Advanced Logic

http://www-sop.inria.fr/members/Martin.Avanzini/teaching/2021/AL/

Martin Avanzini

Summer Semester 2021

Last Lecture

- a language $L \subseteq \Sigma^\omega$ is ω-regular if $L = \bigcup_{0 \le i \le n} U_i \cdot V_i^\omega$ for regular languages U_i, V_i $(0 \le i \le n)$
- a Büchi Automaton is structurally similar to an NFA, but recognizes words $w \in \Sigma^\infty$ that visit final states infinitely often

Theorem

For recognisable $U \in \Sigma^*$ and $V, W \in \Sigma^\omega$ the following are recognisable:

1. *union* $V \cup W$
2. *intersection* $V \cap W$
3. *left-concatenation* $U \cdot V$
4. ω*-iteration* U^ω
5. *complement* $\overline{V}$

Theorem

$L \in \omega REG(\Sigma)$ *if and only if* $L = \mathrm{L}(\mathcal{A})$ *for some NBA* $\mathcal{A}$

Theorem

For every MSO formula ϕ *there exists an NBA* $\mathcal{A}_\phi$ *s.t.* $\hat{\mathrm{L}}(\phi) = \mathrm{L}(\mathcal{A}_\phi)$.

Today's Lecture

1. Linear temporal logic (LTL)
2. LTL model checking

Linear temporal logic

Motivation

- linear temporal logic is a logic for reasoning about events in time
  - always not $(\phi \wedge \psi)$ — safety
  - always (Request implies eventually Grant) — liveness
  - always (Request implies (Request until Grant)) — liveness
- LTL shares algorithmic solutions with MSO

Formal Definition

- the set of LTL formulas over propositions $\mathcal{P} = \{p, q, \ldots\}$ is given by

$$\phi, \psi ::= p \;\big|\; \phi \vee \psi \;\big|\; \neg\phi \qquad \textit{(Propositional Formulas)}$$

$$\big|\; \mathsf{X}\,\phi \;\big|\; \phi\,\mathsf{U}\,\psi \qquad \textit{(Next and Until)}$$

Formal Definition

- the set of LTL formulas over propositions $\mathcal{P} = \{p, q, \ldots\}$ is given by

$$\phi, \psi ::= p \;\big|\; \phi \vee \psi \;\big|\; \neg\phi \qquad \textit{(Propositional Formulas)}$$

$$\big|\; \mathsf{X}\,\phi \;\big|\; \phi\,\mathsf{U}\,\psi \qquad \textit{(Next and Until)}$$

- LTL is a logic of temporal sequences, modeled as infinite words over $\Sigma \triangleq 2^{\mathcal{P}}$

Formal Definition

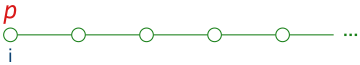
- the set of LTL formulas over propositions $\mathcal{P} = \{p, q, \dots\}$ is given by

$$\phi, \psi ::= p \;\big|\; \phi \vee \psi \;\big|\; \neg\phi \qquad\qquad \textit{(Propositional Formulas)}$$

$$\big|\; \mathsf{X}\,\phi \;\big|\; \phi \,\mathsf{U}\, \psi \qquad\qquad \textit{(Next and Until)}$$

- LTL is a logic of temporal sequences, modeled as infinite words over $\Sigma \triangleq 2^{\mathcal{P}}$
- for a sentence ϕ and $w = P_0 P_1 P_2 \dots$, we define $w \models \phi$ as $w; 0 \models \phi$ where

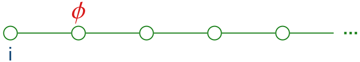
$$w; i \models p \quad :\Longleftrightarrow \quad p \in P_i$$



$$w; i \models \phi \vee \psi \quad :\Longleftrightarrow \quad w; i \models \phi \text{ or } w; i \models \psi$$

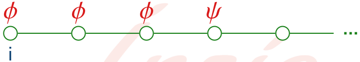
$$w; i \models \neg\phi \quad :\Longleftrightarrow \quad w; i \not\models \phi$$

$$w, i \models \mathsf{X}\,\phi \quad :\Longleftrightarrow \quad w; i+1 \models \phi$$



$$w; i \models \phi \,\mathsf{U}\, \psi \quad :\Longleftrightarrow \quad \text{exists } k \geq i \text{ s.t. } w; k \models \phi$$

$$\text{and } w; j \models \psi \text{ for all } i \leq j < k$$



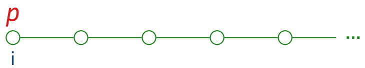
Formal Definition

- the set of LTL formulas over propositions $\mathcal{P} = \{p, q, \ldots\}$ is given by

$$\begin{aligned} \phi, \psi ::= p \mid \phi \vee \psi \mid \neg \phi & \qquad \textit{(Propositional Formulas)} \\ \mid \mathsf{X}\,\phi \mid \phi\;\mathsf{U}\;\psi & \qquad \textit{(Next and Until)} \end{aligned}$$

- LTL is a logic of temporal sequences, modeled as infinite words over $\Sigma \triangleq 2^{\mathcal{P}}$
- for a sentence ϕ and $w = P_0 P_1 P_2 \ldots$, we define $w \vDash \phi$ as $w; 0 \vDash \phi$ where

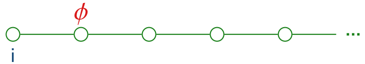
$$w; i \vDash p \quad :\Longleftrightarrow \quad p \in P_i$$



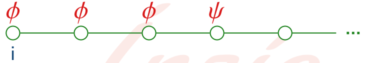
$$w; i \vDash \phi \vee \psi \quad :\Longleftrightarrow \quad w; i \vDash \phi \text{ or } w; i \vDash \psi$$

$$w; i \vDash \neg \phi \quad :\Longleftrightarrow \quad w; i \nvDash \phi$$

$$w, i \vDash \mathsf{X}\,\phi \quad :\Longleftrightarrow \quad w; i+1 \vDash \phi$$



$$w; i \vDash \phi\;\mathsf{U}\;\psi \quad :\Longleftrightarrow \quad \text{exists } k \geq i \text{ s.t. } w; k \vDash \phi \text{ and } w; j \vDash \psi \text{ for all } i \leq j < k$$



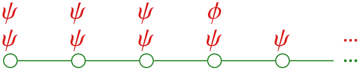
- a LTL formula ϕ defines the language $\mathsf{L}(\phi) \triangleq \{w \mid w \vDash \phi\}$

Derived Operators and Positive Normal Forms

finally: $\mathsf{F}\,\phi \quad :\Longleftrightarrow \quad \top \;\mathsf{U}\; \phi$

globally: $\mathsf{G}\,\phi \quad :\Longleftrightarrow \quad \neg(\mathsf{F}\,\neg\phi)$

release: $\phi \;\mathsf{R}\; \psi \quad :\Longleftrightarrow \quad \neg(\neg\phi \;\mathsf{U}\; \neg\psi)$

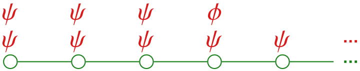


Derived Operators and Positive Normal Forms

finally: $\mathsf{F}\,\phi \;:\Longleftrightarrow\; \top\;\mathsf{U}\;\phi$

globally: $\mathsf{G}\,\phi \;:\Longleftrightarrow\; \neg(\mathsf{F}\,\neg\phi)$

release: $\phi\;\mathsf{R}\;\psi \;:\Longleftrightarrow\; \neg(\neg\phi\;\mathsf{U}\;\neg\psi)$



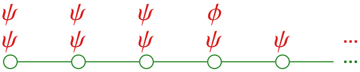
★ $\mathsf{F}\,\phi$, $\mathsf{G}\,\phi$ and $\mathsf{X}\,\phi$ are sometimes denoted by $\diamond\phi$, $\Box\phi$ and $\circ\phi$, respectively

Derived Operators and Positive Normal Forms

finally: $\mathsf{F}\,\phi \;:\Longleftrightarrow\; \top\;\mathsf{U}\;\phi$

globally: $\mathsf{G}\,\phi \;:\Longleftrightarrow\; \neg(\mathsf{F}\,\neg\phi)$

release: $\phi\;\mathsf{R}\;\psi \;:\Longleftrightarrow\; \neg(\neg\phi\;\mathsf{U}\;\neg\psi)$



- ⋆ $\mathsf{F}\,\phi$, $\mathsf{G}\,\phi$ and $\mathsf{X}\,\phi$ are sometimes denoted by $\diamond\phi$, $\square\phi$ and $\circ\phi$, respectively
- ⋆ a formula ϕ is in positive normal form (PNF) if it is derived from the following grammar:

$$\phi,\psi ::= p \;\mid\; \neg p \;\mid\; \phi\wedge\psi \;\mid\; \phi\vee\psi \;\mid\; \mathsf{X}\,\phi \;\mid\; \phi\;\mathsf{U}\;\psi \;\mid\; \phi\;\mathsf{R}\;\psi$$

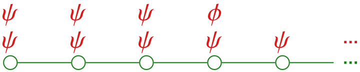
  – negation only in front of literals

Derived Operators and Positive Normal Forms

finally: $\mathsf{F}\,\phi \quad :\Longleftrightarrow \quad \top \,\mathsf{U}\, \phi$

globally: $\mathsf{G}\,\phi \quad :\Longleftrightarrow \quad \neg(\mathsf{F}\,\neg\phi)$

release: $\phi\,\mathsf{R}\,\psi \quad :\Longleftrightarrow \quad \neg(\neg\phi\,\mathsf{U}\,\neg\psi)$



- ★ $\mathsf{F}\,\phi$, $\mathsf{G}\,\phi$ and $\mathsf{X}\,\phi$ are sometimes denoted by $\diamond\phi$, $\Box\phi$ and $\circ\phi$, respectively
- ★ a formula ϕ is in positive normal form (PNF) if it is derived from the following grammar:

$$\phi, \psi ::= p \;\mid\; \neg p \;\mid\; \phi \wedge \psi \;\mid\; \phi \vee \psi \;\mid\; \mathsf{X}\,\phi \;\mid\; \phi\,\mathsf{U}\,\psi \;\mid\; \phi\,\mathsf{R}\,\psi$$

  – negation only in front of literals

Lemma

Every formula ϕ can be turned into an equivalent formula ψ in PNF with $|\psi| \leq 2|\phi|$

Safety Properties in LTL

Safety = something bad never happens = $\mathsf{G}\,\neg\phi_{\mathsf{bad}}$

Safety Properties in LTL

Safety = something bad never happens = $G \neg \phi_{bad}$

Example

- a ...A train is approaching
- c ...A train is crossing
- l ...The light is blinking
- b ...The barrier is down

Safety Properties in LTL

Safety = something bad never happens = $G \neg \phi_{bad}$

Example

- a ...A train is approaching
- c ...A train is crossing
- l ...The light is blinking
- b ...The barrier is down

- when a train is crossing, the barrier is down:

Safety Properties in LTL

Safety = something bad never happens = $\mathsf{G}\,\neg\phi_{\mathsf{bad}}$

Example

- a ...A train is approaching
- c ...A train is crossing
- l ...The light is blinking
- b ...The barrier is down

- when a train is crossing, the barrier is down:

$$\mathsf{G}\,(c \rightarrow b) \equiv \mathsf{G}\,\neg(c \wedge \neg b)$$

Safety Properties in LTL

Safety = something bad never happens = $\mathsf{G}\,\neg\phi_{\mathsf{bad}}$

Example

- ★ a ...A train is approaching
- ★ c ...A train is crossing
- ★ l ...The light is blinking
- ★ b ...The barrier is down

- ★ when a train is crossing, the barrier is down:

$$\mathsf{G}\,(c \rightarrow b) \equiv \mathsf{G}\,\neg(c \wedge \neg b)$$

- ★ if a train is approaching or crossing, the light must be blinking:

Safety Properties in LTL

Safety = something bad never happens = $\mathsf{G}\,\neg\phi_{\mathrm{bad}}$

Example

- ⋆ a ...A train is approaching
- ⋆ c ...A train is crossing
- ⋆ l ...The light is blinking
- ⋆ b ...The barrier is down

- ⋆ when a train is crossing, the barrier is down:

$$\mathsf{G}\,(c \rightarrow b) \equiv \mathsf{G}\,\neg(c \wedge \neg b)$$

- ⋆ if a train is approaching or crossing, the light must be blinking:

$$\mathsf{G}\,(a \vee c \rightarrow b) \equiv \mathsf{G}\,\neg((a \vee c) \wedge \neg l)$$

Safety Properties in LTL

Safety = something bad never happens = $\mathsf{G}\,\neg\phi_{\text{bad}}$

Example

- ★ a ...A train is approaching
- ★ c ...A train is crossing
- ★ l ...The light is blinking
- ★ b ...The barrier is down

- ★ when a train is crossing, the barrier is down:

$$\mathsf{G}\,(c \rightarrow b) \equiv \mathsf{G}\,\neg(c \wedge \neg b)$$

- ★ if a train is approaching or crossing, the light must be blinking:

$$\mathsf{G}\,(a \vee c \rightarrow b) \equiv \mathsf{G}\,\neg((a \vee c) \wedge \neg l)$$

- ★ if the barrier is up and the light is off, no train is approaching or crossing:

Safety Properties in LTL

Safety = something bad never happens = $G \neg \phi_{bad}$

Example

- a ...A train is approaching
- c ...A train is crossing
- l ...The light is blinking
- b ...The barrier is down

- when a train is crossing, the barrier is down:

$$G\,(c \rightarrow b) \equiv G \neg (c \wedge \neg b)$$

- if a train is approaching or crossing, the light must be blinking:

$$G\,(a \vee c \rightarrow b) \equiv G \neg ((a \vee c) \wedge \neg l)$$

- if the barrier is up and the light is off, no train is approaching or crossing:

$$G\,(\neg b \wedge \neg l \rightarrow \neg a \wedge \neg c) \equiv G \neg (\neg b \wedge \neg l \wedge (a \vee c))$$

Liveness Properties in LTL

Liveness = something intiated eventually terminates = $\mathsf{G}\,(\phi_{\mathsf{init}} \rightarrow \mathsf{F}\,\phi_{\mathsf{term}})$

Liveness Properties in LTL

Liveness = something intiated eventually terminates = $\mathsf{G}\,(\phi_{\text{init}} \rightarrow \mathsf{F}\,\phi_{\text{term}})$

- approaching trains eventually cross:

Liveness Properties in LTL

Liveness = something intiated eventually terminates = $\mathsf{G}\,(\phi_{\mathsf{init}} \rightarrow \mathsf{F}\,\phi_{\mathsf{term}})$

- approaching trains eventually cross:

$$\mathsf{G}\,(\mathsf{a} \rightarrow \mathsf{F}\,\mathsf{c})$$

Liveness Properties in LTL

Liveness = something intiated eventually terminates = $\mathsf{G}\,(\phi_{\mathrm{init}} \rightarrow \mathsf{F}\,\phi_{\mathrm{term}})$

- approaching trains eventually cross:

$$\mathsf{G}\,(a \rightarrow \mathsf{F}\,c)$$

- when a train is approaching, the barrier is down before it crosses:

Liveness Properties in LTL

Liveness = something intiated eventually terminates = $\mathsf{G}\,(\phi_{\mathsf{init}} \rightarrow \mathsf{F}\,\phi_{\mathsf{term}})$

- ⋆ approaching trains eventually cross:

$$\mathsf{G}\,(a \rightarrow \mathsf{F}\,c)$$

- ⋆ when a train is approaching, the barrier is down before it crosses:

$$\mathsf{G}\,(a \rightarrow \neg c\ \mathsf{U}\ b)$$

Liveness Properties in LTL

Liveness = something intiated eventually terminates = $G\,(\phi_{\text{init}} \to F\,\phi_{\text{term}})$

- approaching trains eventually cross:

$$G\,(a \to F\,c)$$

- when a train is approaching, the barrier is down before it crosses:

$$G\,(a \to \neg c\; U\; b)$$

- if a train finished crossing, the barrier will be eventually risen

Liveness Properties in LTL

Liveness = something intiated eventually terminates = $\mathsf{G}\,(\phi_{init} \rightarrow \mathsf{F}\,\phi_{term})$

- approaching trains eventually cross:

$$\mathsf{G}\,(a \rightarrow \mathsf{F}\,c)$$

- when a train is approaching, the barrier is down before it crosses:

$$\mathsf{G}\,(a \rightarrow \neg c\;\mathsf{U}\;b)$$

- if a train finished crossing, the barrier will be eventually risen

$$\mathsf{G}\,(c \wedge \mathsf{X}\,\neg c \rightarrow \mathsf{X}\,\mathsf{F}\,\neg b)$$

Characterising LTL

- ⋆ LTL can be "expressed" within MSO $\equiv$ Büchi Automata
- ⋆ MSO and Büchi Automata are strictly more expressive

LTL recognisability $<$ ω-regular

- ⋆ LTL most naturally translated to alternating Büchi Automata (ABA)
- ⋆ loop-free (very weak) ABA characterise precisely the class of LTL recognisable languages

Characterising LTL

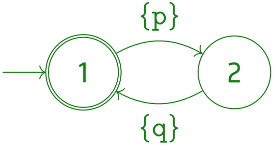
- ⋆ LTL can be "expressed" within MSO ≡ Büchi Automata
- ⋆ MSO and Büchi Automata are strictly more expressive

$$\text{LTL recognisability} < \omega\text{-regular}$$

- ⋆ LTL most naturally translated to alternating Büchi Automata (ABA)
- ⋆ loop-free (very weak) ABA characterise precisely the class of LTL recognisable languages

Example

the Büchi Automaton $\mathcal{A}$ over $\mathcal{P} = \{p, q\}$ given by



is not loop-free $\Rightarrow$ $L(\mathcal{A})$ not expressible in LTL

(Very Weak) Alternating Büchi Automata

- ⋆ an alternating Büchi Automaton (ABA) is a tuple $\mathcal{A} = (Q, \Sigma, q_I, \delta, F)$ identical to an AFA
- ⋆ execution on words $w \in \Sigma^\omega$ are now infinite tree T_w
- ⋆ an execution is accepting in the sense of Büchi: every path visits F infinitely often
- ⋆ $L(\mathcal{A}) \triangleq \{w \in \Sigma^\omega \mid \text{there exist an accepting execution } T_w \text{ for } w\}$

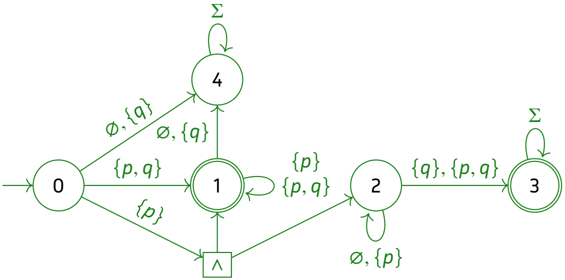
(Very Weak) Alternating Büchi Automata

- ⋆ an alternating Büchi Automaton (ABA) is a tuple $\mathcal{A} = (Q, \Sigma, q_I, \delta, F)$ identical to an AFA
- ⋆ execution on words $w \in \Sigma^\omega$ are now infinite tree T_w
- ⋆ an execution is accepting in the sense of Büchi: every path visits F infinitely often
- ⋆ $\mathrm{L}(\mathcal{A}) \triangleq \{w \in \Sigma^\omega \mid \text{there exist an accepting execution } T_w \text{ for } w\}$
- ⋆ very weak ABA (VWABA) is an ABA if for every $\mathrm{a} \in \Sigma$, $\xrightarrow{\mathrm{a}} \subseteq \leq$ for some linear order $\leq \subseteq Q \times Q$

(Very Weak) Alternating Büchi Automata

- ⋆ an alternating Büchi Automaton (ABA) is a tuple $\mathcal{A} = (Q, \Sigma, q_I, \delta, F)$ identical to an AFA
- ⋆ execution on words $w \in \Sigma^\omega$ are now infinite tree T_w
- ⋆ an execution is accepting in the sense of Büchi: every path visits F infinitely often
- ⋆ $L(\mathcal{A}) \triangleq \{w \in \Sigma^\omega \mid \text{there exist an accepting execution } T_w \text{ for } w\}$
- ⋆ very weak ABA (VWABA) is an ABA if for every $a \in \Sigma$, $\xrightarrow{a} \subseteq \leq$ for some linear order $\leq \subseteq Q \times Q$

Example



$\mathrm{G}\, p \wedge \mathrm{F}\, q$

LTL and Automata

Theorem

Let L be a language over $\Sigma = 2^{\mathcal{P}}$. The following are equivalent:

- ⋆ *L is LTL definable.*
- ⋆ *L is recognizable by VWABA.*

From Automata to LTL

fix a VWABA $\mathcal{A} = (\{q_0, \ldots, q_n\}, 2^{\mathcal{P}}, q_0, \delta, F)$ where wlog. $q_0 > q_1 > \cdots > q_n$

From Automata to LTL

fix a VWABA $\mathcal{A} = (\{q_0, \ldots, q_n\}, 2^{\mathcal{P}}, q_0, \delta, F)$ where wlog. $q_0 > q_1 > \cdots > q_n$

- ⋆ since $\mathcal{A}$ is very weak, there are transitions from q_i to q_j only if $i \geq j$

From Automata to LTL

fix a VWABA $\mathcal{A} = (\{q_0, \ldots, q_n\}, 2^{\mathcal{P}}, q_0, \delta, F)$ where wlog. $q_0 > q_1 > \cdots > q_n$

- since $\mathcal{A}$ is very weak, there are transitions from q_i to q_j only if $i \geq j$
- we now associate each state q_i with a formula ϕ_i s.t.

$$\mathsf{L}(\phi_i) = \mathsf{L}_{\mathcal{A}}(q_i)$$

From Automata to LTL

fix a VWABA $\mathcal{A} = (\{q_0, \ldots, q_n\}, 2^{\mathcal{P}}, q_0, \delta, F)$ where wlog. $q_0 > q_1 > \cdots > q_n$

- since $\mathcal{A}$ is very weak, there are transitions from q_i to q_j only if $i \geq j$
- we now associate each state q_i with a formula ϕ_i s.t.

$$\mathsf{L}(\phi_i) = \mathsf{L}_{\mathcal{A}}(q_i)$$

- this can be done inductively: while construction ϕ_i, we already have suitable formulas ϕ_j for $i > j$

From Automata to LTL

fix a VWABA $\mathcal{A} = (\{q_0, \ldots, q_n\}, 2^{\mathcal{P}}, q_0, \delta, F)$ where wlog. $q_0 > q_1 > \cdots > q_n$

- since $\mathcal{A}$ is very weak, there are transitions from q_i to q_j only if $i \geq j$
- we now associate each state q_i with a formula ϕ_i s.t.

$$\mathrm{L}(\phi_i) = \mathrm{L}_{\mathcal{A}}(q_i)$$

- this can be done inductively: while construction ϕ_i, we already have suitable formulas ϕ_j for $i > j$
- for propositions $P \subseteq \mathcal{P}$, the construction uses the characteristic function

$$\chi_P \triangleq \left(\bigwedge_{p \in P} p\right) \wedge \left(\bigwedge_{p \notin P} \neg p\right)$$

From Automata to LTL

fix a VWABA $\mathcal{A} = (\{q_0, \ldots, q_n\}, 2^{\mathcal{P}}, q_0, \delta, F)$ where wlog. $q_0 > q_1 > \cdots > q_n$

- since $\mathcal{A}$ is very weak, there are transitions from q_i to q_j only if $i \geq j$
- we now associate each state q_i with a formula ϕ_i s.t.

$$\mathsf{L}(\phi_i) = \mathsf{L}_{\mathcal{A}}(q_i)$$

- this can be done inductively: while construction ϕ_i, we already have suitable formulas ϕ_j for $i > j$
- for propositions $P \subseteq \mathcal{P}$, the construction uses the characteristic function

$$\chi_P \triangleq \left(\bigwedge_{p \in P} p\right) \wedge \left(\bigwedge_{p \notin P} \neg p\right)$$

- the construction differs whether the state is final, we thus consider two cases

From Automata to LTL (II)

fix a VWABA $\mathcal{A} = (\{q_0, \ldots, q_n\}, 2^{\mathcal{P}}, q_0, \delta, F)$ where wlog. $q_0 > q_1 > \cdots > q_n$

- note that $\mathrm{L}_{\mathcal{A}}(q_i)$ satisfies

$$\mathrm{L}_{\mathcal{A}}(q_i) \equiv \bigvee_{P \subseteq \mathcal{P}} \chi_P \wedge \mathrm{X}\,(\delta(q_i, P)[q_i/\mathrm{L}_{\mathcal{A}}(q_i), q_{i+1}/\mathrm{L}_{\mathcal{A}}(q_{i+1}) \ldots, q_n/\mathrm{L}_{\mathcal{A}}(q_i)])$$

- if $q_i \notin F$ then we rewrite $\mathrm{L}_{\mathcal{A}}(q_i)$ as $\psi \vee (\rho \wedge \mathrm{X}\,\mathrm{L}_{\mathcal{A}}(q_i))$ and set

$$\phi_i \triangleq \rho \;\mathrm{U}\; \psi$$

- if $q_i \in F$ then we rewrite $\mathrm{L}_{\mathcal{A}}(q_i)$ as $\psi \wedge (\rho \vee \mathrm{X}\,\mathrm{L}_{\mathcal{A}}(q_i))$ and set

$$\phi_i \triangleq \mathrm{G}\,\psi \vee (\psi \;\mathrm{U}\; (\rho \wedge \psi))$$

From LTL to Automata

the ABA $\mathcal{A}_\phi$ for a PNF formula ϕ is given by $(Q, 2^{\mathcal{P}}, \phi, \delta, F)$ where

- ⋆ $Q \triangleq \{\top, \bot\} \cup \{q_\psi \mid \psi$ occurs as sub-formula in $\phi\}$

From LTL to Automata

the ABA $\mathcal{A}_\phi$ for a PNF formula ϕ is given by $(Q, 2^{\mathcal{P}}, \phi, \delta, F)$ where

- ⋆ $Q \triangleq \{\top, \bot\} \cup \{q_\psi \mid \psi \text{ occurs as sub-formula in } \phi\}$
- ⋆ the transition function $\delta : Q \times 2^{\mathcal{P}} \to \mathbb{B}^+(Q)$ is given by

$$\delta(\top, P) \triangleq \top \quad \delta(\bot, P) \triangleq \bot \quad \delta(q_p, P) \triangleq \begin{cases} \top & \text{if } p \in P \\ \bot & \text{if } p \notin P \end{cases} \quad \delta(q_{\neg p}, P) \triangleq \begin{cases} \bot & \text{if } p \in P \\ \top & \text{if } p \notin P \end{cases}$$

$$\delta(q_{\psi_1 \wedge \psi_2}, P) \triangleq \delta(q_{\psi_1}, P) \wedge \delta(q_{\psi_2}, P) \qquad \delta(q_{\psi_1 \vee \psi_2}, P) \triangleq \delta(q_{\psi_1}, P) \vee \delta(q_{\psi_2}, P)$$

$$\delta(q_{\mathsf{X}\,\psi}, P) \triangleq q_\psi$$

$$\delta(q_{\psi_1 \mathsf{U} \psi_2}, P) \triangleq \delta(q_{\psi_2}, P) \vee (\delta(q_{\psi_1}, P) \wedge q_{\psi_1 \mathsf{U} \psi_2})$$

$$\delta(q_{\psi_1 \mathsf{R} \psi_2}, P) \triangleq \delta(q_{\psi_2}, P) \wedge (\delta(q_{\psi_1}, P) \vee q_{\psi_1 \mathsf{R} \psi_2})$$

From LTL to Automata

the ABA $\mathcal{A}_\phi$ for a PNF formula ϕ is given by $(Q, 2^{\mathcal{P}}, \phi, \delta, F)$ where

- ⋆ $Q \triangleq \{\top, \bot\} \cup \{q_\psi \mid \psi \text{ occurs as sub-formula in } \phi\}$
- ⋆ the transition function $\delta : Q \times 2^{\mathcal{P}} \to \mathbb{B}^+(Q)$ is given by

$$\delta(\top, P) \triangleq \top \quad \delta(\bot, P) \triangleq \bot \quad \delta(q_p, P) \triangleq \begin{cases} \top & \text{if } p \in P \\ \bot & \text{if } p \notin P \end{cases} \quad \delta(q_{\neg p}, P) \triangleq \begin{cases} \bot & \text{if } p \in P \\ \top & \text{if } p \notin P \end{cases}$$

$$\delta(q_{\psi_1 \wedge \psi_2}, P) \triangleq \delta(q_{\psi_1}, P) \wedge \delta(q_{\psi_2}, P) \qquad \delta(q_{\psi_1 \vee \psi_2}, P) \triangleq \delta(q_{\psi_1}, P) \vee \delta(q_{\psi_2}, P)$$

$$\delta(q_{\mathsf{X}\,\psi}, P) \triangleq q_\psi$$

$$\delta(q_{\psi_1 \mathsf{U} \psi_2}, P) \triangleq \delta(q_{\psi_2}, P) \vee (\delta(q_{\psi_1}, P) \wedge q_{\psi_1 \mathsf{U} \psi_2})$$

$$\delta(q_{\psi_1 \mathsf{R} \psi_2}, P) \triangleq \delta(q_{\psi_2}, P) \wedge (\delta(q_{\psi_1}, P) \vee q_{\psi_1 \mathsf{R} \psi_2})$$

- ⋆ the only final states are $\top$ and $q_{\psi_1 \mathsf{R} \psi_2} \in Q$

From LTL to Automata

the ABA $\mathcal{A}_\phi$ for a PNF formula ϕ is given by $(Q, 2^{\mathcal{P}}, \phi, \delta, F)$ where

- ⋆ $Q \triangleq \{\top, \bot\} \cup \{q_\psi \mid \psi \text{ occurs as sub-formula in } \phi\}$
- ⋆ the transition function $\delta : Q \times 2^{\mathcal{P}} \to \mathbb{B}^+(Q)$ is given by

$$\delta(\top, P) \triangleq \top \quad \delta(\bot, P) \triangleq \bot \quad \delta(q_p, P) \triangleq \begin{cases} \top & \text{if } p \in P \\ \bot & \text{if } p \notin P \end{cases} \quad \delta(q_{\neg p}, P) \triangleq \begin{cases} \bot & \text{if } p \in P \\ \top & \text{if } p \notin P \end{cases}$$

$$\delta(q_{\psi_1 \wedge \psi_2}, P) \triangleq \delta(q_{\psi_1}, P) \wedge \delta(q_{\psi_2}, P) \qquad \delta(q_{\psi_1 \vee \psi_2}, P) \triangleq \delta(q_{\psi_1}, P) \vee \delta(q_{\psi_2}, P)$$

$$\delta(q_{\mathsf{X}\,\psi}, P) \triangleq q_\psi$$

$$\delta(q_{\psi_1 \mathsf{U} \psi_2}, P) \triangleq \delta(q_{\psi_2}, P) \vee (\delta(q_{\psi_1}, P) \wedge q_{\psi_1 \mathsf{U} \psi_2})$$

$$\delta(q_{\psi_1 \mathsf{R} \psi_2}, P) \triangleq \delta(q_{\psi_2}, P) \wedge (\delta(q_{\psi_1}, P) \vee q_{\psi_1 \mathsf{R} \psi_2})$$

- ⋆ the only final states are $\top$ and $q_{\psi_1 \mathsf{R} \psi_2} \in Q$

Notes

- ⋆ $\mathcal{A}_\phi$ is linear in size in $|\phi|$
- ⋆ using the construction for AFAs, this ABA can be transformed to an NBA of size $O(2^{|\phi|})$

Example

consider $\phi = \mathsf{G}\, p \wedge \mathsf{F}\, q \equiv ((p \wedge \neg p)\ \mathsf{R}\ p) \wedge ((p \vee \neg p)\ \mathsf{U}\ q)$

Example

consider $\phi = \mathsf{G}\, p \wedge \mathsf{F}\, q \equiv ((p \wedge \neg p)\ \mathsf{R}\ p) \wedge ((p \vee \neg p)\ \mathsf{U}\ q)$

$$\delta(q_p, P) = \begin{cases} \top & \text{if } p \in P \\ \bot & \text{if } p \notin P \end{cases}$$

$$\delta(q_{\neg p}, P) = \begin{cases} \bot & \text{if } p \in P \\ \top & \text{if } p \notin P \end{cases}$$

Example

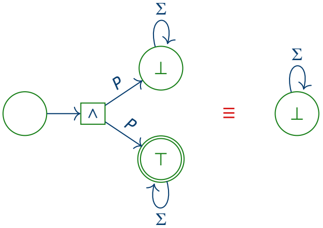
consider $\phi = \mathsf{G}\, p \wedge \mathsf{F}\, q \equiv ((p \wedge \neg p)\ \mathsf{R}\ p) \wedge ((p \vee \neg p)\ \mathsf{U}\ q)$

$$\delta(q_p, P) = \begin{cases} \top & \text{if } p \in P \\ \bot & \text{if } p \notin P \end{cases}$$

$$\delta(q_{\neg p}, P) = \begin{cases} \bot & \text{if } p \in P \\ \top & \text{if } p \notin P \end{cases}$$

$$\delta(q_{p \wedge \neg p}, P) = \delta(q_p, P) \wedge \delta(q_{\neg p}, P) = \top \wedge \bot \approx \bot$$

$$\delta(q_{p \vee \neg p}, P) = \delta(q_p, P) \vee \delta(q_{\neg p}, P) = \bot \vee \top \approx \top$$



Example

consider $\phi = \mathsf{G}\, p \wedge \mathsf{F}\, q \equiv ((p \wedge \neg p)\ \mathsf{R}\ p) \wedge ((p \vee \neg p)\ \mathsf{U}\ q)$

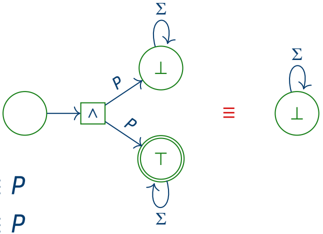
$$\delta(q_p, P) = \begin{cases} \top & \text{if } p \in P \\ \bot & \text{if } p \notin P \end{cases}$$

$$\delta(q_{\neg p}, P) = \begin{cases} \bot & \text{if } p \in P \\ \top & \text{if } p \notin P \end{cases}$$

$$\delta(q_{p \wedge \neg p}, P) = \delta(q_p, P) \wedge \delta(q_{\neg p}, P) = \top \wedge \bot \approx \bot$$

$$\delta(q_{p \vee \neg p}, P) = \delta(q_p, P) \vee \delta(q_{\neg p}, P) = \bot \vee \top \approx \top$$

$$\delta(q_{(p \wedge \neg p)\mathsf{R}p}, P) = \delta(p, P) \wedge (\delta(q_{p \wedge \neg p}, P) \vee q_{(p \wedge \neg p)\mathsf{R}p}) \approx \begin{cases} q_{(p \wedge \neg p)\mathsf{R}p} & \text{if } p \in P \\ \bot & \text{if } p \notin P \end{cases}$$



Example

consider $\phi = \mathsf{G}\, p \wedge \mathsf{F}\, q \equiv ((p \wedge \neg p)\ \mathsf{R}\ p) \wedge ((p \vee \neg p)\ \mathsf{U}\ q)$

$$\delta(q_p, P) = \begin{cases} \top & \text{if } p \in P \\ \bot & \text{if } p \notin P \end{cases}$$

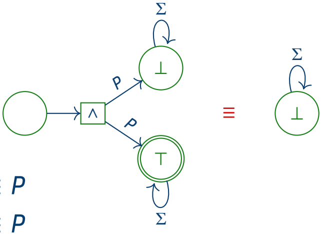
$$\delta(q_{\neg p}, P) = \begin{cases} \bot & \text{if } p \in P \\ \top & \text{if } p \notin P \end{cases}$$

$$\delta(q_{p \wedge \neg p}, P) = \delta(q_p, P) \wedge \delta(q_{\neg p}, P) = \top \wedge \bot \approx \bot$$

$$\delta(q_{p \vee \neg p}, P) = \delta(q_p, P) \vee \delta(q_{\neg p}, P) = \bot \vee \top \approx \top$$

$$\delta(q_{(p \wedge \neg p)\mathsf{R}p}, P) = \delta(p, P) \wedge (\delta(q_{p \wedge \neg p}, P) \vee q_{(p \wedge \neg p)\mathsf{R}p}) \approx \begin{cases} q_{(p \wedge \neg p)\mathsf{R}p} & \text{if } p \in P \\ \bot & \text{if } p \notin P \end{cases}$$

$$\delta(q_{(p \vee \neg p)\mathsf{U}q}, P) = \delta(q, P) \vee (\delta(q_{p \vee \neg p}, P) \wedge q_{(p \vee \neg p)\mathsf{R}q}) \approx \begin{cases} \top & \text{if } q \in P \\ q_{(p \vee \neg p)\mathsf{U}q} & \text{if } q \notin P \end{cases}$$



Example

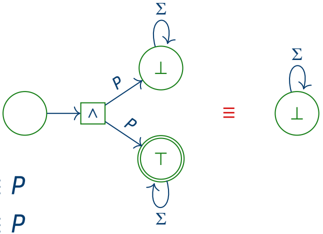
consider $\phi = \mathsf{G}\, p \wedge \mathsf{F}\, q \equiv ((p \wedge \neg p)\ \mathsf{R}\ p) \wedge ((p \vee \neg p)\ \mathsf{U}\ q)$

$$\delta(q_p, P) = \begin{cases} \top & \text{if } p \in P \\ \bot & \text{if } p \notin P \end{cases}$$

$$\delta(q_{\neg p}, P) = \begin{cases} \bot & \text{if } p \in P \\ \top & \text{if } p \notin P \end{cases}$$

$$\delta(q_{p \wedge \neg p}, P) = \delta(q_p, P) \wedge \delta(q_{\neg p}, P) = \top \wedge \bot \approx \bot$$

$$\delta(q_{p \vee \neg p}, P) = \delta(q_p, P) \vee \delta(q_{\neg p}, P) = \bot \vee \top \approx \top$$



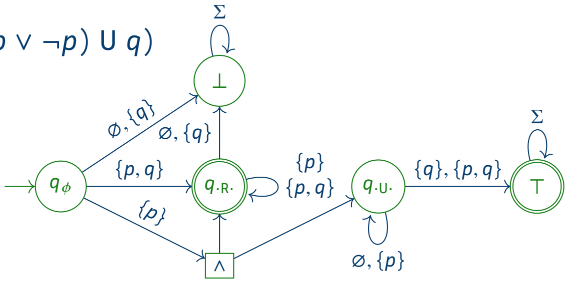
$$\delta(q_{(p \wedge \neg p)\mathsf{R}p}, P) = \delta(p, P) \wedge (\delta(q_{p \wedge \neg p}, P) \vee q_{(p \wedge \neg p)\mathsf{R}p}) \approx \begin{cases} q_{(p \wedge \neg p)\mathsf{R}p} & \text{if } p \in P \\ \bot & \text{if } p \notin P \end{cases}$$

$$\delta(q_{(p \vee \neg p)\mathsf{U}q}, P) = \delta(q, P) \vee (\delta(q_{p \vee \neg p}, P) \wedge q_{(p \vee \neg p)\mathsf{R}q}) \approx \begin{cases} \top & \text{if } q \in P \\ q_{(p \vee \neg p)\mathsf{U}q} & \text{if } q \notin P \end{cases}$$

$$\delta(\phi, P) = \delta(q_{(p \wedge \neg p)\mathsf{R}p}, P) \wedge \delta(q_{(p \vee \neg p)\mathsf{U}q}, P) \approx \begin{cases} \bot & \text{if } P = \emptyset \\ q_{(p \wedge \neg p)\mathsf{R}p} \wedge q_{(p \vee \neg p)\mathsf{U}q} & \text{if } P = \{p\} \\ \bot & \text{if } P = \{q\} \\ q_{(p \wedge \neg p)\mathsf{R}p} & \text{if } P = \{p, q\} \end{cases}$$

Example

consider $\phi = \mathsf{G}\, p \wedge \mathsf{F}\, q \equiv ((p \wedge \neg p)\ \mathsf{R}\ p) \wedge ((p \vee \neg p)\ \mathsf{U}\ q)$



$$\delta(q_p, P) = \begin{cases} \top & \text{if } p \in P \\ \bot & \text{if } p \notin P \end{cases}$$

$$\delta(q_{\neg p}, P) = \begin{cases} \bot & \text{if } p \in P \\ \top & \text{if } p \notin P \end{cases}$$

$$\delta(q_{p\wedge\neg p}, P) = \delta(q_p, P) \wedge \delta(q_{\neg p}, P) = \top \wedge \bot \approx \bot$$

$$\delta(q_{p\vee\neg p}, P) = \delta(q_p, P) \vee \delta(q_{\neg p}, P) = \bot \vee \top \approx \top$$

$$\delta(q_{(p\wedge\neg p)\mathsf{R}p}, P) = \delta(p, P) \wedge (\delta(q_{p\wedge\neg p}, P) \vee q_{(p\wedge\neg p)\mathsf{R}p}) \approx \begin{cases} q_{(p\wedge\neg p)\mathsf{R}p} & \text{if } p \in P \\ \bot & \text{if } p \notin P \end{cases}$$

$$\delta(q_{(p\vee\neg p)\mathsf{U}q}, P) = \delta(q, P) \vee (\delta(q_{p\vee\neg p}, P) \wedge q_{(p\vee\neg p)\mathsf{R}q}) \approx \begin{cases} \top & \text{if } q \in P \\ q_{(p\vee\neg p)\mathsf{U}q} & \text{if } q \notin P \end{cases}$$

$$\delta(\phi, P) = \delta(q_{(p\wedge\neg p)\mathsf{R}p}, P) \wedge \delta(q_{(p\vee\neg p)\mathsf{U}q}, P) \approx \begin{cases} \bot & \text{if } P = \varnothing \\ q_{(p\wedge\neg p)\mathsf{R}p} \wedge q_{(p\vee\neg p)\mathsf{U}q} & \text{if } P = \{p\} \\ \bot & \text{if } P = \{q\} \\ q_{(p\wedge\neg p)\mathsf{R}p} & \text{if } P = \{p, q\} \end{cases}$$

Model Checking

Transition Systems (TSs)

- ★ transition systems capture evolution of state based programs etc.
- ★ they can be seen as finite representations of potentially infinitely many program runs

Transition Systems (TSs)

- ⋆ transition systems capture evolution of state based programs etc.
- ⋆ they can be seen as finite representations of potentially infinitely many program runs

- ⋆ a transition system (TR) is a tuple $\mathcal{S} = (S, \rightarrow, s_I, \lambda)$ where
  1. S is a set of states
  2. $\rightarrow \subseteq S \times S$ is a transition relation
  3. $s_I \in S$ is an initial state
  4. $\lambda : S \rightarrow 2^{\mathcal{P}}$ a labeling of states by propositions $\mathcal{P}$

Transition Systems (TSs)

- transition systems capture evolution of state based programs etc.
- they can be seen as finite representations of potentially infinitely many program runs

- a transition system (TR) is a tuple $\mathcal{S} = (S, \rightarrow, s_I, \lambda)$ where
  1. S is a set of states
  2. $\rightarrow \subseteq S \times S$ is a transition relation
  3. $s_I \in S$ is an initial state
  4. $\lambda : S \rightarrow 2^{\mathcal{P}}$ a labeling of states by propositions $\mathcal{P}$

- we assume $\mathcal{S}$ is total, i.e. every node has a successor: $\forall s \in S. \exists t \in S.\ s \rightarrow t$

Transition Systems (TSs)

- transition systems capture evolution of state based programs etc.
- they can be seen as finite representations of potentially infinitely many program runs

- a transition system (TR) is a tuple $\mathcal{S} = (S, \rightarrow, s_I, \lambda)$ where
  1. S is a set of states
  2. $\rightarrow \subseteq S \times S$ is a transition relation
  3. $s_I \in S$ is an initial state
  4. $\lambda : S \rightarrow 2^{\mathcal{P}}$ a labeling of states by propositions $\mathcal{P}$

- we assume $\mathcal{S}$ is total, i.e. every node has a successor: $\forall s \in S. \exists t \in S.\ s \rightarrow t$

- a run in a total TS is an infinite word $w = P_0 P_1 P_2 \ldots$ such that $\lambda(s_i) = P_i$ for an infinite path

$$s_I = s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow \ldots$$

Transition Systems (TSs)

- ⋆ transition systems capture evolution of state based programs etc.
- ⋆ they can be seen as finite representations of potentially infinitely many program runs

- ⋆ a transition system (TR) is a tuple $\mathcal{S} = (S, \rightarrow, s_I, \lambda)$ where
  1. S is a set of states
  2. $\rightarrow \subseteq S \times S$ is a transition relation
  3. $s_I \in S$ is an initial state
  4. $\lambda : S \rightarrow 2^{\mathcal{P}}$ a labeling of states by propositions $\mathcal{P}$

- ⋆ we assume $\mathcal{S}$ is total, i.e. every node has a successor: $\forall s \in S. \exists t \in S.\ s \rightarrow t$

- ⋆ a run in a total TS is an infinite word $w = P_0 P_1 P_2 \ldots$ such that $\lambda(s_i) = P_i$ for an infinite path

$$s_I = s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow \ldots$$

- ⋆ $\mathrm{L}(\mathcal{S}) \triangleq \{w \mid w \text{ is a run in } \mathcal{S}\}$ is the set of all runs

LTL Model Checking

We are interested in the following decision problem:

- Given: An TS $\mathcal{S} = (S, \rightarrow, s_I, \lambda)$ and specification as LTL formula ϕ
- Question: $L(\mathcal{S}) \subseteq L(\phi)$?

LTL Model Checking

We are interested in the following decision problem:

- Given: An TS $\mathcal{S} = (S, \rightarrow, s_I, \lambda)$ and specification as LTL formula ϕ
- Question: $L(\mathcal{S}) \subseteq L(\phi)$?

Theorem

The above model checking problem is decidable in time $O(|S|^2) \cdot 2^{O(|\phi|)}$

Proof Outline.

LTL Model Checking

We are interested in the following decision problem:

- Given: An TS $\mathcal{S} = (S, \rightarrow, s_I, \lambda)$ and specification as LTL formula ϕ
- Question: $L(\mathcal{S}) \subseteq L(\phi)$?

Theorem

The above model checking problem is decidable in time $O(|S|^2) \cdot 2^{O(|\phi|)}$

Proof Outline.

- let $\mathcal{A}_{\neg\phi} = (Q, 2^{\mathcal{P}}, q_I, \delta, F)$ be the NBA with $L(\neg\phi) = L(\mathcal{A}_{\neg\phi})$ of size $2^{O(|\phi|)}$

LTL Model Checking

We are interested in the following decision problem:

- Given: An TS $\mathcal{S} = (S, \rightarrow, s_I, \lambda)$ and specification as LTL formula ϕ
- Question: $L(\mathcal{S}) \subseteq L(\phi)$?

Theorem

The above model checking problem is decidable in time $O(|S|^2) \cdot 2^{O(|\phi|)}$

Proof Outline.

- let $\mathcal{A}_{\neg\phi} = (Q, 2^{\mathcal{P}}, q_I, \delta, F)$ be the NBA with $L(\neg\phi) = L(\mathcal{A}_{\neg\phi})$ of size $2^{O(|\phi|)}$
- define the NBA $\mathcal{S} \otimes \mathcal{A}_{\neg\phi} \triangleq (S \times Q, \{\bullet\}, (s_I, q_I), \Delta, S \times F)$ where

$$\Delta((s, q), \bullet) \triangleq \{(s', q') \mid s \rightarrow s' \text{ and } q' \in \delta(q, \lambda(s))\}$$

LTL Model Checking

We are interested in the following decision problem:

- Given: An TS $\mathcal{S} = (S, \to, s_I, \lambda)$ and specification as LTL formula ϕ
- Question: $L(\mathcal{S}) \subseteq L(\phi)$?

Theorem

The above model checking problem is decidable in time $O(|S|^2) \cdot 2^{O(|\phi|)}$

Proof Outline.

- let $\mathcal{A}_{\neg\phi} = (Q, 2^{\mathcal{P}}, q_I, \delta, F)$ be the NBA with $L(\neg\phi) = L(\mathcal{A}_{\neg\phi})$ of size $2^{O(|\phi|)}$
- define the NBA $\mathcal{S} \otimes \mathcal{A}_{\neg\phi} \triangleq (S \times Q, \{\bullet\}, (s_I, q_I), \Delta, S \times F)$ where

$$\Delta((s, q), \bullet) \triangleq \{(s', q') \mid s \to s' \text{ and } q' \in \delta(q, \lambda(s))\}$$

- then $L(\mathcal{S}) \subseteq L(\phi) \quad \Longleftrightarrow \quad L(\mathcal{S}) \cap L(\neg\phi) = \varnothing \quad \Longleftrightarrow \quad L(\mathcal{S} \otimes \mathcal{A}_{\neg\phi}) = \varnothing$

LTL Model Checking

We are interested in the following decision problem:

- ⋆ Given: An TS $\mathcal{S} = (S, \rightarrow, s_I, \lambda)$ and specification as LTL formula ϕ
- ⋆ Question: $L(\mathcal{S}) \subseteq L(\phi)$?

Theorem

The above model checking problem is decidable in time $O(|S|^2) \cdot 2^{O(|\phi|)}$

Proof Outline.

- ⋆ let $\mathcal{A}_{\neg\phi} = (Q, 2^{\mathcal{P}}, q_I, \delta, F)$ be the NBA with $L(\neg\phi) = L(\mathcal{A}_{\neg\phi})$ of size $2^{O(|\phi|)}$
- ⋆ define the NBA $\mathcal{S} \otimes \mathcal{A}_{\neg\phi} \triangleq (S \times Q, \{\bullet\}, (s_I, q_I), \Delta, S \times F)$ where

$$\Delta((s, q), \bullet) \triangleq \{(s', q') \mid s \rightarrow s' \text{ and } q' \in \delta(q, \lambda(s))\}$$

- ⋆ then $L(\mathcal{S}) \subseteq L(\phi) \iff L(\mathcal{S}) \cap L(\neg\phi) = \emptyset \iff L(\mathcal{S} \otimes \mathcal{A}_{\neg\phi}) = \emptyset$
- ⋆ emptyness of $\mathcal{S} \otimes \mathcal{A}_{\neg\phi}$ is decidable in time linear in $|\mathcal{S} \otimes \mathcal{A}_{\neg\phi}| \in O(|S|^2) \cdot 2^{O(|\phi|)}$

LTL Model Checking In Practice

Explicit Model Checking: each automaton node is an individual state

- ⋆ SPIN model checker: `http://spinroot.com/`

Symbolic Model Checking: each automaton node represents a set of state, symbolically

- ⋆ SMV model checker: `http://www.cs.cmu.edu/~modelcheck/smv.html`

LTL Model Checking In Practice

Explicit Model Checking: each automaton node is an individual state

- ⋆ SPIN model checker: `http://spinroot.com/`

Symbolic Model Checking: each automaton node represents a set of state, symbolically

- ⋆ SMV model checker: `http://www.cs.cmu.edu/~modelcheck/smv.html`

they have been successfully applied in industrial contexts (see e.g. `http://spinroot.com/spin/success.html`)

LTL Model Checking In Practice

Explicit Model Checking: each automaton node is an individual state

- ⋆ SPIN model checker: `http://spinroot.com/`

Symbolic Model Checking: each automaton node represents a set of state, symbolically

- ⋆ SMV model checker: `http://www.cs.cmu.edu/~modelcheck/smv.html`

they have been successfully applied in industrial contexts (see e.g. `http://spinroot.com/spin/success.html`)

Main Challenge

- ⋆ while real problems have a finite number of states, we deal with an astronmoical number of cases

LTL Model Checking In Practice

Explicit Model Checking: each automaton node is an individual state

- ⋆ SPIN model checker: `http://spinroot.com/`

Symbolic Model Checking: each automaton node represents a set of state, symbolically

- ⋆ SMV model checker: `http://www.cs.cmu.edu/~modelcheck/smv.html`

they have been successfully applied in industrial contexts (see e.g. `http://spinroot.com/spin/success.html`)

Main Challenge

- ⋆ while real problems have a finite number of states, we deal with an astronmoical number of cases
- ⋆ industrial-strength tools such as the ones above generate $\mathcal{S} \otimes \mathcal{A}_{\neg\phi}$ on-the-fly and implement several techniques to combat state-space explosion
  - partial order reduction: detects when an ordering of interleavings is irrelevant. E.g., the $n!$ transitions of n concurrently executing processes is reduced to 1 representative transition, when ordering irrelevant for property under investigation
  - Bounded Model Checking: check that ϕ is violated in $\leq k$ steps
  - ...

Thanks!