universität innsbruck

SS 2024 lecture 9

Logic

Diana Gründlinger Aart Middeldorp Fabian Mitterwallner

Alexander Montag Johannes Niederhauser Daniel Rainer

Outline

1. **Summary of Previous Lecture**
2. **Post's Adequacy Theorem**
3. **Intermezzo**
4. **Model Checking**
5. **Branching-Time Temporal Logic (CTL)**
6. **CTL Model Checking Algorithm**
7. **Further Reading**

Definitions

- atomic formula: $P \mid P(t, \ldots, t)$
- literal is atomic formula or negation of atomic formula
- clause is set of literals $\{\ell_1, \ldots, \ell_n\}$
- clausal form is set of clauses $\{C_1, \ldots, C_m\}$, representing $\forall\,(C_1 \wedge \cdots \wedge C_m)$
- clauses C_1 and C_2 without common variables clash on literals $\ell_1 \in C_1$ and $\ell_2 \in C_2$ if ℓ_1 and ℓ_2^c are unifiable
- resolvent of clauses C_1 and C_2 clashing on literals $\ell_1 \in C_1$ and $\ell_2 \in C_2$ is clause

$$((C_1 \setminus \{\ell_1\}) \cup (C_2 \setminus \{\ell_2\}))\theta$$

  where θ is mgu of ℓ_1 and ℓ_2^c
- $C\sigma$ is factor of C if two or more literals in C have mgu σ

Resolution with Factoring

input: clausal form S

output: yes if S is satisfiable

no if S is unsatisfiable

∞ if S is satisfiable

① repeatedly add resolvents (renaming clauses if necessary) and factors

② return no as soon as empty clause $\square$ is derived

③ return yes if all clashing clauses have been resolved and factoring produces no new clauses (modulo renaming)

Theorem

resolution with factoring is sound and complete:

clausal form S is unsatisfiable if and only if S admits refutation

Decision Problem (Church's Theorem)

instance: set of formulas Γ, first-order formula ψ

question: $\Gamma \vDash \psi$?

is undecidable even when $\Gamma = \varnothing$

Definition

set X of boolean functions is called adequate or functionally complete if every boolean function can be expressed using functions from X

Theorem (Algebraic Normal Form)

every boolean function $f\colon \{0,1\}^n \to \{0,1\}$ can be uniquely written as

$$f(x_1,\dots,x_n) = \bigoplus_{A \subseteq \{1,\dots,n\}} c_A \cdot \prod_{i \in A} x_i$$

with $c_A \in \{0,1\}$ for all $A \subseteq \{1,\dots,n\}$

Part I: Propositional Logic

algebraic normal forms, binary decision diagrams, conjunctive normal forms, DPLL, Horn formulas, natural deduction, Post's adequacy theorem, resolution, SAT, semantics, sorting networks, soundness and completeness, syntax, Tseitin's transformation

Part II: Predicate Logic

natural deduction, quantifier equivalences, resolution, semantics, Skolemization, syntax, undecidability, unification

Part III: Model Checking

adequacy, branching-time temporal logic, CTL*, fairness, linear-time temporal logic, model checking algorithms, symbolic model checking

Outline

Theorem (Post's Adequacy Theorem)

set X of boolean functions is adequate if and only if following conditions hold:

1. there exists $f \in X$ such that $f(0,\dots,0) \neq 0$
2. there exists $f \in X$ such that $f(1,\dots,1) \neq 1$
3. there exists $f \in X$ which is not monotone
4. there exists $f \in X$ which is not self-dual
5. there exists $f \in X$ which is not affine

Definitions

boolean function f is

- monotone if $f(x_1,\dots,x_n) \leqslant f(y_1,\dots,y_n)$ for all $x_1 \leqslant y_1, \dots, x_n \leqslant y_n$
- self-dual if $f(x_1,\dots,x_n) = \overline{f(\overline{x}_1,\dots,\overline{x}_n)}$
- affine if $f(x_1,\dots,x_n) = c_0 \oplus c_1x_1 \oplus \cdots \oplus c_nx_n$ for some $c_0,\dots,c_n \in \{0,1\}$

Lemma

boolean function f is not monotone if and only if

$$f(b_1, \dots, b_{i-1}, x, b_{i+1}, \dots, b_n) = \overline{x} \qquad \text{for all } x \in \{0,1\}$$

for some i and $b_1, \dots, b_{i-1}, b_{i+1}, \dots, b_n \in \{0,1\}$

Lemma

boolean function f is not self-dual if and only if

$$f(b_1, \dots, b_n) = f(\overline{b}_1, \dots, \overline{b}_n)$$

for some $b_1, \dots, b_n \in \{0,1\}$

Remark

boolean function f is affine if and only if algebraic normal form of f is linear

Examples

	$-$	$\cdot$	$+$	$=$	$\oplus$	$\mid$	0	1
$f(0,\ldots,0) \neq 0$	✓	×	×	✓	×	✓	×	✓
$f(1,\ldots,1) \neq 1$	✓	×	×	×	✓	✓	✓	×
not monotone	✓	×	×	✓	✓	✓	×	×
not self-dual	×	✓	✓	✓	✓	✓	✓	✓
not affine	×	✓	✓	×	×	✓	×	×

Definitions

boolean function f is

- monotone if $f(x_1,\ldots,x_n) \leqslant f(y_1,\ldots,y_n)$ for all $x_1 \leqslant y_1, \ldots, x_n \leqslant y_n$
- self-dual if $f(x_1,\ldots,x_n) = \overline{f(\overline{x}_1,\ldots,\overline{x}_n)}$
- affine if $f(x_1,\ldots,x_n) = c_0 \oplus c_1x_1 \oplus \cdots \oplus c_nx_n$ for some $c_0,\ldots,c_n \in \{0,1\}$

Theorem (Post's Adequacy Theorem)

set X of boolean functions is adequate if and only if following conditions hold:

1. $\exists\, f_1 \in X$ such that $f_1(0,\dots,0) \neq 0$
2. $\exists\, f_2 \in X$ such that $f_2(1,\dots,1) \neq 1$
3. $\exists\, f_3 \in X$ which is not monotone
4. $\exists\, f_4 \in X$ which is not self-dual
5. $\exists\, f_5 \in X$ which is not affine

Proof ($\Longleftarrow$)

- first task: define 0, 1, $\overline{x}$
- define $g(x) = f_1(x,\dots,x)$ and $h(x) = f_2(x,\dots,x)$
- $g(x) = 1$ or $g(x) = \overline{x}$ and $h(x) = 0$ or $h(x) = \overline{x}$
- we distinguish four cases:
  1. $g(x) = 1$ and $h(x) = \overline{x}$
  2. $g(x) = \overline{x}$ and $h(x) = 0$
  3. $g(x) = 1$ and $h(x) = 0$
  4. $g(x) = \overline{x}$ and $h(x) = \overline{x}$

Proof ($\Longleftarrow$)

- first task: define 0, 1, $\overline{x}$

① $g(x) = 1$ and $h(x) = \overline{x}$ $\quad\quad h(g(x)) = 0$

② $g(x) = \overline{x}$ and $h(x) = 0$ $\quad\quad g(h(x)) = 1$

③ $g(x) = 1$ and $h(x) = 0$

there exist $i \in \{1, \dots, m\}$ and $b_1, \dots, b_{i-1}, b_{i+1}, \dots, b_m \in \{0, 1\}$ such that

$$f_3(b_1, \dots, b_{i-1}, x, b_{i+1}, \dots, b_m) = \overline{x}$$

$b_j = g(x)$ or $b_j = h(x)$ for $j \neq i$

so $\overline{x}$ is defined using f_3, g, h

❸ there exists $f_3 \in X$ which is not monotone

Proof ($\Longleftarrow$)

- first task: define 0, 1, $\overline{x}$

④ $g(x) = \overline{x}$ and $h(x) = \overline{x}$

there exists $b_1, \dots, b_k \in \{0,1\}$ such that $f_4(\overline{b}_1, \dots, \overline{b}_k) = f_4(b_1, \dots, b_k)$

define $i(x) = f_4(x \oplus b_1, \dots, x \oplus b_k)$

$x \oplus b_j = x$ or $x \oplus b_j = \overline{x} = g(x)$, so $i(x)$ is defined using f_4 and g

$i(x) = 0$ or $i(x) = 1$

$g(i(x)) = 1$ or $g(i(x)) = 0$

④ there exists $f_4 \in X$ which is not self-dual

Proof ($\Longleftarrow$)

- second task: define xy

  there exist g_1, g_2, g_3, g_4 such that (wlog)

  $$f_5(x_1,\dots,x_l) = x_1x_2\,g_1(x_3,\dots,x_l) \oplus x_1\,g_2(x_3,\dots,x_l) \oplus x_2\,g_3(x_3,\dots,x_l) \oplus g_4(x_3,\dots,x_l)$$

  with $g_1(x_3,\dots,x_l) \neq 0$

  there exist $c_3,\dots,c_l \in \{0,1\}$ such that $g_1(c_3,\dots,c_l) = 1$

  define $c = g_2(c_3,\dots,c_l)$, $d = g_3(c_3,\dots,c_l)$, $e = g_4(c_3,\dots,c_l)$

  $f_5(x_1,x_2,c_3,\dots,c_l) = x_1x_2 \oplus x_1c \oplus x_2d \oplus e$

  define $h(x,y) = f_5(x \oplus d, y \oplus c, c_3,\dots,c_l) \oplus cd \oplus e$

  $h(x,y) = (x \oplus d)(y \oplus c) \oplus (x \oplus d)c \oplus (y \oplus c)d \oplus e \oplus cd \oplus e = xy$

5. there exists $f_5 \in X$ which is not affine

Remark

proof of "if direction" is constructive

Demo

BoolTool

by Patrick Muxel (2004), Philipp Ruff (2006), Caroline Terzer (2006), Markus Plattner (2007), Elias Zischg (2012)

BoolTool Reloaded

by Martin Neuner (2023)

Proof sketch ($\Longrightarrow$)

- suppose X has no functions that satisfy condition ⓘ
- claim: all functions constructed from X violate condition ⓘ
- X cannot be adequate because $x \mid y$ cannot be expressed

Outline

Question

Which of the following statements are true ?

A If $f(1,\ldots,1) = 0$ and f is monotone then $f(x_1,\ldots,x_n) = 0$

B A set containing only constants and unary functions can be adequate.

C $\{\overline{\vee}\}$ is adequate where $x \mathbin{\overline{\vee}} y = \overline{x \vee y}$.

D There are more affine than non-affine binary boolean functions.

Outline

Formal Verification comprises

- framework for modeling systems (description language)
- specification language for describing properties to be verified
- verification method to establish whether description of system satisfies specification

Model Checking

automatic formal verification approach for concurrent systems based on temporal logic

Temporal Logic

- formulas are not statically true or false in model
- models of temporal logic contain several states and truth is dynamic
- formula can be true in some states and false in others

Model Checking

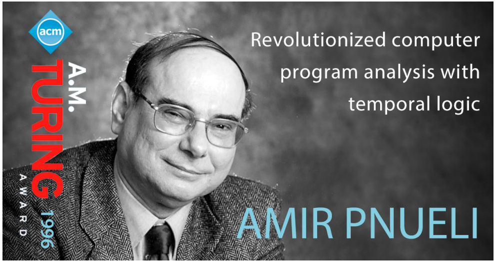
- models are transition systems $\mathcal{M}$
- properties are formulas φ in temporal logic
- model checker determines whether $\mathcal{M} \vDash \varphi$ is true or not

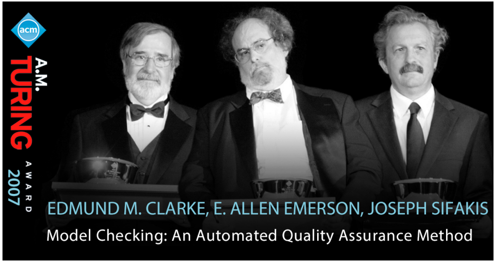
Two Temporal Logics

- computation tree logic (CTL) — lectures 9 and 10
- linear-time temporal logic (LTL) — lectures 10 and 11

Impact

both logics have been proven to be extremely fruitful in verifying hardware and communication protocols, and are increasingly applied to software verification





ACM Turing Awards

1996 Amir Pnueli

2007 Edmund M. Clarke, E. Allen Emerson, Joseph Sifakis

Outline

Definition

- CTL (computation tree logic) formulas are built from
  - atoms $p,\ q,\ r,\ p_1,\ p_2,\ \ldots$
  - logical connectives $\bot,\ \top,\ \neg,\ \wedge,\ \vee,\ \rightarrow$
  - temporal connectives AX, EX, AF, EF, AG, EG, AU, EU

  according to following BNF grammar:

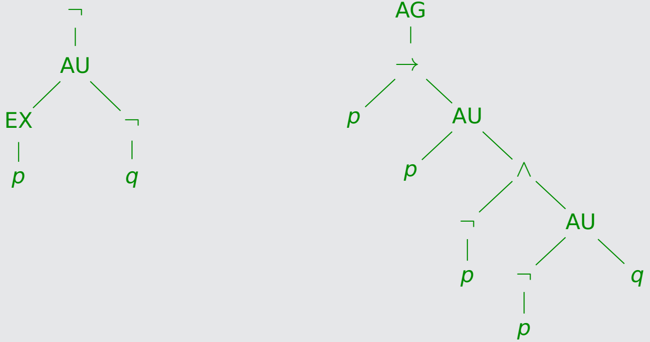
$$\varphi \ ::= \bot \mid \top \mid p \mid (\neg\varphi) \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid (\varphi \rightarrow \varphi) \mid (\mathsf{AX}\,\varphi) \mid (\mathsf{EX}\,\varphi) \mid$$
$$(\mathsf{AF}\,\varphi) \mid (\mathsf{EF}\,\varphi) \mid (\mathsf{AG}\,\varphi) \mid (\mathsf{EG}\,\varphi) \mid \mathsf{A}[\varphi\,\mathsf{U}\,\varphi] \mid \mathsf{E}[\varphi\,\mathsf{U}\,\varphi]$$

- notational conventions:
  - binding precedence $\neg,\ \mathsf{AX},\ \mathsf{EX},\ \mathsf{AF},\ \mathsf{EF},\ \mathsf{AG},\ \mathsf{EG}\ >\ \wedge,\ \vee\ >\ \rightarrow,\ \mathsf{AU},\ \mathsf{EU}$
  - omit outer parentheses
  - $\rightarrow,\ \wedge,\ \vee$ are right-associative

Example

formula $\neg \mathsf{A}[\,\mathsf{EX}\,p\,\mathsf{U}\,\neg q\,]$ $\quad$ $\mathsf{AG}\,(p \rightarrow \mathsf{A}[\,p\,\mathsf{U}\,\neg p \wedge \mathsf{A}[\neg p\,\mathsf{U}\,q]\,])$

parse tree



A	∀ paths	G	∀ states	globally	X		next state
E	∃ path	F	∃ state	future	U		until

Outline

1. **Summary of Previous Lecture**
2. **Post's Adequacy Theorem**
3. **Intermezzo**
4. **Model Checking**
5. **Branching-Time Temporal Logic (CTL)**
   Syntax Semantics
6. **CTL Model Checking Algorithm**
7. **Further Reading**

Definition

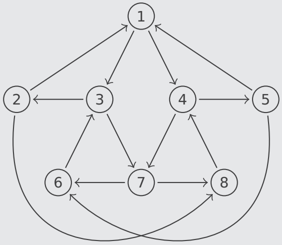
transition system (model) is triple $\mathcal{M} = (S, \rightarrow, L)$ with

① set of states S

② transition relation $\rightarrow \subseteq S \times S$ such that $\forall s \in S \;\; \exists t \in S$ with $s \rightarrow t$ ("no deadlock")

③ labelling function $L \colon S \rightarrow \mathcal{P}(\text{atoms})$

Example



model $\mathcal{M} = (S, \rightarrow, L)$

$$S = \{1, 2, 3, 4, 5, 6, 7, 8\}$$

$$L(1) = \{I_A, I_B\} \qquad L(5) = \{I_A, P_B\}$$
$$L(2) = \{P_A, I_B\} \qquad L(6) = \{R_A, P_B\}$$
$$L(3) = \{R_A, I_B\} \qquad L(7) = \{R_A, R_B\}$$
$$L(4) = \{I_A, R_B\} \qquad L(8) = \{P_A, R_B\}$$

Definition

satisfaction of CTL formula φ in state $s \in S$ of model $\mathcal{M} = (S, \to, L)$

$$\mathcal{M}, s \vDash \varphi$$

is defined by induction on φ:

$$\mathcal{M}, s \vDash \top \qquad \mathcal{M}, s \nvDash \bot \qquad \mathcal{M}, s \vDash \varphi \wedge \psi \iff \mathcal{M}, s \vDash \varphi \text{ and } \mathcal{M}, s \vDash \psi$$

$$\mathcal{M}, s \vDash p \iff p \in L(s) \qquad \mathcal{M}, s \vDash \varphi \vee \psi \iff \mathcal{M}, s \vDash \varphi \text{ or } \mathcal{M}, s \vDash \psi$$

$$\mathcal{M}, s \vDash \neg\varphi \iff \mathcal{M}, s \nvDash \varphi \qquad \mathcal{M}, s \vDash \varphi \to \psi \iff \mathcal{M}, s \nvDash \varphi \text{ or } \mathcal{M}, s \vDash \psi$$

$$\mathcal{M}, s \vDash \mathrm{AX}\,\varphi \iff \forall \text{ paths } s = s_1 \to s_2 \to s_3 \to \cdots \quad \mathcal{M}, s_2 \vDash \varphi$$

$$\mathcal{M}, s \vDash \mathrm{EX}\,\varphi \iff \exists \text{ path } s = s_1 \to s_2 \to s_3 \to \cdots \quad \mathcal{M}, s_2 \vDash \varphi$$

$$\mathcal{M}, s \vDash \mathrm{AF}\,\varphi \iff \forall \text{ paths } s = s_1 \to s_2 \to s_3 \to \cdots \quad \exists\, i \geqslant 1 \quad \mathcal{M}, s_i \vDash \varphi$$

$$\mathcal{M}, s \vDash \mathrm{EF}\,\varphi \iff \exists \text{ path } s = s_1 \to s_2 \to s_3 \to \cdots \quad \exists\, i \geqslant 1 \quad \mathcal{M}, s_i \vDash \varphi$$

Definition (cont'd)

satisfaction of CTL formula φ in state $s \in S$ of model $\mathcal{M} = (S, \to, L)$

$$\mathcal{M}, s \vDash \varphi$$

is defined by induction on φ:

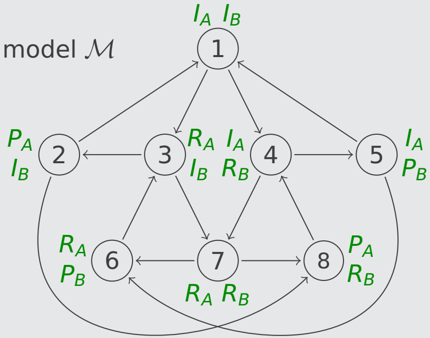
$$\mathcal{M}, s \vDash \mathsf{AG}\,\varphi \iff \forall \text{ paths } s = s_1 \to s_2 \to s_3 \to \cdots \quad \forall\, i \geqslant 1 \quad \mathcal{M}, s_i \vDash \varphi$$

$$\mathcal{M}, s \vDash \mathsf{EG}\,\varphi \iff \exists \text{ path } s = s_1 \to s_2 \to s_3 \to \cdots \quad \forall\, i \geqslant 1 \quad \mathcal{M}, s_i \vDash \varphi$$

$$\mathcal{M}, s \vDash \mathsf{A}[\varphi\,\mathsf{U}\,\psi] \iff \forall \text{ paths } s = s_1 \to s_2 \to s_3 \to \cdots$$
$$\exists\, i \geqslant 1 \quad \mathcal{M}, s_i \vDash \psi \text{ and } \forall\, j < i \quad \mathcal{M}, s_j \vDash \varphi$$

$$\mathcal{M}, s \vDash \mathsf{E}[\varphi\,\mathsf{U}\,\psi] \iff \exists \text{ path } s = s_1 \to s_2 \to s_3 \to \cdots$$
$$\exists\, i \geqslant 1 \quad \mathcal{M}, s_i \vDash \psi \text{ and } \forall\, j < i \quad \mathcal{M}, s_j \vDash \varphi$$

Example



$\mathcal{M}, 1 \nvDash I_A \wedge R_B$

$\mathcal{M}, 4 \vDash I_A \wedge R_B$

$\mathcal{M}, 1 \vDash \mathrm{AX}(R_A \vee R_B)$

$\mathcal{M}, 3 \nvDash \mathrm{AX}\, P_A$

$\mathcal{M}, 1 \vDash \mathrm{AF}(R_A \vee R_B)$

$\mathcal{M}, 5 \nvDash \mathrm{AF}\, R_B$

$\mathcal{M}, 1 \vDash \mathrm{AG}(R_A \to \mathrm{EF}\, P_A)$

$\mathcal{M}, 1 \nvDash \mathrm{AG}(R_A \to \mathrm{AF}\, P_A)$

$\mathcal{M}, 1 \vDash \neg \mathrm{A}[R_A \,\mathrm{U}\, P_A]$

$\mathcal{M}, 7 \vDash \mathrm{A}[P_A \,\mathrm{U}\, R_A]$

$\mathcal{M}, 1 \nvDash I_B \to P_A \vee R_B$

$\mathcal{M}, 2 \vDash I_B \to P_A \vee R_B$

$\mathcal{M}, 1 \nvDash \mathrm{EX}\, P_B$

$\mathcal{M}, 3 \vDash \mathrm{EX}\, P_A$

$\mathcal{M}, 1 \vDash \mathrm{EF}(R_A \wedge R_B)$

$\mathcal{M}, 5 \nvDash \mathrm{EF}(P_A \wedge P_B)$

$\mathcal{M}, 2 \vDash \mathrm{EG}(\neg P_A \to R_B)$

$\mathcal{M}, 2 \nvDash \mathrm{EG}\, P_A$

$\mathcal{M}, 1 \vDash \mathrm{EX}\,\mathrm{E}[R_A \,\mathrm{U}\, P_A]$

$\mathcal{M}, 7 \nvDash \mathrm{E}[P_A \wedge P_B \,\mathrm{U}\, I_A \vee I_B]$

Theorem

satisfaction of CTL formulas in finite models is decidable

Definition

CTL formulas φ and ψ are semantically equivalent $(\varphi \equiv \psi)$ if

$$\mathcal{M}, s \vDash \varphi \iff \mathcal{M}, s \vDash \psi$$

for all models $\mathcal{M} = (S, \rightarrow, L)$ and states $s \in S$

Theorem

$$\neg\, \mathsf{AF}\, \varphi \equiv \mathsf{EG}\, \neg\varphi \qquad \mathsf{AF}\, \varphi \equiv \mathsf{A}[\top\, \mathsf{U}\, \varphi]$$

$$\neg\, \mathsf{EF}\, \varphi \equiv \mathsf{AG}\, \neg\varphi \qquad \mathsf{EF}\, \varphi \equiv \mathsf{E}[\top\, \mathsf{U}\, \varphi]$$

$$\neg\, \mathsf{AX}\, \varphi \equiv \mathsf{EX}\, \neg\varphi \qquad \mathsf{A}[\varphi\, \mathsf{U}\, \psi] \equiv \neg(\mathsf{E}[\neg\psi\, \mathsf{U}\, (\neg\varphi \wedge \neg\psi)] \vee \mathsf{EG}\, \neg\psi)$$

Outline

1. Summary of Previous Lecture
2. Post's Adequacy Theorem
3. Intermezzo
4. Model Checking
5. Branching-Time Temporal Logic (CTL)
6. **CTL Model Checking Algorithm**
7. Further Reading

CTL Model Checking Algorithm ❶

input: • model $\mathcal{M} = (S, \rightarrow, L)$ and CTL formula φ

output: • $\{s \in S \mid \mathcal{M}, s \vDash \varphi\}$

label each state $s \in S$ by those subformulas of φ that are satisfied in s

$\top$	label every state		
$\bot$	label no state		
p	label s	$\Longleftrightarrow$	$p \in L(s)$
$\neg\varphi$	label s	$\Longleftrightarrow$	s is not labelled with φ
$\varphi \wedge \psi$	label s	$\Longleftrightarrow$	s is labelled with both φ and ψ
$\varphi \vee \psi$	label s	$\Longleftrightarrow$	s is labelled with φ or ψ
$\varphi \rightarrow \psi$	label s	$\Longleftrightarrow$	s is not labelled with φ or s is labelled with ψ
$\mathrm{AX}\,\varphi$	label s	$\Longleftrightarrow$	t is labelled with φ for all t with $s \rightarrow t$

CTL Model Checking Algorithm ❷

$\mathrm{EX}\,\varphi$ label s $\Longleftrightarrow$ t is labelled with φ for some t with $s \to t$

$\mathrm{AF}\,\varphi$ label s $\Longleftrightarrow$
① s is labelled with φ
② t is labelled with $\mathrm{AF}\,\varphi$ for all t with $s \to t$
③ repeat ② until no change

$\mathrm{EF}\,\varphi$ label s $\Longleftrightarrow$
① s is labelled with φ
② t is labelled with $\mathrm{EF}\,\varphi$ for some t with $s \to t$
③ repeat ② until no change

$\mathrm{AG}\,\varphi$
① label every s that is labelled with φ
② remove label from s $\Longleftrightarrow$ t is not labelled with $\mathrm{AG}\,\varphi$ for some t with $s \to t$
③ repeat ② until no change

CTL Model Checking Algorithm ③

$\mathsf{EG}\,\varphi$

① label every s that is labelled with φ

② remove label from s $\iff$ t is not labelled with $\mathsf{EG}\,\varphi$ for all t with $s \to t$

③ repeat ② until no change

$\mathsf{A}[\varphi\,\mathsf{U}\,\psi]$ label s $\iff$

① s is labelled with ψ

② s is labelled with φ and t with $\mathsf{A}[\varphi\,\mathsf{U}\,\psi]$ for all t with $s \to t$

③ repeat ② until no change

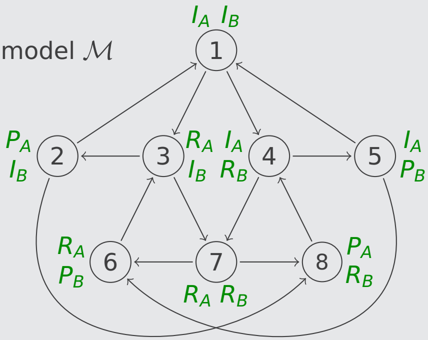
$\mathsf{E}[\varphi\,\mathsf{U}\,\psi]$ label s $\iff$

① s is labelled with ψ

② s is labelled with φ and t with $\mathsf{E}[\varphi\,\mathsf{U}\,\psi]$ for some t with $s \to t$

③ repeat ② until no change

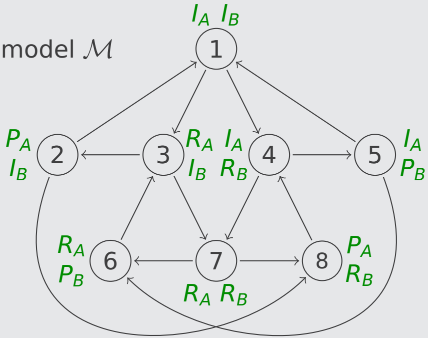
Example ①



	R_A	P_A	AF P_A	$R_A \to$ AF P_A	AG $(R_A \to$ AF $P_A)$
1				✓	$(1 \to 3)$
2		✓	✓	✓	$(2 \to 1)$
3	✓				
4				✓	$(4 \to 7)$
5				✓	$(5 \to 6)$
6	✓				
7	✓				
8		✓	✓	✓	$(8 \to 4)$

Example ❷

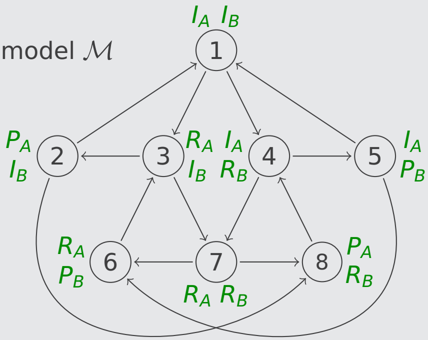
model $\mathcal{M}$



	R_A	P_A	EF P_A	$R_A \rightarrow$ EF P_A	AG ($R_A \rightarrow$ EF P_A)
1			✓	✓	✓
2		✓	✓	✓	✓
3	✓		✓	✓	✓
4			✓	✓	✓
5			✓	✓	✓
6	✓		✓	✓	✓
7	✓		✓	✓	✓
8		✓	✓	✓	✓

Example ❸

model $\mathcal{M}$



	R_B	$\neg R_B$	P_B	$\mathsf{E}[\neg R_B \mathbin{\mathsf{U}} P_B]$	$\neg \mathsf{E}[\neg R_B \mathbin{\mathsf{U}} P_B]$
1		✓			✓
2		✓			✓
3		✓			✓
4	✓				✓
5		✓	✓	✓	
6		✓	✓	✓	
7	✓				✓
8	✓				✓

More Efficient Algorithm for EG

EG φ ① restrict graph to states satisfying φ:

$$S' = \{s \in S \mid \mathcal{M}, s \vDash \varphi\}$$

$$\rightarrow' = \{(s,t) \mid s \rightarrow t \text{ and } s,t \in S'\}$$

② compute non-trivial strongly connected components of $(S', \rightarrow')$

③ label all states in such SCCs

④ compute and label all states that in $(S', \rightarrow')$ can reach labelled state

Complexity

$\mathcal{O}(f \cdot (V + E))$ with

f : # connectives

V : # states

E : # transitions

instead of $\mathcal{O}(f \cdot V \cdot (V + E))$

State Explosion Problem

size of model is more often than not exponential in number of variables and number of components which execute in parallel

- OBDDs to represent sets of states lecture 11
- abstraction
- partial order reduction
- induction
- composition

Demo

CMCV

by Matthias Perktold (2014)

Outline

Huth and Ryan

- Section 3.4.1
- Section 3.4.2
- Section 3.6.1

Post Adequacy Theorem

- Post's Functional Completeness Theorem
  Francis Jeffry Pelletier and Norman M. Martin
  Notre Dame Journal of Formal Logic 31(2), pp. 462 – 475, 1990
  doi: 10.1305/ndjfl/1093635508
- Boolean Function and Computation Models
  Peter Clote and Evangelos Kranakis
  Texts in Theoretical Computer Science, Springer, 2012
  doi: 10.1007/978-3-662-04943-3

Important Concepts

- AF
- affinity
- AG
- AU
- AX
- computation tree logic
- CTL
- EF
- EG
- EU
- EX
- model
- monotonicity
- Post’s adequacy theorem
- self-duality
- temporal connective

homework for May 23