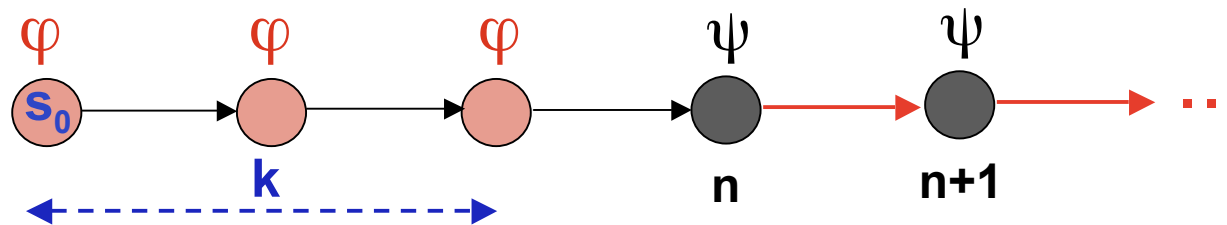
E φ :
(exists a path)



X φ :
(next)



φ **U** ψ :
(until)



A Kripke Structure

$$K = \langle \Sigma, s_{in}, \rho, \lambda \rangle$$

Σ = finite set of states

s_{in} = initial state

ρ = **total** transition relation $\subseteq \Sigma \times \Sigma$

λ = labelling function: $\Sigma \rightarrow 2^{\text{Elem}}$

A **computation path** $\pi = [s_0, s_1, \dots]$ is an **infinite list** of states.

Formulas of CTL*

elementary formulas :

$d \in \text{Elem}$

state formulas :

$\varphi = d \mid \neg \varphi \mid \varphi \wedge \varphi \mid E \psi$
(exists a path)

path formulas :

$\psi = \varphi \mid \neg \psi \mid \psi \wedge \psi \mid X \psi \mid \psi U \psi$
(next) (until)

Semantics of CTL*

Let \mathbf{K} be a Kripke structure.

Let π be the infinite list $[s_0, s_1, \dots, s_k, \dots, s_n, \dots]$ of states.

Let \mathbf{d} , φ , ψ be formulas of CTL*.

$\mathbf{K}, \pi \models \mathbf{d}$ iff $\mathbf{d} \in \lambda(s_0)$

$\mathbf{K}, \pi \models \neg \varphi$ iff $\mathbf{K}, \pi \models \varphi$ does not hold

$\mathbf{K}, \pi \models \varphi \wedge \psi$ iff $\mathbf{K}, \pi \models \varphi$ and $\mathbf{K}, \pi \models \psi$

$\mathbf{K}, \pi \models \mathbf{E} \varphi$ iff $\exists \pi' = [s_0, \dots], \mathbf{K}, \pi' \models \varphi$ (exists a path)

$\mathbf{K}, \pi \models \mathbf{X} \varphi$ iff $\mathbf{K}, [s_1, \dots] \models \varphi$ (next)

$\mathbf{K}, \pi \models \varphi \mathbf{U} \psi$ iff $\exists n \geq 0$ ($(\forall k, 0 \leq k < n, \mathbf{K}, [s_k, \dots] \models \varphi)$
and $\mathbf{K}, [s_n, \dots] \models \psi$) (until)

CTL* Model Checking

Definition.

A **state formula** φ holds in the **Kripke structure** \mathbf{K} with **initial state** s_{in} :

$$\mathbf{K} \models \varphi \quad \text{iff} \quad \exists \pi = [s_{in}, \dots], \quad \mathbf{K}, \pi \models \varphi$$

CTL* Model Checking:

given \mathbf{K} and φ , verify whether or not $\mathbf{K} \models \varphi$ holds.

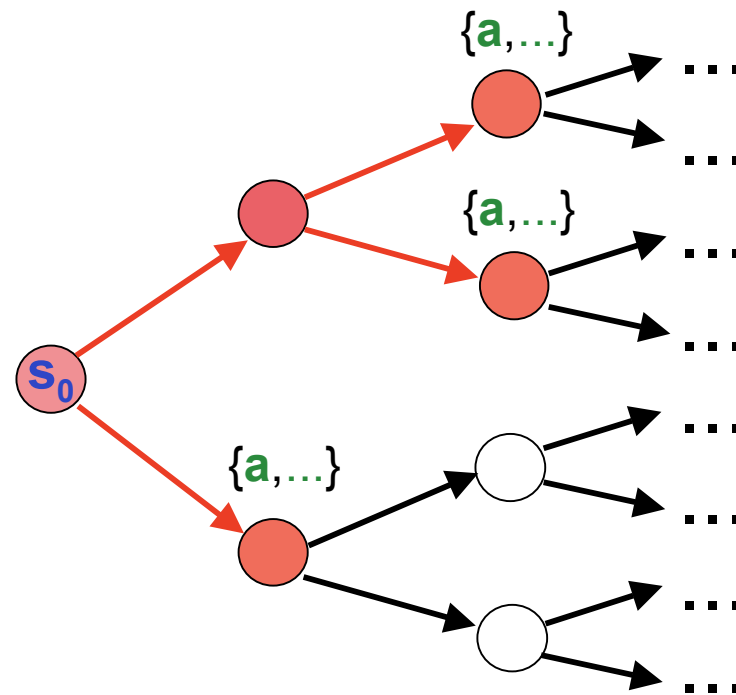
More Syntax

in the **F**uture : $\mathbf{F} \varphi = \mathbf{tt} \mathbf{U} \varphi$

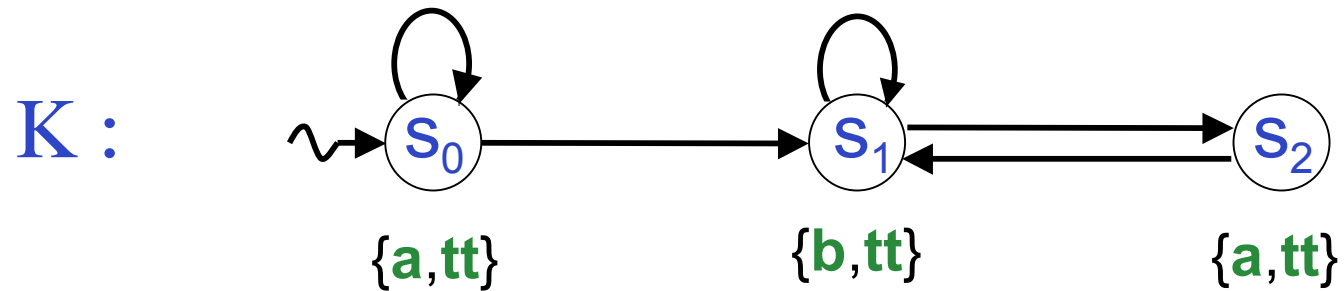
Globally : $\mathbf{G} \varphi = \neg \mathbf{F} \neg \varphi$

for **A**ll : $\mathbf{A} \varphi = \neg \mathbf{E} \neg \varphi$

$\mathbf{K}, [s_0, \dots] \models \mathbf{AFa}$ holds if



An Example



$$\begin{aligned}\varphi &= \mathbf{E}(\mathbf{a} \mathbf{U} \neg \mathbf{E}(\mathbf{tt} \mathbf{U} \neg (\mathbf{tt} \mathbf{U} \mathbf{b}))) \\ &= \mathbf{E}(\mathbf{a} \mathbf{U} \mathbf{A} \mathbf{G} \mathbf{F} \mathbf{b})\end{aligned}$$

path witnesses: $s_0^+ (s_1^+ s_2)^{\omega}$

Thus, $\mathbf{K} \models \varphi$

Overview

- ✓ • Modeling reactive systems and their properties via **Kripke structures** and **CTL* formulas**. $K \models \varphi$.

- Encoding CTL* formulas as **ω -programs** (programs on infinite lists)
- Transforming ω -programs into **monadic ω -programs**
- **Proof system** for monadic ω -programs

ω -programs

- ω -programs are a **typed, locally stratified logic programs** where $[_|_]$ is interpreted as the constructor of **infinite lists**.
- Semantics of ω -programs = **perfect model** constructed over **infinite lists** (= least Herbrand model for definite programs).

$\Sigma = \{s_0, s_1\}$. L ranges over $(s_0+s_1)^\omega$.

■ P: $p([s_0|L]) \leftarrow q(L)$ $p(L) \in M(P)$ iff $L \in s_0(s_0+s_1)^\omega$
 $q(L) \leftarrow$

■ P: $p(L) \leftarrow \neg q(L)$ $p(L) \in M(P)$ iff $L \in s_0^\omega$
 $q([s_0|L]) \leftarrow q(L)$ $q(L) \in M(P)$ iff $L \in s_0^* s_1 (s_0+s_1)^\omega$
 $q([s_1|L]) \leftarrow$

Negation gives extra expressivity. For P: $p([s_0|L]) \leftarrow p(L)$, $M(P) = \emptyset$.

Encoding the Satisfaction Relation \models (1)

$P_{K,\varphi}$: $\text{prop} \leftarrow \text{sat}([s_0|X], [\varphi])$

$\text{sat}([S|X], D) \leftarrow \text{elem}(D,S)$

$\text{sat}(X, \text{not } F) \leftarrow \neg \text{sat}(F,S)$

$\text{sat}(X, \text{and}(F_1,F_2)) \leftarrow \text{sat}(X,F_1) \wedge \text{sat}(X,F_2)$

$\text{sat}([S|X], e(F)) \leftarrow \text{exists-sat}(S,F)$

$\text{exists-sat}(S,F) \leftarrow \text{path}([S|Y]) \wedge \text{sat}([S|Y], F)$

$\text{sat}([S|X], x(F)) \leftarrow \text{sat}(X,F)$

$\text{sat}(X, u(F_1,F_2)) \leftarrow \text{sat}(X,F_2)$

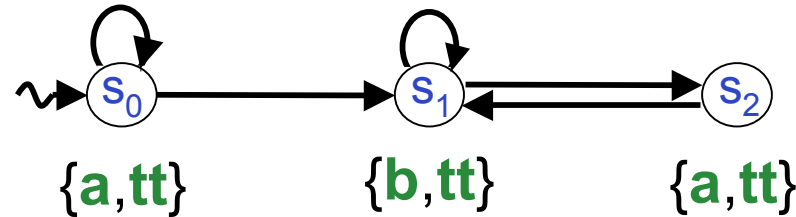
$\text{sat}([S|X], u(F_1,F_2)) \leftarrow \text{sat}([S|X], F_1) \wedge \text{sat}(X, u(F_1,F_2))$

$\text{path}(X) \leftarrow \neg \text{notpath}(X)$

$\text{notpath}([S_1,S_2|X]) \leftarrow \neg \text{transition}(S_1,S_2)$

$\text{notpath}([S|X]) \leftarrow \text{notpath}(X)$

Encoding the Satisfaction Relation \models (2)



elem(a, s₀) ←

elem(b, s₁) ←

elem(a, s₂) ←

elem(tt, X) ←

transition(s₀, s₀) ←

transition(s₀, s₁) ←

transition(s₁, s₁) ←

transition(s₁, s₂) ←

transition(s₂, s₁) ←

$$\varphi = \mathbf{E} (\mathbf{a} \mathbf{U} \neg \mathbf{E} (\mathbf{tt} \mathbf{U} \neg (\mathbf{tt} \mathbf{U} \mathbf{b})))$$

$$[\varphi] = \mathbf{e}(\mathbf{u}(\mathbf{a}, \mathbf{not}(\mathbf{e}(\mathbf{u}(\mathbf{tt}, \mathbf{not}(\mathbf{u}(\mathbf{tt}, \mathbf{b}))))))))$$

Correctness of the Encoding

Theorem 1. $K \models \varphi$ iff $M(P_{K,\varphi}) \models \text{prop}$

How to check whether or not $M(P_{K,\varphi}) \models \text{prop}$?

- *Top-down* evaluation of the query **prop** does not terminate.
- *Bottom-up* construction of $M(P_{K,\varphi})$ does not terminate, because of infinite lists.

Overview

- ✓• Modeling reactive systems and their properties via **Kripke structures** and **CTL*** formulas. $K \models \varphi$.
- ✓• Encoding CTL* formulas as **ω -programs** (programs on infinite lists)
 - 1. Transforming ω -programs into **monadic ω -programs**
 - 2. **Decision algorithm** for monadic ω -programs

Monadic ω -programs

A monadic ω -program is a stratified set of monadic ω -clauses of the form:

$$p_0([s|X_0]) \leftarrow \boxed{p_1(X_1) \wedge \dots \wedge p_k(X_k)} \wedge \boxed{\neg p_{k+1}(X_{k+1}) \wedge \dots \wedge \neg p_m(X_m)}$$

where:

S is a constant of type **state**,

X₀, **X**₁, ..., **X**_k, **X**_{k+1}, ..., **X**_m are variables of type **infinite-list**, and

let a clause be $A_0 \leftarrow \dots \wedge L_i \wedge \dots$

there exists a *level mapping* h such that for $i=1, \dots, m$,

if $\text{vars}(L_i) \not\subseteq \text{vars}(A_0)$ **and** $1 \leq i \leq k$ **then** $h(L_i) < h(A_0)$ **else** $h(L_i) \leq h(A_0)$

(Recall that: If L_i is $p_i(X_i)$ then $h(L_i) = h(p_i)$. If L_i is $\neg p_i(X_i)$ then $h(L_i) = h(p_i) + 1$.)

Some of the predicates p_i 's may be **nullary**, and they may be the same.

Some of the variables may be the same.

From $\mathbf{P}_{K,\varphi}$ to a Monadic ω -program \mathbf{T}

By applying the strategy:

(**instantiate** ; **unfold_{pos&neg}*** ; **subsume** ; **define-fold_{pos&neg}***)*

from $\mathbf{P}_{K,\varphi}$ we get a monadic ω -program \mathbf{T} .

Theorem 2. $\mathbf{M}(\mathbf{P}_{K,\varphi}) \models \mathbf{prop}$ iff $\mathbf{M}(\mathbf{T}) \models \mathbf{prop}$

Proof. Extension of the rules in [Seki 91].

Theorems

$\text{prop} \leftarrow \text{sat}([s_0|X], [\varphi])$

Theorem 1. $K \models \varphi$ iff $M(P_{K,\varphi}) \models \text{prop}$

Theorem 2. $M(P_{K,\varphi}) \models \text{prop}$ iff $M(T) \models \text{prop}$

The Monadic ω -program \mathbf{T}

$$\mathbf{T} : \quad \text{prop} \leftarrow \neg p_1(X) \wedge p_2(X)$$

$$\text{prop} \leftarrow \neg p_1(X) \wedge \neg p_3$$

$$p_1([s_0|X]) \leftarrow p_1(X)$$

$$p_1([s_1|X]) \leftarrow p_4(X)$$

$$p_1([s_2|X]) \leftarrow$$

$$p_2([s_0|X]) \leftarrow \neg p_3$$

$$p_2([s_0|X]) \leftarrow p_2(X)$$

$$p_2([s_1|X]) \leftarrow \neg p_5$$

$$p_2([s_2|X]) \leftarrow \neg p_6$$

$$p_2([s_2|X]) \leftarrow p_2(X)$$

$$p_3 \leftarrow \neg p_1(X) \wedge \neg p_7(X)$$

$$p_3 \leftarrow \neg p_1(X) \wedge p_8(X)$$

$$p_4([s_0|X]) \leftarrow$$

$$p_4([s_1|X]) \leftarrow p_4(X)$$

$$p_4([s_2|X]) \leftarrow p_9(X)$$

$$p_5 \leftarrow \neg p_4(X) \wedge p_8(X)$$

$$p_6 \leftarrow \neg p_9(X) \wedge \neg p_7(X)$$

$$p_6 \leftarrow \neg p_9(X) \wedge p_8(X)$$

$$p_7([s_0|X]) \leftarrow p_7(X)$$

$$p_7([s_1|X]) \leftarrow$$

$$p_7([s_2|X]) \leftarrow p_7(X)$$

$$p_8([s_0|X]) \leftarrow \neg p_7(X)$$

$$p_8([s_0|X]) \leftarrow p_8(X)$$

$$p_8([s_1|X]) \leftarrow p_8(X)$$

$$p_8([s_2|X]) \leftarrow \neg p_7(X)$$

$$p_8([s_2|X]) \leftarrow p_8(X)$$

$$p_9([s_0|X]) \leftarrow$$

$$p_9([s_1|X]) \leftarrow p_4(X)$$

$$p_9([s_2|X]) \leftarrow$$

Completeness of the Algorithm

Let \mathbf{T} be a monadic ω -program.

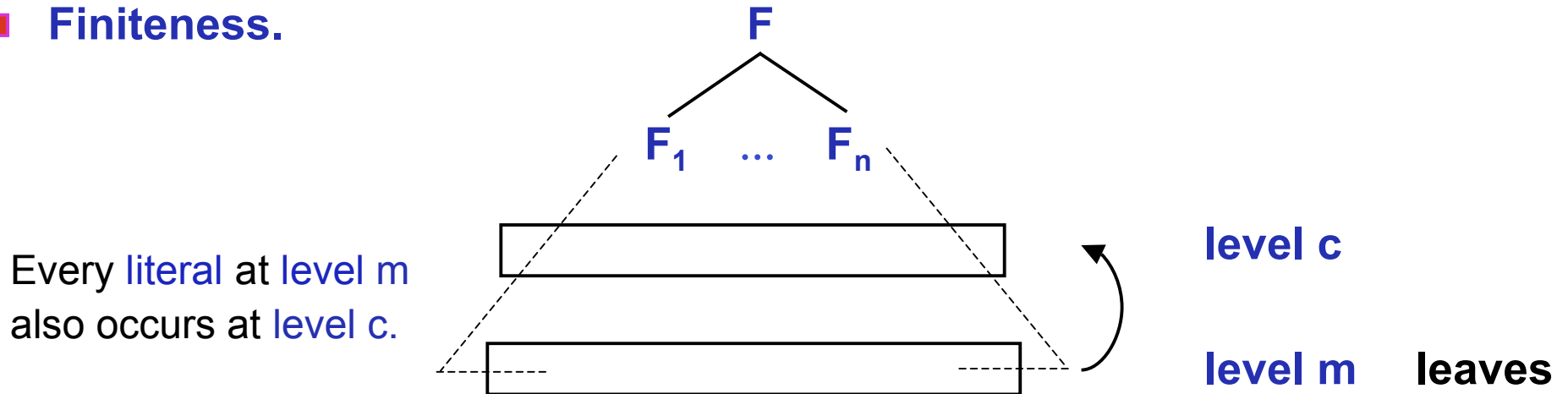
Let \mathbf{F} be a formula in \mathcal{F} .

Theorem 3. $\mathbf{M}(\mathbf{T}) \models \mathbf{F}$ iff \mathbf{F} has a **proof**.

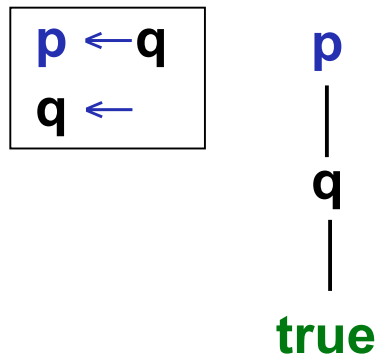
-
- **Derivation Tree:** - w.r.t. a program \mathbf{T}
 - an AND-tree
 - constructed level-by-level.
 - **Proof and Refutation.**

Derivation Tree: Basic Ideas

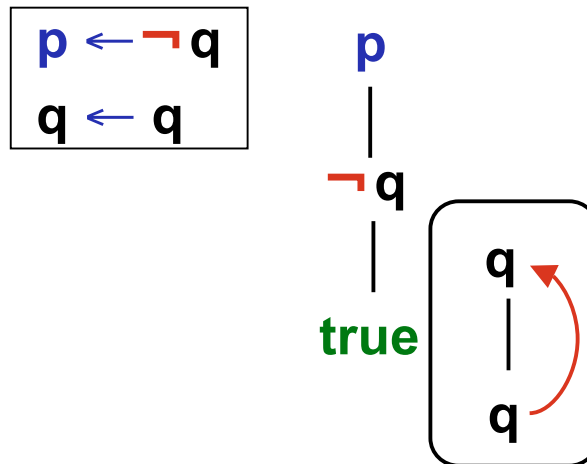
■ Finiteness.



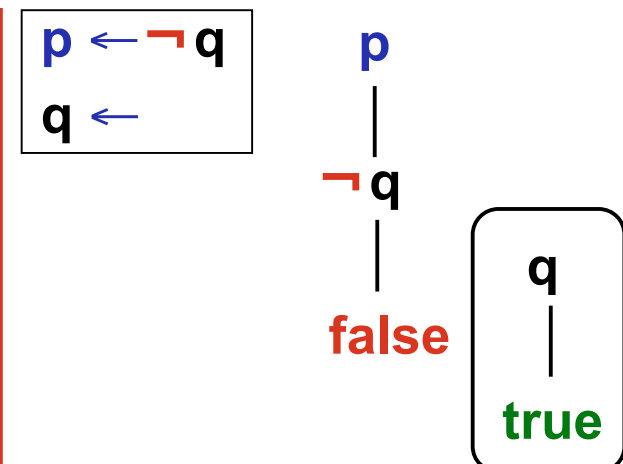
■ Positive loops.



$$M(P) = \{p, q\}$$



$$M(P) = \{p\}$$



$$M(P) = \{q\}$$

Derivation Tree

(1)

true | false

monadic literals: $M = q(X) \mid \neg q(X)$

literals: $M \mid p \mid \neg p$

$F \in \mathcal{F}$ $\mathcal{F} = p \mid \exists X (M_1 \wedge \dots \wedge M_n)$

$L \in \mathcal{L}$ $\mathcal{L} = M \mid \mathcal{F} \mid \neg \mathcal{F}$

complement: $\overline{F} = \neg F$ (with cancellation of $\neg \neg$)

Derivation Tree of $F \in \mathcal{F}$ (2)

1. Explicit existential quantifiers

$$q([s|X]) \leftarrow \dots \wedge q_1(Y) \wedge q_2(Y) \wedge \dots$$



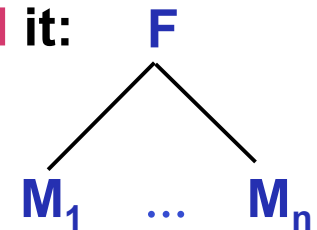
$$q([s|X]) \leftarrow \dots \wedge \exists Y(q_1(Y) \wedge q_2(Y)) \wedge \dots$$

Derivation Tree of $F \in \mathcal{F}$ (3)

2. AND-tree, constructed level-by level.

- The root is F .

If the root F is $\exists X (M_1 \wedge \dots \wedge M_n)$ expand it:



- Stop if: true, false, $\exists X (M_1 \wedge \dots \wedge M_n)$, $\neg \exists X (M_1 \wedge \dots \wedge M_n)$,
literals at level $d \subseteq$ literals at level c , with $c < d$
- Nondeterministically expand every literal L at lowest level.
Choose a state s .

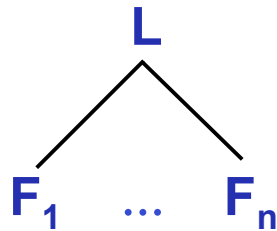
Derivation Tree of $F \in \mathcal{F}$

(4)

■ positive literal L :

Choose a clause for L :

$$\left[q([s|X]) \leftarrow F_1 \wedge \dots \wedge F_n \right]$$



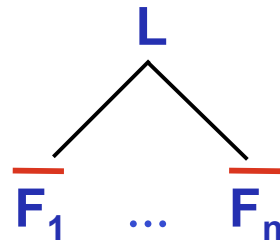
If $n=0$: L
|
true

■ negative literal L :

All clauses for L :

$$\left[\begin{array}{l} q([s|X]) \leftarrow B_1 \\ \dots \\ q([s|X]) \leftarrow B_n \end{array} \right]$$

Choose $F_1 \in B_1, \dots, F_n \in B_n$ (all F_n 's in \mathcal{L}) :



If $n=0$: L
|
true

If $\exists i B_i = \text{true}$: L
|
false

Proof and Refutation of $F \in \mathcal{F}$ w.r.t. \mathcal{T} (5)

■ A *proof* of $F \in \mathcal{F}$ is a derivation tree

- with root F

- every leaf is: *true*, p , $\neg p$, $q(X)$, $\neg q(X)$,

$\exists X (M_1 \wedge \dots \wedge M_n)$ which has a *proof* w.r.t. \mathcal{T}

$\neg \exists X (M_1 \wedge \dots \wedge M_n)$ which has a *refutation* w.r.t. \mathcal{T}

- for every leaf at m with a **positive literal** L , $r^+(L, L)$ does *not* hold, where $r(L_c, L_m)$ holds iff

a node N_c at c has label L_c , a node N_m at m has label L_m , and N_c is ancestor of N_m in \mathcal{T} .

■ $F \in \mathcal{F}$ has a *refutation* w.r.t. \mathcal{T} iff F has **no proof** w.r.t. \mathcal{T} .

Theorems

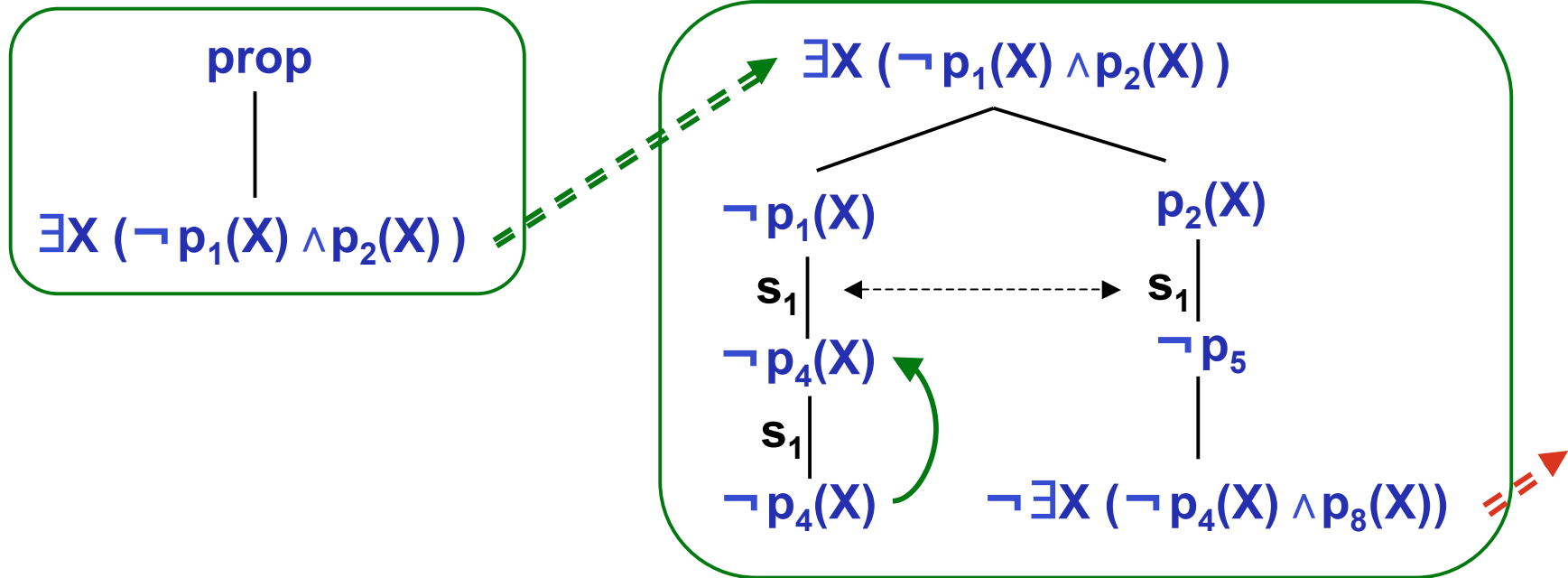
$\text{prop} \leftarrow \text{sat}([s_0|X], [\varphi])$

Theorem 1. $K \models \varphi$ iff $M(P_{K,\varphi}) \models \text{prop}$

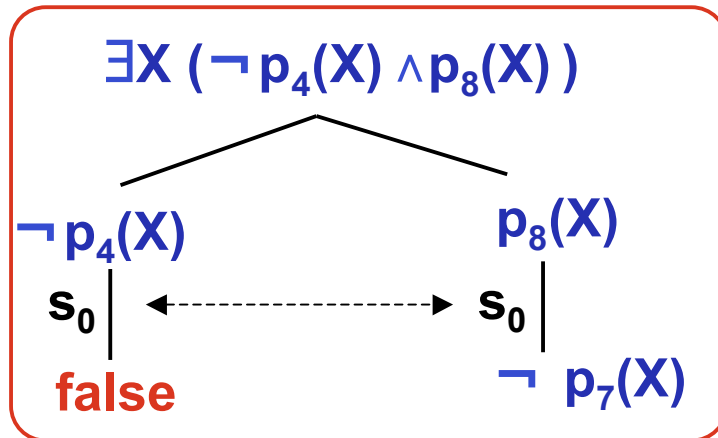
Theorem 2. $M(P_{K,\varphi}) \models \text{prop}$ iff $M(T) \models \text{prop}$

Theorem 3. $M(T) \models \text{prop}$ iff **F** has a **proof**.

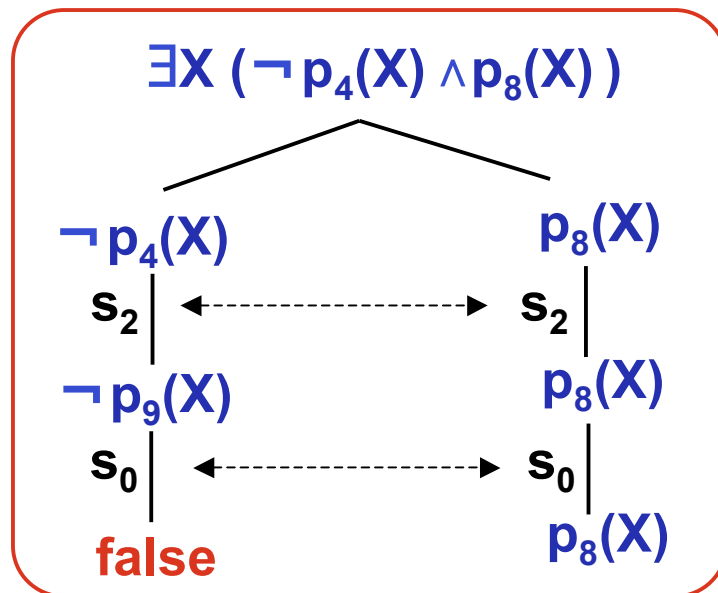
Proof of prop w.r.t. T



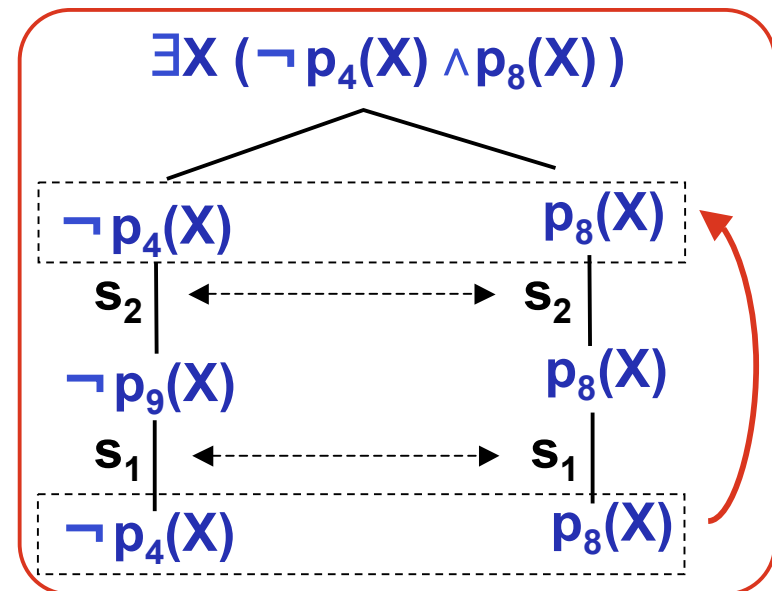
Refutation of $\neg \exists X (\neg p_4(X) \wedge p_8(X))$ w.r.t. \mathbf{T}



...



...



Future Work

- Use of constraints to avoid explicit state representation
- Infinite state model checking