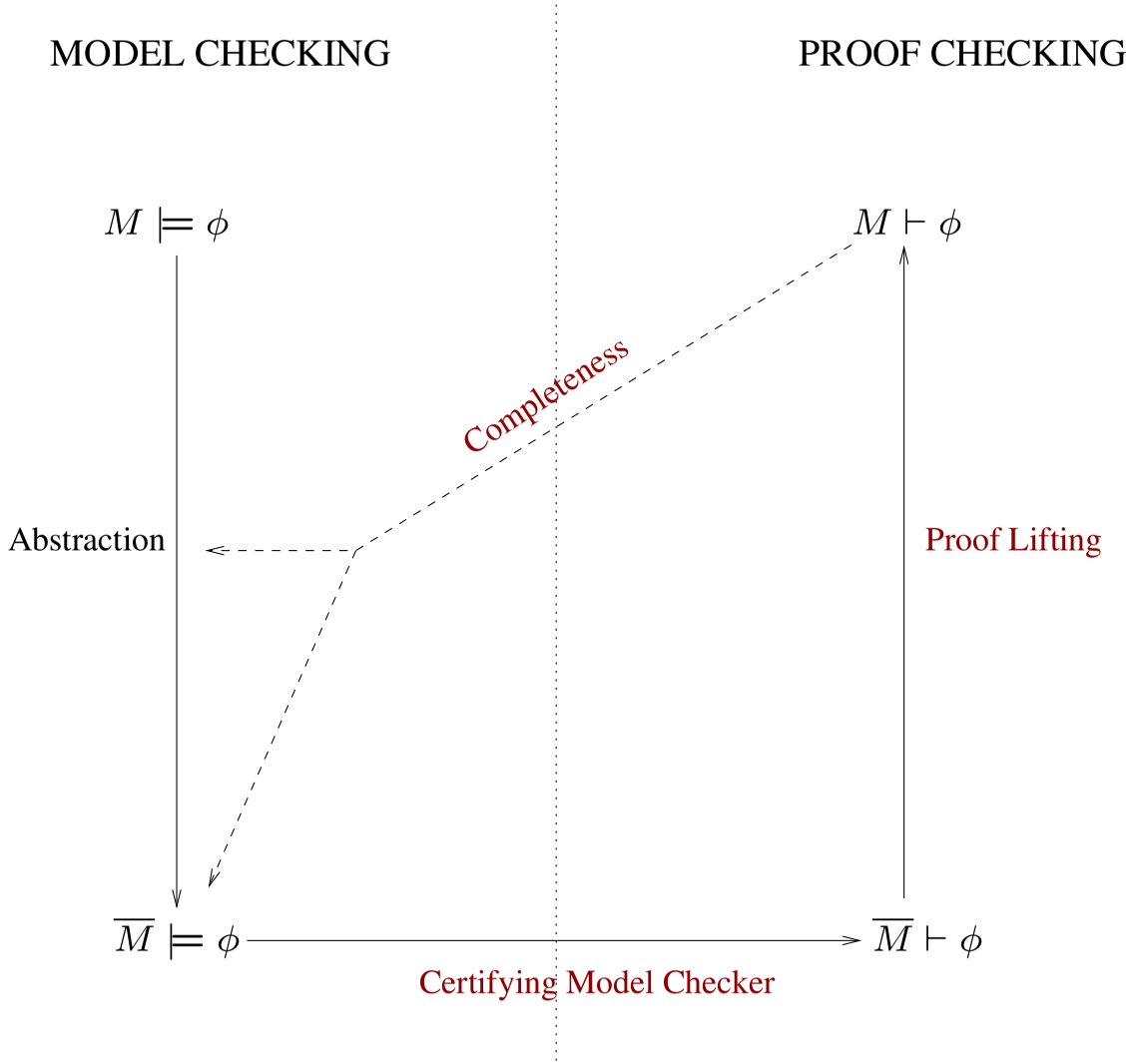
From Model Checking to Proof Checking ... and Back

Kedar Namjoshi
Bell Labs

April 29, 2005

Abstraction ∘ Model Checking = Deductive Proof



I. From Model Checking to Proof Checking

We show how to build a "certifying" model checker, one that generates a *proof* to justify its result.

Why bother?

- Proofs generalize counterexample traces for failure
- A proof is an *independently-checkable* certificate for success (think PCC for temporal properties)
- A proof is a convenient data structure for interactive exploration and incremental model checking

CTL Basics

The CTL logic is built out of atomic propositions, boolean operators, and the *temporal* operators EX(ϕ) ("ϕ holds of some successor"), E(ϕWψ) ("ϕ unless ψ"), and E(ϕUψ) ("ϕ until ψ").

Some derived operators:
EF(ϕ) ("ϕ is reachable") = E($true$Uϕ)
AX(ϕ) ("all successors satisfy ϕ) = $\neg$EX($\neg\phi$)
AG(ϕ) ("ϕ is invariant") = $\neg$EF($\neg\phi$)

CTL via fixpoints

The basic CTL operators can be defined as fixpoints of EX-formulas.

- $\mathsf{EF}(\phi) = (min\ Z : \phi \vee \mathsf{EX}(Z))$
- $\mathsf{E}(\phi \mathsf{W} \psi) = (max\ Z : \psi \vee (\phi \wedge \mathsf{EX}(Z)))$

Fixpoint formulas can be re-worked into a structurally simple notation: alternating automata.

Simple Alternating Automata (SAA)

A SAA is just like an NFA, except that the transition function δ maps a state to a *boolean formula* over atomic propositions and EX.

E.g., EF(P) has a 3-state automaton, with initial state q_0

$\delta(q_0) = q_1 \vee q_2$; $\delta(q_1) = P$; $\delta(q_2) = \mathsf{EX}(q_0)$

This is just the *parse graph* of $(min\ Z : P \vee \mathsf{EX}(Z))$. The (Büchi) acceptance set, F, is empty.

Theorem 0 *Every CTL formula can be represented by an SAA of proportional size.*

An Automaton-based proof system

To show that a program M with state set S and transition relation R satisfies an automaton property $(Q, \widehat{q}, \delta, F)$ we need, for each automaton state q:

- An *invariance* predicate, $\phi_q \subseteq S$, and
- A partial *rank function*, $\rho_q : S \to \mathbf{N}$

Roughly speaking, the invariance assertions state that any (reachable) state of M satisfying q falls within the "safe" set ϕ_q. The rank function marks the "distance" to reaching a Büchi state; it is re-set when the distance is 0.

Conditions for a valid Proof

☐ *Consistency:* ρ_q is defined for every state in ϕ_q

☐ *Initiality:* Every initial state of M satisfies $\phi_{\widehat{q}}$

☐ *Safety and Progress:* Based on $\delta(q)$

- l (a literal): $\phi_q(s) \Rightarrow l(s)$, for all s.
- $(\vee j : q_j)$: (similarly for $\wedge$)
  $\phi_q(s) \Rightarrow (\exists j : \phi_{q_j}(s) \wedge (\rho_{q_j}(s) <_q \rho_q(s)))$
- EX(r): (similarly for AX)
  $\phi_q(s) \Rightarrow (\exists t : sRt : \phi_r(t) \wedge (\rho_r(t) <_q \rho_q(s)))$

The relation $a <_q b = \mathbf{if}\ q \notin F\ \mathbf{then}\ a < b\ \mathbf{else}\ true$

Progress and safety have to be checked together because of the EX and $\vee$ operators.

Generating a Proof-I

Key: model check with automata instead of CTL

1. Turn CTL specification into a simple automaton
2. Form an AND-OR product graph of the program M and automaton A
3. Check the canonical property: does Player I have a winning strategy?

$$W_I = max\ Z; min\ Y :$$
$$tt \vee$$
$$(OR \wedge (F \Rightarrow \mathsf{EX}(Z)) \wedge (\neg F \Rightarrow \mathsf{EX}(Y))) \vee$$
$$(AND \wedge (F \Rightarrow \mathsf{AX}(Z)) \wedge (\neg F \Rightarrow \mathsf{AX}(Y)))$$

Generating a Proof-II

Now set:

1. the invariant ϕ_q to be $\{s : (s, q) \in W_I\}$
2. the rank $\rho_q(s)$ to the index of the *earliest* stage for Y where (s, q) is added, during the *last* Z iteration.

This works!

Theorem 1 *The proof system is sound and (relatively) complete.*

Generating Proofs-IV

Problem: we do not know before-hand whether the check succeeds or fails.

Immediate Solution: Generate proofs *after* normal model checking. (this requires two runs of the model checker)

Better Solution? Exploit duality. If W_I fails to hold of all initial states, then its dual, W_{II}, holds of some initial state. So keep approximations for both Y and Z, and use whichever is appropriate at the end.

A Simple Example

2-process, Atomic Bakery Protocol

var $st_1, st_2 : \{N, W, C\}$
(* N="Non-critical", W="Waiting", C="Critical" *)
var y_1, y_2 : **natural**
initially $(st_1 = N) \ \wedge \ (y_1 = 0) \ \wedge \ (st_2 = N) \ \wedge \ (y_2 = 0)$

$$
\begin{array}{ll}
wait_1 & st_1 = N \ \hookrightarrow \ st_1, y_1 \ := \ W, y_2 + 1 \\
enter_1 & st_1 = W \ \wedge \ (y_2 = 0 \ \vee \ y_1 \leq y_2) \ \hookrightarrow \ st_1 \ := \ C \\
release_1 & st_1 = C \ \hookrightarrow \ st_1, y_1 \ := \ N, 0 \\
 & \\
wait_2 & st_2 = N \ \hookrightarrow \ st_2, y_2 \ := \ W, y_1 + 1 \\
enter_2 & st_2 = W \ \wedge \ (y_1 = 0 \ \vee \ y_2 < y_1) \ \hookrightarrow \ st_2 \ := \ C \\
release_2 & st_2 = C \ \hookrightarrow \ st_2, y_2 \ := \ N, 0
\end{array}
$$

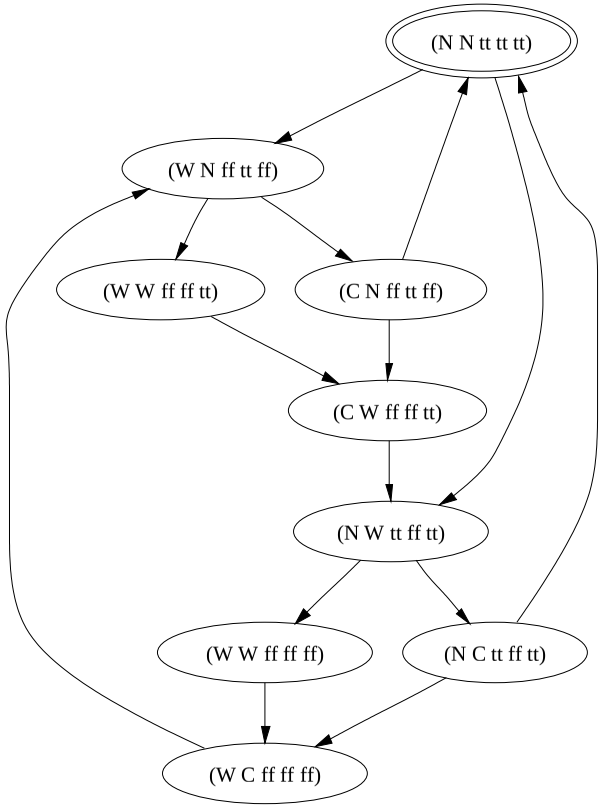
The Abstracted Protocol

Abstraction: $b_1 = (y_1 = 0); b_2 = (y_2 = 0); b_3 = (y_1 \leq y_2)$

var $st_1, st_2 : \{N, W, C\}$
var b_1, b_2, b_3 : **boolean**
initially $(st_1 = N) \ \wedge \ b_1 \ \wedge \ (st_2 = N) \ \wedge \ b_2 \ \wedge \ b_3$

$wait_1$	$st_1 = N \ \hookrightarrow \ st_1, b_1, b_2, b_3 \ := \ W, false, b_2, false$
$enter_1$	$st_1 = W \ \wedge \ (b_2 \ \vee \ b_3) \ \hookrightarrow \ st_1, b_1, b_2, b_3 \ := \ C, b_1, b_2, b_3$
$release_1$	$st_1 = C \ \hookrightarrow \ st_1, b_1, b_2, b_3 \ := \ N, true, b_2, true$
$wait_2$	$st_2 = N \ \hookrightarrow \ st_2, b_1, b_2, b_3 \ := \ W, b_1, false, true$
$enter_2$	$st_2 = W \ \wedge \ (b_1 \ \vee \ \neg b_3) \ \hookrightarrow \ st_2, b_1, b_2, b_3 \ := \ C, b_1, b_2, b_3$
$release_2$	$st_2 = C \ \hookrightarrow \ st_2, b_1, b_2, b_3 \ := \ N, b_1, true, b_1$

Abstract Proof



For the mutual exclusion property $\phi = \mathsf{AG}(\neg(C_1 \wedge C_2))$, the invariants are just the set of reachable states.

Concretizing this Proof

Let ξ be a simulation relation from M to $\overline{M}$. A proof (ϕ, ρ) on $\overline{M}$ can be concretized to a proof (ϕ', ρ') on M by letting

$\phi'_q(s) \equiv (\exists t : s \xi t : \phi_q(t))$, and
$\rho'_q(s) = (\min t : s \xi t \wedge \phi_q(t) : \rho_q(t))$

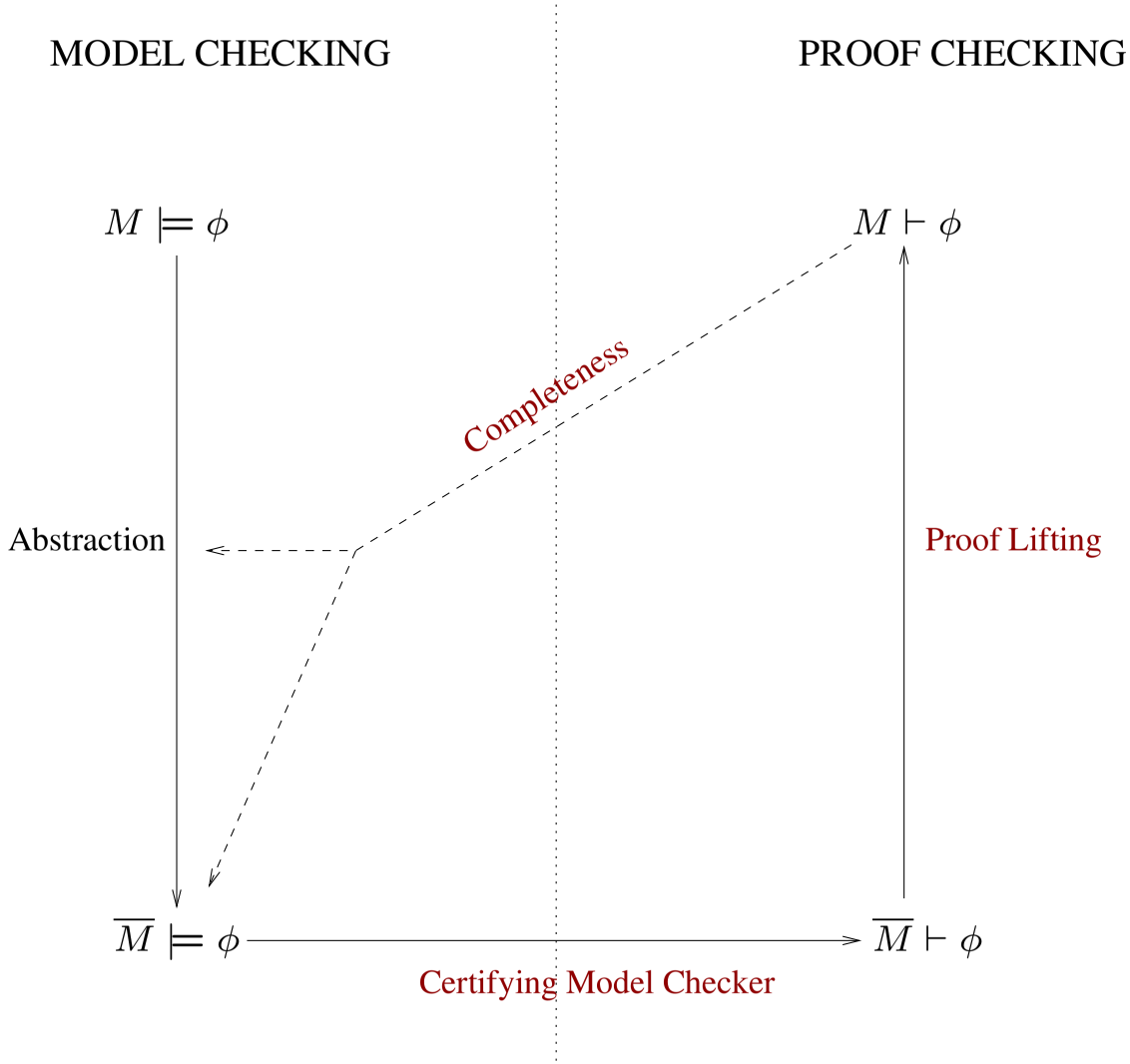
So:

$\phi'_q(st_1, st_2, y_1, y_2)$
$=$ (by definition)
$(\exists b_1, b_2, b_3 : b_1 \equiv (y_1 = 0) \wedge b_2 \equiv (y_2 = 0) \wedge b_3 = (y_1 \leq y_2) \wedge$
$\quad \phi_q(st_1, st_2, b_1, b_2, b_3))$
$=$ (simplifying)
$\phi_q(st_1, st_2, (y_1 = 0), (y_2 = 0), (y_1 \leq y_2))$

Summary: Proof Generation

- It is possible to design a model checker which generates an independently checkable proof of its results.
- This can be done quite easily: COSPAN modification (experimental) about 200 lines of C.
- Generated proofs have several applications ... and perhaps some as-yet-unknown ones!

Abstraction ∘ Model Checking = Deductive Proof



II. Completeness of Verification via Abstraction (joint work with Dennis Dams)

Given: Program M, property ϕ; to check $M \models \phi$
Construct Abstraction: a **finite** program $\overline{M}$
Model Check: whether $\overline{M} \models \phi$

An *Abstraction Framework* specifies the precise relationship between M and $\overline{M}$.

Soundness: for any M, ϕ: if $\overline{M} \models \phi$, then $M \models \phi$

Completeness: for any M, ϕ: if $M \models \phi$, there **exists** an abstraction $\overline{M}$ such that $\overline{M} \models \phi$

Summary of New Results

For properties expressed in branching time temporal logics (e.g., CTL, CTL*, or the μ-calculus)

* **Negative:** Several well-studied abstraction frameworks are *incomplete*. Examples: bisimulation [Milner71], modal transition system refinement [Larsen-Thomsen88]. This holds even with enhancements such as *fairness* or *stuttering*.

* **Positive:** A simple extension of modal transition systems with new *focus* operations gives rise to a complete framework.

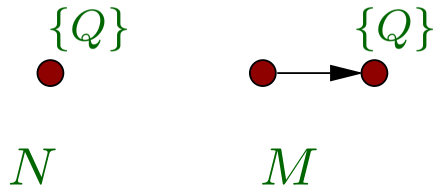
This is intimately connected to the representation of properties by finite tree automata.

Completeness and "Small Model" Theorems

Small Model Theorem [Hossley-Rackoff 72, Emerson85]: Any satisfiable property of the μ-calculus has a finite model.

Why doesn't this settle the question?
... because the small model need not abstract M.

Example:



N is a small model for the property "there is a reachable Q-state"

Bu N and M are unrelated by, say, simulation or modal refinement.

Modal Transition Systems [Larsen-Thomsen 1988]

A (Kripke) MTS is a transition system with

- **two** transition relations: *may* (over-approximate) and *must* (under-approximate) transitions, with $must \subseteq may$

- a **3-valued** ($true, false, \perp$) propositional valuation at states

For temporal logics, *existential path* modalities (e.g., EX) are interpreted over must-transitions; *universal path* modalities (e.g., AX) over may-transitions.

The outcome of model checking is also 3-valued.

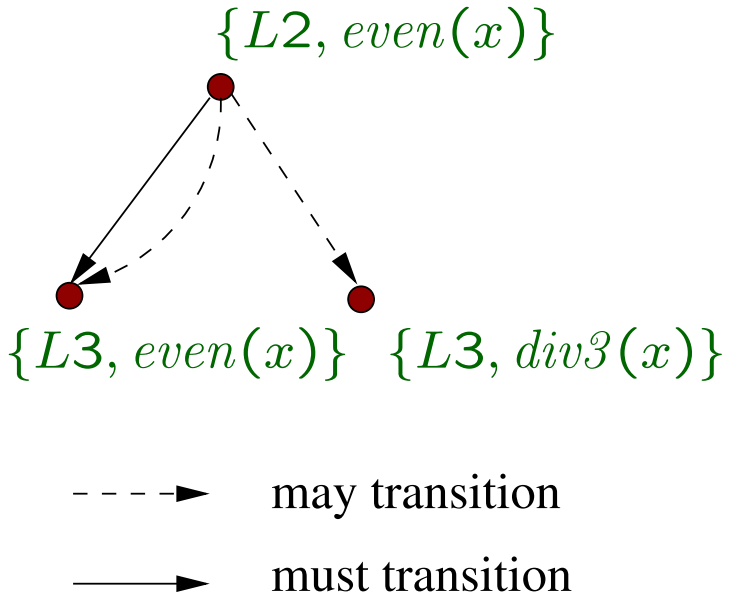
Abstraction with MTS's

If $c \sqsubseteq a$ then:

– $\forall c' : \; c \longrightarrow c' \;\Rightarrow\; (\exists a' : \; a \overset{may}{\longrightarrow} a' \;\wedge\; c' \sqsubseteq a')$

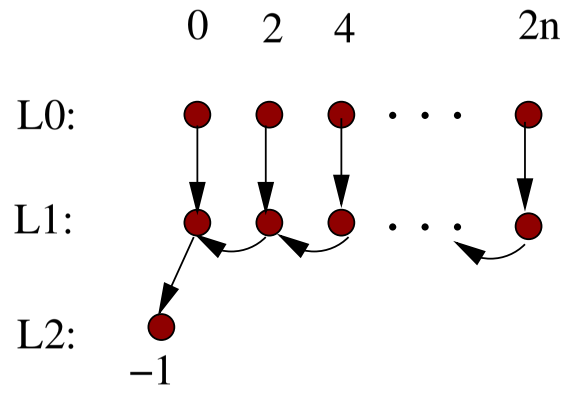
– $\forall a' : \; a \overset{must}{\longrightarrow} a' \;\Rightarrow\; (\exists c' : \; c \longrightarrow c' \;\wedge\; c' \sqsubseteq a')$

```
Program M
  integer x;
  L1: {x is even}
  L2: if (*)
        then x := x+2
        else x := x+4;
  L3:
```



Incompleteness

```
Program M
 L0: initially even(x)
 L1: while (x > 0)
       do x := x-2 od;
 L2: x := -1
```



Let ϕ=E($even(x)$W($x < 0$)).

Theorem 2 *No* **finite** *MTS abstracts* M *and satisfies* ϕ.

Proof by contradiction. The property holds for must-paths in $\overline{M}$; so either (i) $even(x)$ holds forever, or (ii) by *finiteness*, x is negative within a *bounded* number of steps. The must-abstraction enforces these properties at every initial state of M, a contradication!

Consequences and Variations

(Bi-)simulation is a special case of MTS refinement. Hence,

Corollary 0 *Abstraction with reverse simulation or bisimulation is incomplete for existential CTL properties.*

With a slight modification to the example:

Theorem 3 *Abstraction by MTS's with fairness or stuttering is also incomplete for existential CTL properties.*

A more elaborate property shows that the same results can be obtained even if M has a single initial state.

State-of-the-art for Completeness

* **Model Abstraction**: abstract the model, preserve the property

– ACTL,ACTL*: fair simulation [Grumberg-Long 1994, Kupferman-Vardi 1997]
– μ-calculus: *fair Focused Transition System abstraction*

* **Game Abstraction**: abstract the model-checking game, preserve the winning condition.

– linear-time: fair simulation [Uribe 1999, Kesten-Pnueli 2000, Kesten-Pnueli-Vardi 2001]
– μ-calculus: fair alternating refinement+choice [Namjoshi 2003]

The Need for Focus Operations

Transition $a \overset{must}{\longrightarrow} b$ exists only if **every** $c : c \sqsubseteq a$ has a transition to a state abstracted by b.

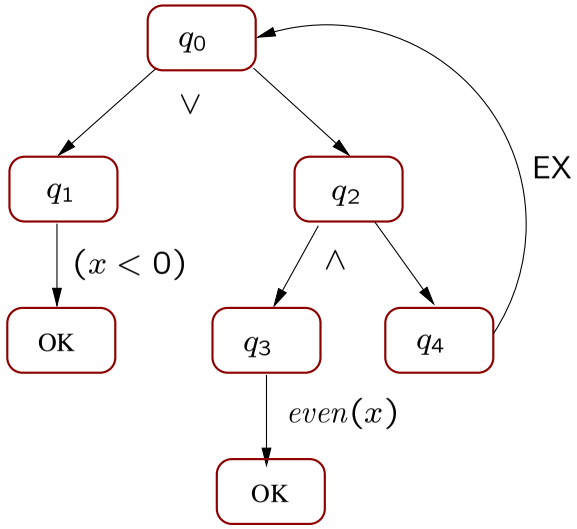
This forces any abstract MTS for our example to be infinite. E.g., $\overline{L1 : even(x)} \overset{must}{\not\longrightarrow} \overline{L2 : (x < 0)}$; so the source must be split; say to $\overline{L1 : (x < 0)}$, $\overline{L1 : (x \geq 0) \wedge even(x)}$.

But again $\overline{L1 : (x \geq 0) \wedge even(x)} \overset{must}{\not\longrightarrow} \overline{(x < 0)}$.

Can one somehow *relax* the must-transition definition? (Such a relaxation must preserve soundness.)

Alternating Automata

An alternating automaton for $\mathsf{E}(even(x)\mathsf{W}(x < 0))$



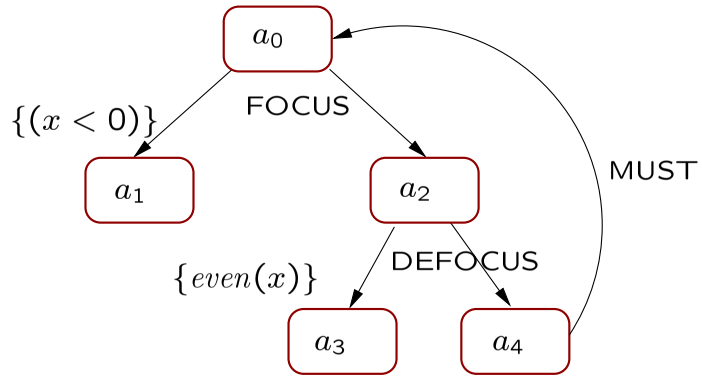
During model checking, each automaton state is associated with a set of program states.

Can an automaton be viewed as an abstract transition system?

Focus Steps

A *focus* step splits an abstract state into a set of more precise abstract states (case-splitting).

A Focused Transition System (FTS) is an MTS with focus and (dual) de-focus steps. For our example:



$$a_0 : \overline{L0, L1 : even(x), L2 : (x < 0)}$$
$$a_1 : \overline{L2 : (x < 0)}$$
$$a_2 : \overline{L0, L1 : even(x)}$$
$$a_3 : \overline{L0, L1 : even(x)}$$
$$a_4 : \overline{L0, L1 : even(x)}$$

Note the similarity to the automaton — this is no accident.

Completeness via Automata

Theorem 4 *For any M and any μ-calculus property ϕ, if $M \models \phi$, there is a finite FTS $\overline{M}$ such that $\overline{M}$ both abstracts M and satisfies ϕ.*

The FTS $\overline{M}$ may be obtained by: (i) converting ϕ to a *finite* alternating tree automaton A_ϕ, then (ii) converting A_ϕ to an FTS $\widehat{A}_\phi$ (roughly) as follows.

AX-move $\Rightarrow$ may transition
EX-move $\Rightarrow$ must transition
$\vee$-move $\Rightarrow$ focus transition
$\wedge$-move $\Rightarrow$ de-focus transition
acceptance condition $\Rightarrow$ fairness condition

Maximal Models

Notice that $\overline{M} = \hat{A}_\phi$ is independent of M! Thus, $\hat{A}_\phi$ is a **maximal model** for ϕ

By results of [Emerson-Jutla 1991], this maximal model has size *linear* in the size of ϕ.

Maximal model results for ACTL, ACTL* [Grumberg-Long 1994, Kupferman-Vardi 1997] require exponential-size models.

Maximal models reduce model checking to simulation-checking.

Completeness: Summary

- May-Must abstraction *does not guarantee* the existence of finite abstractions for existential temporal properties.

- The key to obtaining completeness seems to be a notion of ϵ-state-splitting we call a *focus* step.

- FTS's are intimately connected to alternating tree automata. It turns out [Dams-Namjoshi, VMCAI 2005] that non-deterministic automata suffice. In effect: transition systems + fairness + *choice*

- FTS's also ensure more precision in must-abstractions. (cf. [de Alfaro-Godefroid-Jagadeesan, LICS 2004])

To sum up

Model Checking and Proof Checking are closely linked, with Abstraction as the "glue".

(Partial) Reference List

I. From Model Checking to Proof Checking

[Stevens-Stirling, TACAS 1998] *Practical Model-Checking Using Games*

[Namjoshi, CAV 2001] *Certifying Model Checkers*

[Peled-Zuck, SPIN 2001] *From Model Checking to a Temporal Proof*

[Peled-Pnueli-Zuck, FSTTCS 2001] *From Falsification to Verification*

[Clarke-Jha-Lu-Veith, LICS 2002] *Tree-like Counterexamples in Model Checking*

[Tan-Cleaveland, CAV 2002] *Evidence-Based Model Checking*

[Henzinger-Jhala-Majumdar-Necula-Sutre-Weimer, CAV 2002] *Temporal-Safety Proofs for Systems Code*

[Gurfinkel-Chechik, TACAS 2003] *Proof-like counterexamples*

[Namjoshi, VMCAI 2003] *Lifting Temporal Proofs through Abstractions*

[Namjoshi, CAV 2004] *An Efficiently Checkable, Proof-Based Formulation of Vacuity in Model Checking*

Reference List-II

II. ... and Back

[Uribe, Thesis 2000] *Abstraction-Based Deductive-Algorithmic Verification of Reactive Systems*

[Kesten-Pnueli, Inf. Comp. 2000] *Verification by augmented finitary abstraction*

[Namjoshi, CAV 2003] *Branching-Time Abstraction*

[Dams-Namjoshi, LICS 2004] *The Existence of Finite Abstractions for Branching Time Model Checking*

[Dams-Namjoshi, VMCAI 2005] *Automata as Abstractions*

... Additional Slides ...

FTS's and Disjunctive MTS's [Larsen-Xinxin 1990]

DMTS's introduced to guarantee a solution to CCS equations of the form $\{C_i(X) = E_i\}$

DMTS's split a must-transition into cases: instead of $a \stackrel{must}{\longrightarrow} b$, allow $a \stackrel{must}{\longrightarrow} \{B_0, B_1, \ldots\}$ where the B_i are *sets* of abstract states.

Re-discovered in [Shoham-Grumberg 2004, de Alfaro-Godefroid-Jagadeesan 2004] for increasing the precision of abstractions.

FTS's are different in that one first splits state, then constructs ordinary $must$ transitions.