Algorithmic Verification of Reactive Systems

Eugen Sawin

Research Group for Foundations in Artificial Intelligence
Computer Science Department
University of Freiburg

Seminar: Automata Constructions in Model Checking

Motivation

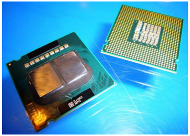
Model Checking 1/2

$$\mathcal{M} \models \varphi$$

Given a program P and specification φ:

does every run of P satisfy φ?

Motivation
Industry



Motivation

Industry

"It is dark."

"It is dark."
"It is *always* dark."

"It is dark."
"It is *always* dark."
"It is *currently* dark."

"It is dark."
"It is *always* dark."
"It is *currently* dark."
"It will *necessarily* be dark."

"It is dark."

"It is *always* dark."

"It is *currently* dark."

"It will *necessarily* be dark."

"It is dark *until* someone puts the light on."

Linear Temporal Logic

Natural language 2/2

It is dark until there is light

Linear Temporal Logic

Natural language 2/2

It is dark	until	there is light
p_0	$\mathcal{U}$	p_1

Syntax

Let $\mathcal{P}$ be the countable set of *atomic propositions*, LTL-formulae φ are defined using following productions:

$$\varphi ::= p \in \mathcal{P} \,|\, \neg\varphi \,|\, \varphi \vee \varphi \,|\, \mathcal{X}\varphi \,|\, \varphi\mathcal{U}\varphi$$

- $\neg, \vee$ corresponds to the Boolean *negation* and *disjunction*.
- $\mathcal{X}$ reads *next*.
- $\mathcal{U}$ reads *until*.

Syntax

Let $\mathcal{P}$ be the countable set of *atomic propositions*, LTL-formulae φ are defined using following productions:

$$\varphi ::= p \in \mathcal{P} \,|\, \neg\varphi \,|\, \varphi \vee \varphi \,|\, \mathcal{X}\varphi \,|\, \varphi\mathcal{U}\varphi$$

- $\neg, \vee$ corresponds to the Boolean *negation* and *disjunction*.
- $\mathcal{X}$ reads *next*.
- $\mathcal{U}$ reads *until*.

Derived connectives

Let φ and ψ be formulae:

- $\top \equiv p \vee \neg p$ for $p \in \mathcal{P}$
- $\bot \equiv \neg\top$
- $\varphi \wedge \psi \equiv \neg(\neg\varphi \vee \neg\psi)$
- $\varphi \rightarrow \psi \equiv \neg\varphi \vee \psi$
- $\varphi \leftrightarrow \psi \equiv (\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$
- $\Diamond\varphi \equiv \top\mathcal{U}\varphi$
- $\Box\varphi \equiv \neg\Diamond\neg\varphi$

Frame

An LTL-*frame* is a tuple $\mathcal{F} = (S, R)$:

- $S = \{s_i \mid i \in \mathbb{N}_0\}$ is the set of states.
- $R = \{(s_i, s_{i+1}) \mid i \in \mathbb{N}_0\}$ is the accessibility relation.

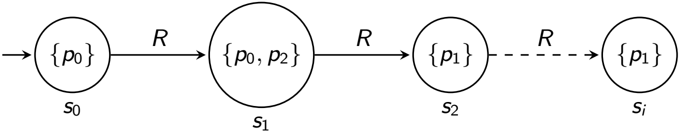
Model

An LTL-*model* is a tuple $\mathcal{M} = (\mathcal{F}, V)$:

- $\mathcal{F}$ is a *frame*.
- $V : S \to 2^{\mathcal{P}}$ is a *valuation function*.
- Intuitively we say $p \in \mathcal{P}$ is *true* at time instant i iff $p \in V(i)$.

Linear Temporal Logic

Model Example



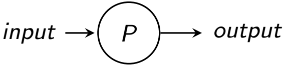
Satisfiability

A model $\mathcal{M} = (\mathcal{F}, V)$ *satisfies* a formula φ at time instant i is denoted by $\mathcal{M}, i \models \varphi$:

- $\mathcal{M}, i \models p$ for $p \in \mathcal{P} \iff p \in V(i)$
- $\mathcal{M}, i \models \neg\varphi \iff$ not $\mathcal{M}, i \models \varphi$
- $\mathcal{M}, i \models \varphi \vee \psi \iff \mathcal{M}, i \models \varphi$ or $\mathcal{M}, i \models \psi$
- $\mathcal{M}, i \models \mathcal{X}\varphi \iff \mathcal{M}, i+1 \models \varphi$
- $\mathcal{M}, i \models \varphi\mathcal{U}\psi \iff \exists k \geq i : \mathcal{M}, k \models \psi$ and $\forall i \leq j < k : \mathcal{M}, j \models \varphi$

Reactive Systems

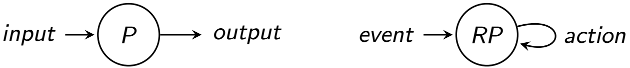
Infinite inputs



(a) Terminating program

Reactive Systems

Infinite inputs

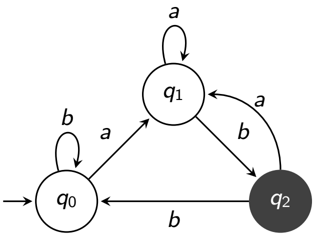


(a) Terminating program

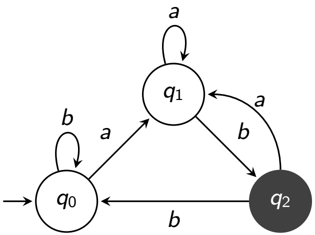
(b) Reactive program

Automata

Example 1/2



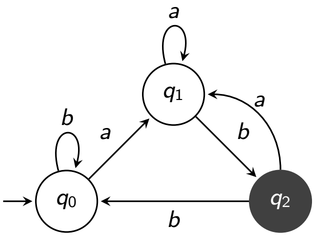
Automata
Example 1/2



$$w_1 = \overline{bbaa} \implies \rho_1 = q_0 q_0 \overline{q_0 q_1 q_1 q_2}$$

Automata

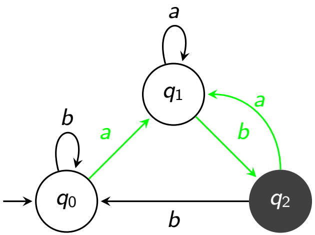
Example 1/2



$$w_1 = \overline{bbaa} \implies \rho_1 = q_0 q_0 \overline{q_0 q_1 q_1 q_2}$$
$$w_2 = bb\overline{ab} \implies \rho_2 = q_0 q_0 \overline{q_1 q_2}$$

Automata

Example 1/2

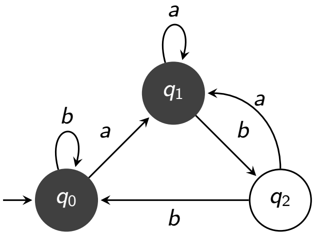


$$w_1 = \overline{bbaa} \implies \rho_1 = q_0 q_0 \overline{q_0 q_1 q_1 q_2}$$
$$w_2 = bb\overline{ab} \implies \rho_2 = q_0 q_0 \overline{q_1 q_2}$$

Accepts all inputs with infinite occurrences of *ab*.

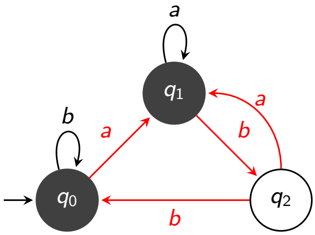
Automata

Example 2/2 (Complement)



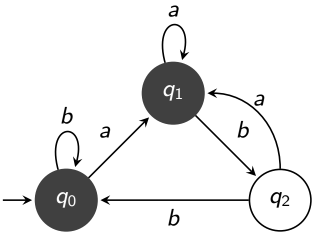
Automata

Example 2/2 (Complement)

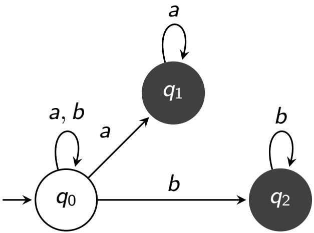


Automata

Example 2/2 (Complement)



(a) Complement automaton ✗



(b) Complement automaton ✓

Accepts all inputs with finite many *ab*.

Automaton

A non-deterministic Büchi automaton is a tuple $\mathcal{A} = (\Sigma, S, S_0, \Delta, F)$ with:

- Σ is a finite *alphabet*.
- S is a finite set of *states*.
- $S_0 \subseteq S$ is the set of *initial states*.
- $\Delta : S \times \Sigma \times S$ is a *transition relation*.
- $F \subseteq S$ is the set of *accepting states*.

Run

Let $\mathcal{A} = (\Sigma, S, S_0, \Delta, F)$ be an automaton:

- A run ρ of $\mathcal{A}$ on an infinite word $w = a_0, a_1, \ldots$ is a sequence $\rho = s_0, s_1, \ldots$:
  - $s_0 \in S_0$.
  - $(s_i, a_i, s_{i+1}) \in \Delta$, for all $i \geq 0$.
- Alternative view of a run ρ as a function $\rho : \mathbb{N}_0 \rightarrow S$, with $\rho(i) = s_i$.

Run

Let $\mathcal{A} = (\Sigma, S, S_0, \Delta, F)$ be an automaton:

- A run ρ of $\mathcal{A}$ on an infinite word $w = a_0, a_1, \ldots$ is a sequence $\rho = s_0, s_1, \ldots$:
  - $s_0 \in S_0$.
  - $(s_i, a_i, s_{i+1}) \in \Delta$, for all $i \geq 0$.
- Alternative view of a run ρ as a function $\rho : \mathbb{N}_0 \to S$, with $\rho(i) = s_i$.

$$w_1 = \overline{bbaa} \implies \rho_1 = q_0 q_0 \overline{q_0 q_1 q_1 q_2}$$

$$w_2 = bb\overline{ab} \implies \rho_2 = q_0 q_0 \overline{q_1 q_2}$$

Infinite occurrences

Let $\mathcal{A} = (\Sigma, S, S_0, \Delta, F)$ be an automaton and let ρ be a run of $\mathcal{A}$:

- $\exists^\omega$ denotes the existential quantifier for *infinitely* many occurrences.
- $\mathit{inf}(\rho) = \{s \in S \mid \exists^\omega n \in \mathbb{N}_0 : \rho(n) = s\}$.

Acceptance

Let $\mathcal{A} = (\Sigma, S, S_0, \Delta, F)$ be an automaton and let ρ be a run of $\mathcal{A}$:

- ρ is *accepting* iff $\mathit{inf}(\rho) \cap F \neq \emptyset$.
- $\mathcal{A}$ *accepts* an input word w iff there exists a run ρ of $\mathcal{A}$ on w, such that ρ is accepting.

Recognised language

Let $\mathcal{A} = (\Sigma, S, S_0, \Delta, F)$ be an automaton:

- $L_\omega(\mathcal{A}) = \{w \in \Sigma^\omega \mid \mathcal{A} \text{ accepts } w\}$, we say $\mathcal{A}$ recognises language $L_\omega(\mathcal{A})$.
- Language L is *Büchi-recognisable* iff there is an automaton $\mathcal{A}$ with $L = L_\omega(\mathcal{A})$.

Generalised Automata

Generalised automaton

A *generalised Büchi automaton* is a tuple $\mathcal{A}_G = (\Sigma, S, S_0, \Delta, \{F_i\}_{i<k})$:

- $\{F_i\}$ is a finite set of sets for a given k.
- Each $F_i \subseteq S$ is a finite set of accepting states.

Acceptance

Let $\mathcal{A}_G = (\Sigma, S, S_0, \Delta, \{F_i\}_{i<k})$ be a generalised automaton and let ρ be a run of $\mathcal{A}_G$:

- ρ is *accepting* iff $\forall i < k : \mathit{inf}(\rho) \cap F_i \neq \emptyset$.
- $\mathcal{A}_G$ *accepts* an input word w iff there exists a run ρ of $\mathcal{A}_G$ on w, such that ρ is accepting.

Proposition

Let $\mathcal{A}_G = (\Sigma, S, S_0, \Delta, \{F_i\}_{i<k})$ be a generalised automaton and let $\mathcal{A}_i = (\Sigma, S, S_0, \Delta, F_i)$ be non-deterministic automata:

$$L_\omega(\mathcal{A}_G) = \bigcap_{i<k} L_\omega(\mathcal{A}_i)$$

Model $\mathcal{M}_\varphi$ for formula φ

$\Downarrow$

Closure $CL(\varphi)$ of φ

$\Downarrow$

Automaton $\mathcal{A}_\varphi$ for φ

Model $\mathcal{M}_\varphi$ for formula φ

$\Downarrow$

Closure $CL(\varphi)$ of φ

$\Downarrow$

Automaton $\mathcal{A}_\varphi$ for φ

On-the-fly method à la Gerth et al.

Automata Construction

System automata 1/2

Program

Given a program $P = (S_P, s_0, R, V)$:

- S is the set of possible states.
- s_0 is the initial state.
- $R : S \times \mathcal{P} \times S$ is the transition relation.
- $V : S \to 2^{\mathcal{P}}$ is a valuation function.

A *computation* of P is a run $\rho = (V(s_0), V(s_1), ...)$.

System automaton

We construct automaton $\mathcal{A}_P = (\Sigma, S, S_0, \Delta, F)$ for program P:

- $\Sigma = 2^{\mathcal{P}}$
- $S = S_P$
- $\Delta = \{(s, V(s), s') \mid \exists a \in \mathcal{P} : (s, a, s') \in R\}$
- $S_0 = \{s_0\}$
- $F = S$

Proposition

Let $\mathcal{A}_P = (\Sigma, S, S_0, \Delta, F)$, note that $F = S$, it follows:

$$L_\omega(\mathcal{A}_P) = \{\rho \mid \rho \text{ is a run of } \mathcal{A}_P\}$$

Verification

Given a program P and specification φ:

does every run of P satisfy φ?

Verification

Given a program P and specification φ:

$$L_\omega(\mathcal{A}_P) \subseteq L_\omega(\mathcal{A}_\varphi)$$

Verification

Given a program P and specification φ:

$$L_\omega(\mathcal{A}_P) \cap L_\omega(\mathcal{A}_{\neg\varphi}) = \emptyset$$

Literature I

Madhavan Mukund.
Linear-Time Temporal Logic and Büchi Automata.
Winter School on Logic and Computer Science, Indian Statistical Institute, Calcutta, 1997.

Moshe Y. Vardi.
Alternating Automata and Program Verification.
Computer Science Today, Volume 1000 of Lecture Notes in Computer Science, Springer-Verlag, Berlin, 1995.

Rob Gerth, Doron Peled, Moshe Y. Vardi and Pierre Wolper.
Simple On-the-fly Automatic Verification of Linear Temporal Logic.
Proceeding IFIO/WG6.1 Symposium on Protocol Specification, Testing and Verification, Warsaw, 1995.

Patrick Blackburn, Frank Wolter and Johan van Benthem.
Handbook of Modal Logic.
3rd Edition, Elsevier, Amsterdam, Chapter 11 P. 655-720 and Chapter 17 P. 975-989, 2007.

Literature II

Moshe Y. Vardi.
Automated Verification: Graphs, Logic and Automata.
Proceeding of the International Joint Conference on Artificial Intelligence, Acapulco, 2003.