Model Checking with Computation Tree Logic

Joe Scott

February 17, 2012

Different Approaches to Verification

- Proof-Based
  - Representation:
    - ★ System description is a set of formulas Γ in a suitable logic.
    - ★ Specification is another formula ϕ
  - Find a proof that $\Gamma \vdash \phi$
    - ★ Deductive
    - ★ Usually requires guidance from the user

- Model-Based
  - Representation:
    - ★ System description is a model $\mathcal{M}$ of a suitable logic.
    - ★ Specification still a formula ϕ
  - Determine whether $\mathcal{M} \models \phi$
    - ★ Algorithmic
    - ★ Automatic

Why model checking?

- Given a logical proof system that is sound and complete[1]:
  $\Gamma \vdash \phi$ (provability) holds iff $\Gamma \models \phi$ (semantic entailment).
- Semantic entailment means for all models $\mathcal{M}$:
  if for all $\psi \in \Gamma$ we have $\mathcal{M} \models \psi$, then $\mathcal{M} \models \phi$.

Intuition

A verification method based on a single model $\mathcal{M}$ should be simpler than a method based on a potentially infinite class of them.

[1] Of course, Hoare Logic is not complete.

Temporal Logic

- Classical propositional and predicate logics are static
  - formulas are always true or false
- In modal logic, truth is dynamic
  - models contain several states
  - a formula may be true in some states, false in others
- Temporal logic is a modal logic with a semantics based on "when"
  - a path is a sequence of time instances (states)

	Linear	Branching
Temporal model	set of paths	tree
Path Quantification	$\forall$ (implicit)	$\forall$ or $\exists$ (explicit)
	LTL	CTL
		CTL*

Model: Transition System

$$\mathcal{M} = \langle S, \longrightarrow, L \rangle$$

- (finite) set of states S
- transition relation $\longrightarrow$:
  - Binary relation on S
  - Every $s \in S$ has some $s' \in S$ such that $s \longrightarrow s'$
- labelling function $L : S \to \mathcal{P}(atoms)$

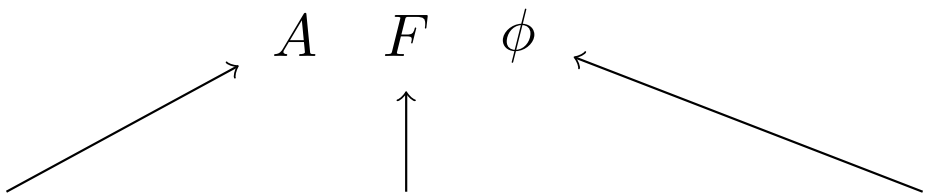
CTL Syntax

Valid formulas:

- True, False
- Any atomic proposition p
- For valid subformulas ϕ_1, ϕ_2:

$$\neg\phi_1,\ \phi_1 \wedge \phi_2,\ \phi_1 \vee \phi_2,\ \phi_1 \implies \phi_2,\ \ldots$$

- temporal formula:



path quantifier:
All paths
Exists a path

temporal operator:
ne**X**t state
some **F**uture state
all future states (**G**lobally)
all states **U**ntil (binary)

subformula

CTL Equivalences

- $\neg AF\phi \equiv EG\neg\phi$
- $\neg EF\phi \equiv AG\neg\phi$
- $\neg AX\phi \equiv EX\neg\phi$
- $AF\phi \equiv A[\top U\phi]$
- $EF\phi \equiv E[\top U\phi]$

Theorem (Adequate sets of CTL connectives[a])

[a] A. Martin, **Adequate sets of temporal connectives in CTL**. *Elec. Notes in Theor. Comp. Sc.* 52(1), 2001.

A set of temporal connectives in CTL is adequate iff it contains:

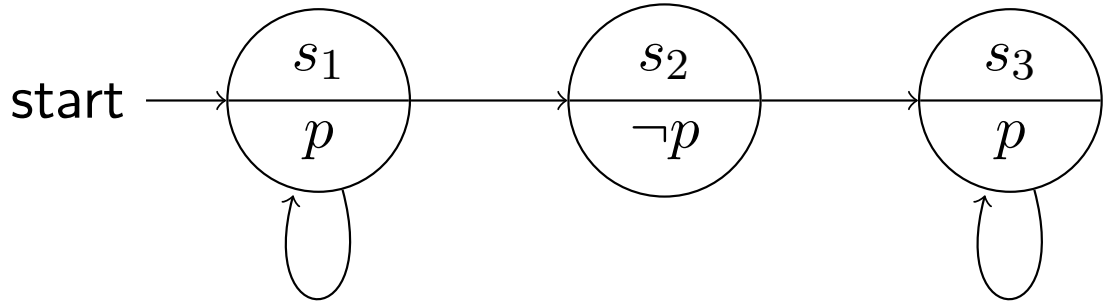
- *at least one of* $\{AX, EX\}$
- *at least one of* $\{EG, AF, AU\}$
- *EU*

Labelling Algorithm for $\{AF, EU, EX\}$

Starting from innermost subformulas:

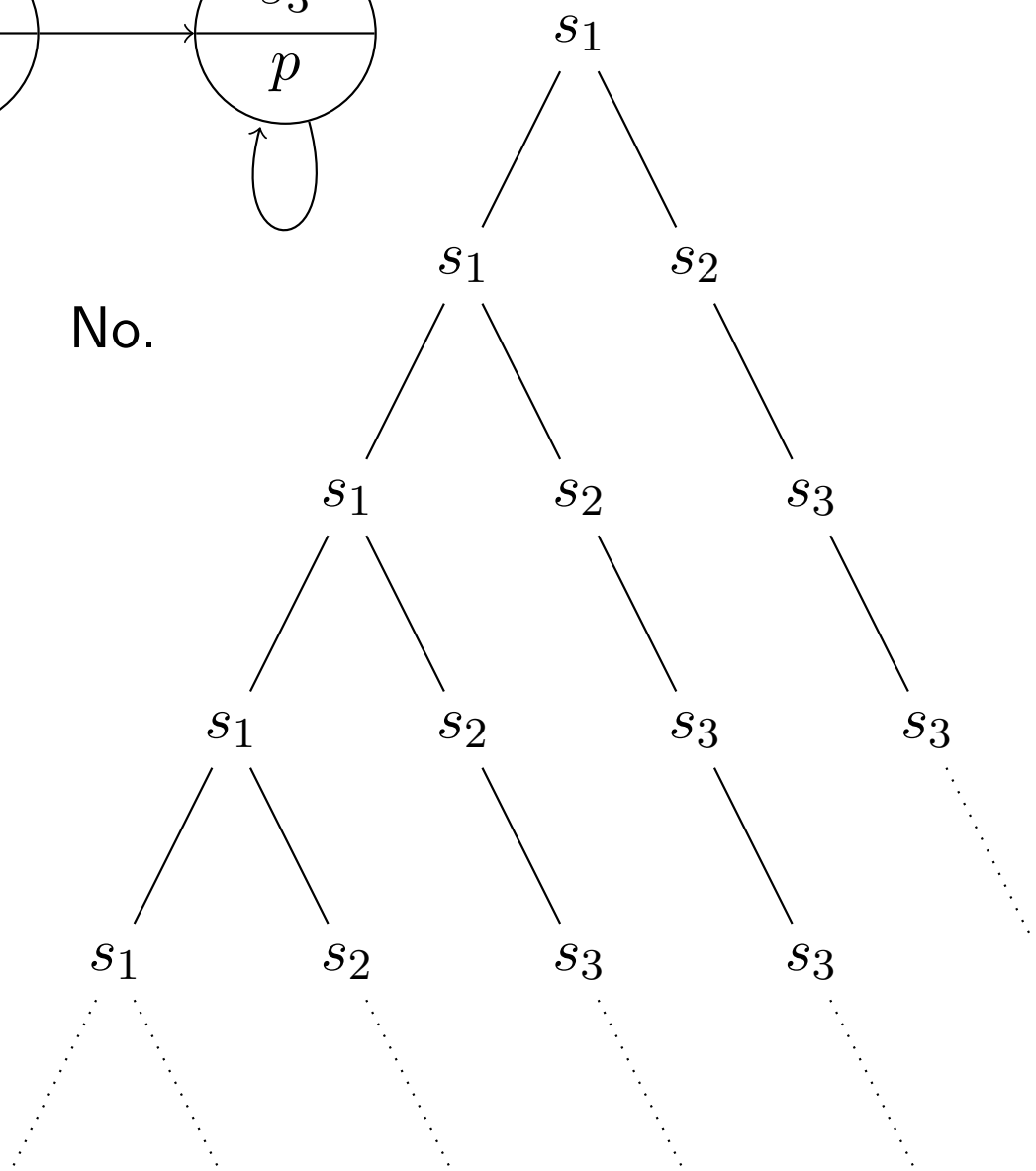
- $\bot$: do nothing
- p: label s if $p \in L(s)$
- $\phi_1 \wedge \phi_2$: label s if s is already labelled with ϕ_1 and ϕ_2
- $\neg\phi_1$: label all s not already labelled ϕ_1
- $AF\phi_1$:
  1. If any s is labelled ϕ_1, label it $AF\phi_1$
  2. Label any state $AF\phi_1$ if all its successor states are labelled $AF\phi_1$
  3. Repeat 2 until no change
- $E[\phi_1 U \phi_2]$:
  1. If any s is labelled ϕ_2, label it $E[\phi_1 U \phi_2]$
  2. For any s labelled ϕ_1, label s if it has a successor labelled $E[\phi_1 U \phi_2]$
  3. Repeat 2 until no change
- $EX\phi_1$: label any state that has a successor labelled ϕ_1

A final example



Does $AF\ AG\ p$ hold?

No.



LTL formula $F\ G\ p$ does hold.

References

- Today's paper:
  Clarke, Emerson, Sistla
  Automatic verification of finite-state concurrent systems using temporal logic specifications.
  ACM Trans. Program. Lang. Syst., 8:244–263. 1986

- Supplementary slides courtesy:
  `http://www.tn.refer.org/unesco/semestre6/CoursUNESCO_intro2.pdf`

- Other material based on:
  - *Logic in Computer Science*, Huth & Ryan, 2004.
  - *Principles of Model Checking*, Baier & Katoen, 2008.