A.A. 2003-2004, CDLS in Informatica

Introduction to Formal Methods for SW and HW Development

08: Automata-Theoretic LTL Model Checking

Roberto Sebastiani – rseba@dit.unitn.it

Some material (text, figures) displayed in these slides is courtesy of M. Benerecetti, A. Cimatti, F. Giunchiglia, P. Pandya, M. Pistore, M. Roveri.

Content

The problem

- Given a Kripke structure M and an LTL specification ψ, does M satisfy ψ?:

$$M \models \psi$$

- Equivalent to the CTL* M.C. problem:

$$M \models \mathbf{A}\psi$$

- Dual CTL* M.C. problem:

$$M \models \mathbf{E}\neg\psi$$

Automata-Theoretic LTL Model Checking

- $M \models \mathbf{A}\psi$ (CTL*)

$\Longleftrightarrow M \models \psi$ (LTL)

$\Longleftrightarrow \mathcal{L}(M) \subseteq \mathcal{L}(\psi)$

$\Longleftrightarrow \mathcal{L}(M) \cap \overline{\mathcal{L}(\psi)} = \{\}$

$\Longleftrightarrow \mathcal{L}(A_M) \cap \mathcal{L}(A_{\neg\psi}) = \{\}$

$\Longleftrightarrow \mathcal{L}(A_M \times A_{\neg\psi}) = \{\}$

- A_M is a Büchi Automaton equivalent to M (which represents all and only the executions of M)
- $A_{\neg\psi}$ is a Büchi Automaton which represents all and only the paths that satisfy $\neg\psi$ (do not satisfy ψ)

$\Longrightarrow$ $A_M \times A_{\neg\psi}$ represents all and only the paths appearing in M and not in ψ.

Automata-Theoretic LTL M.C. (dual version)

- $M \models \mathbf{E}\varphi$

$\Longleftrightarrow$ $M \not\models \mathbf{A}\neg\varphi$

$\Longleftrightarrow$ ...

$\Longleftrightarrow$ $\mathcal{L}(A_M \times A_\varphi) \neq \{\}$

- A_M is a Büchi Automaton equivalent to M (which represents all and only the executions of M)
- A_φ is a Büchi Automaton which represents all and only the paths that satisfy φ

$\Longrightarrow$ $A_M \times A_\varphi$ represents all and only the paths appearing in both A_M and A_φ.

Automata-Theoretic LTL Model Checking

Four steps:

1. Compute A_M
2. Compute A_φ
3. Compute the product $A_M \times A_\varphi$
4. Check the emptiness of $\mathcal{L}(A_M \times A_\varphi)$

Content

Finite Word Languages

- An Alphabet Σ is a collection of symbols (letters).
  E.g. $\Sigma = \{a, b\}$.
- A finite word is a finite sequence of letters. (E.g. $aabb$.)
  The set of all finite words is denoted by Σ^*.
- A language U is a set of words, i.e. $U \subseteq \Sigma^*$.

Example: Words over $\Sigma = \{a, b\}$ with equal number of a's and b's. (E.g. $aabb$ or $abba$.)

Language recognition problem:
determine whether a word belongs to a language.

Automata are computational devices able to solve language recognition problems.

Finite State Automata

Basic model of computational systems with finite memory.

Widely applicable

- ▷ Embedded System Controllers.

  Languages: Ester-el, Lustre, Verilog.

- ▷ Synchronous Circuits.
- ▷ Regular Expression Pattern Matching

  Grep, Lex, Emacs.

- ▷ Protocols

  Network Protocols

  Architecture: Bus, Cache Coherence, Telephony,...

Notation

$a, b \in \Sigma$ finite alphabet.

$u, v, w \in \Sigma^*$ finite words.

- λ empty word.
- $u.v$ catenation.
- $u^i = u.u. \ldots .u$ repeated i-times.

$U, V \subseteq \Sigma^*$ Finite word languages.

FSA Definition

Nondeterministic Finite State Automaton (NFA):

NFA is $(Q, \Sigma, \delta, I, F)$

Q Finite set of states.

$I \subseteq Q$ set of initial states.

$F \subseteq Q$ set of final states.

$\rightarrow \subseteq Q \times \Sigma \times Q$ transition relation (edges).

We use $q \xrightarrow{a} q'$ to denote $(q, a, q') \in \delta$.

Deterministic Finite State Automaton (DFA):

DFA has $\delta : Q \times \Sigma \rightarrow Q$, a total function.

Single initial state $I = \{q_0\}$.

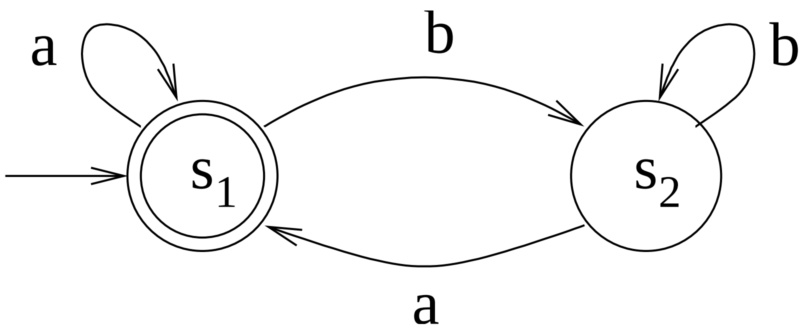
Regular Languages

- A run of NFA A on $u = a_0, a_1, \ldots, a_{n-1}$ is a finite sequence of states $q_0, q_1, \ldots, q_n$ s.t. $q_0 \in I$ and $q_i \xrightarrow{a_i} q_{i+1}$ for $0 \leq i < n$.
- An accepting run is one where the last state $q_n \in F$.
- The language accepted by A
  $\mathcal{L}(A) = \{u \in \Sigma^* \mid A \text{ has an accepting run on } u\}$
- The languages accepted by a NFA are called regular languages.

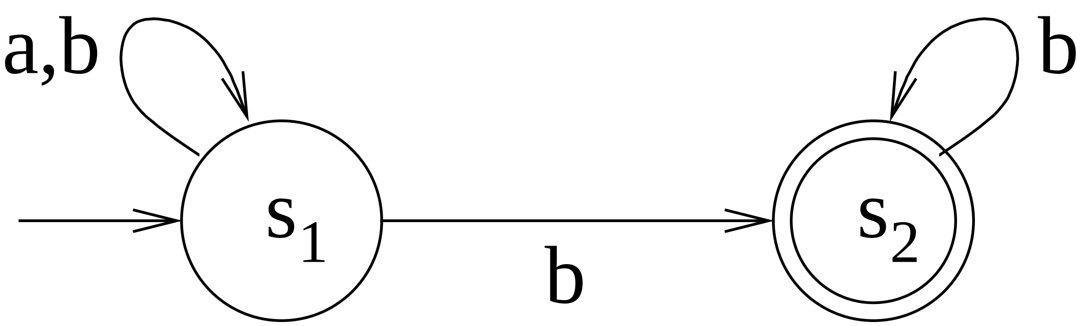
Finite State Automata

Example: DFA A_1 over $\Sigma = \{a, b\}$.

Recognizes words which do not end in b.



NFA A_2. Recognizes words which end in b.

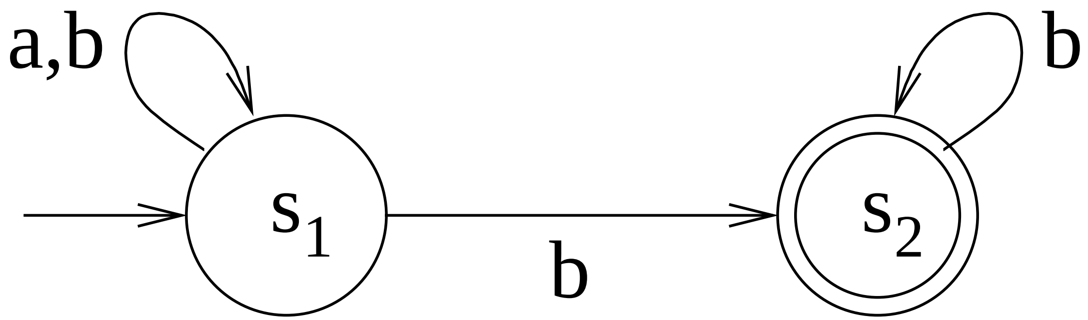


Determinisation

Theorem (determinisation) Given a NFA A we can construct a DFA A' s.t. $\mathcal{L}(A) = \mathcal{L}(A')$. Size $|A'| = 2^{O(|A|)}$.

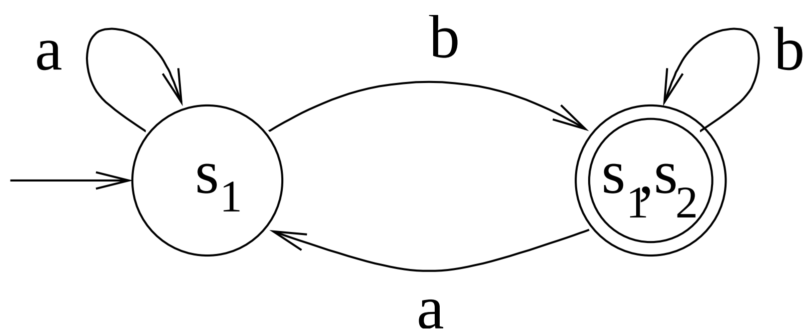
Determinisation [cont.]

NFA A_2: Words which end in b.



A_2 can be determinised into the automaton DA_2 below.

States = 2^Q.



Study Topic There are NFA's of size n for which the size of the minimum sized DFA must have size $O(2^n)$.

Closure Properties

Theorem (boolean closure) Given NFA A_1, A_2 over Σ we can construct NFA A over Σ s.t.

- $\mathcal{L}(A) = \overline{\mathcal{L}(A_1)}$ (Complement). $|A| = 2^{O(|A_1|)}$.
- $\mathcal{L}(A) = \mathcal{L}(A_1) \cup \mathcal{L}(A_2)$ (union). $|A| = |A_1| + |A_2|$.
- $\mathcal{L}(A) = \mathcal{L}(A_1) \cap \mathcal{L}(A_2)$ (intersection). $|A| = |A_1| \cdot |A_2|$.

Complementation of a NFA

A NFA $A = (Q, \Sigma, \delta, I, F)$ is complemented by:

- ▷ determinizing it into a DFA $A' = (Q', \Sigma', \delta', I', F')$
- ▷ complementing it: $\overline{A'} = (Q', \Sigma', \delta', I', \overline{F'})$
- ▷ $|\overline{A'}| = |A'| = 2^{O(|A_1|)}$

Union of two NFA's

Two NFA's $A_1 = (Q_1, \Sigma_1, \delta_1, I_1, F_1)$, $A_2 = (Q_2, \Sigma_2, \delta_2, I_2, F_2)$, $A = A_1 \cup A_2 = (Q, \Sigma, \delta, I, F)$ is defined as follows

- $Q := Q_1 \cup Q_2$, $I := I_1 \cup I_2$, $F := F_1 \cup F_2$
- $R(s,s') := \begin{cases} R_1(s,s') \; if \; s \in Q_1 \\ R_2(s,s') \; if \; s \in Q_2 \end{cases}$

$\Longrightarrow$ A is an automaton which just runs nondeterministically either A_1 or A_2

- $\mathcal{L}(A) \; = \; \mathcal{L}(A_1) \cup \mathcal{L}(A_2)$
- $|A| = |A_1| + |A_2|$

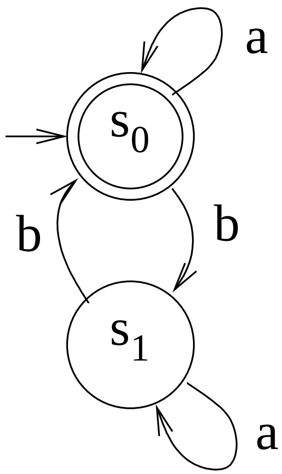
Synchronous Product Construction

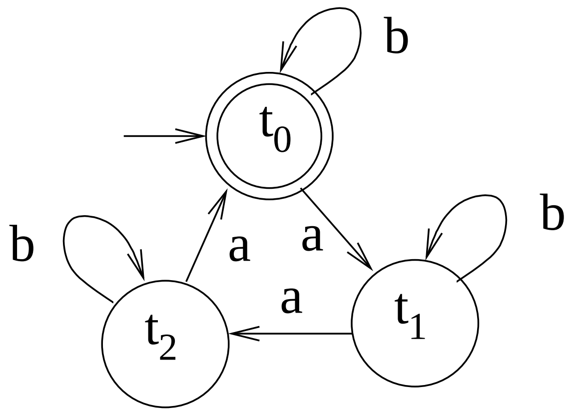
Let $A_1 = (Q_1, \Sigma, \delta_1, I_1, F_1)$ and $A_2 = (Q_2, \Sigma, \delta_2, I_2, F_2)$. Then, $A_1 \times A_2 = (Q, \Sigma, \delta, I, F)$ where

- $Q = Q_1 \times Q_2$. $I = I_1 \times I_2$.
  $F = F_1 \times F_2$.
- $< p, q > \stackrel{a}{\longrightarrow} < p', q' >$ iff $p \stackrel{a}{\longrightarrow} p'$ and $q \stackrel{a}{\longrightarrow} q'$.

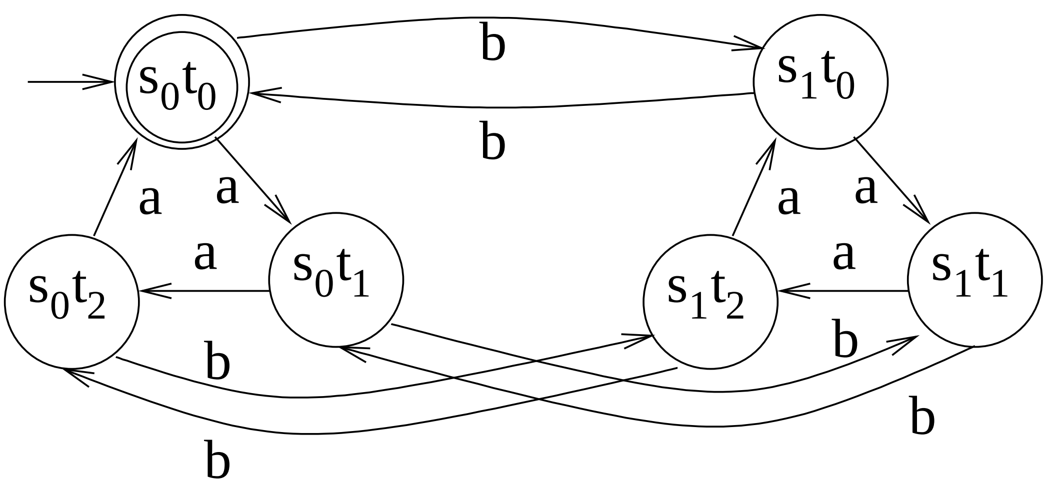
Theorem $\mathcal{L}(A_1 \times A_2) = \mathcal{L}(A_1) \cap \mathcal{L}(A_2)$.

Example





- A_1 recognizes words with an even number of b.
- A_2 recognizes words with a number of $a\ mod\ 3\ = 0$.
- The Product Automaton $A_1 \times A_2$ with $F = \{s_0, t_0\}$.



Synchronized Product Construction

Let $A_1 = (Q_1, \Sigma_1, \delta_1, I_1, F_1)$ and $A_2 = (Q_2, \Sigma_2, \delta_2, I_2, F_2)$.
Then,

$A_1 || A_2 = (Q, \Sigma, \delta, I, F)$, where

- $Q = Q_1 \times Q_2$. $\Sigma = \Sigma_1 \cup \Sigma_2$.
  $I = I_1 \times I_2$. $F = F_1 \times F_2$.

- $< p, q > \stackrel{a}{\longrightarrow} < p', q' >$ if $a \in \Sigma_1 \cap \Sigma_2$ and $p \stackrel{a}{\longrightarrow} p'$ and $q \stackrel{a}{\longrightarrow} q'$.
- $< p, q > \stackrel{a}{\longrightarrow} < p', q >$ if $a \in \Sigma_1$, $a \not\in \Sigma_2$ and $p \stackrel{a}{\longrightarrow} p'$.
- $< p, q > \stackrel{a}{\longrightarrow} < p, q' >$ if $a \not\in \Sigma_1$, $a \in \Sigma_2$ and $q \stackrel{a}{\longrightarrow} q'$.

Asynchronous Product Construction

Let $A_1 = (Q_1, \Sigma_1, \delta_1, I_1, F_1)$ and $A_2 = (Q_2, \Sigma_2, \delta_2, I_2, F_2)$.
Then,

$A_1 \,||_A\, A_2 = (Q, \Sigma, \delta, I, F)$, where

- $Q = Q_1 \times Q_2$. $\Sigma = \Sigma_1 \cup \Sigma_2$.
  $I = I_1 \times I_2$. $F = F_1 \times F_2$.

- $< p, q > \stackrel{a}{\longrightarrow} < p', q >$ if $a \in \Sigma_1$ and $p \stackrel{a}{\longrightarrow} p'$.
- $< p, q > \stackrel{a}{\longrightarrow} < p, q' >$ if $a \in \Sigma_2$ and $q \stackrel{a}{\longrightarrow} q'$.

Decision Problems

Theorem (Emptiness) Given a NFA A we can decide whether $\mathcal{L}(A) = \emptyset$.

Method Forward/Backward Reachability of acceptance states in Automaton graph. Complexity is $O(|Q| + |\delta|)$.

Theorem (Language Containment) Given NFA A_1 and A_2 we can decide whether $\mathcal{L}(A_1) \subseteq \mathcal{L}(A_2)$.

Method: $\mathcal{L}(A_1) \subseteq \mathcal{L}(A_2)$ iff $\mathcal{L}(A_1) \cap \overline{\mathcal{L}(A_2)} = \emptyset$. Complexity is $O(|A_1| \cdot 2^{|A_2|})$.

N.B. **Model Checking**:

Typically, $\mathcal{L}(A_1 \times A_2 \times \ldots \times A_n) \quad \subseteq \quad \mathcal{L}(A_{prop})$.

Regular Expressions

Syntax: $\emptyset \mid \varepsilon \mid a \mid reg_1.reg_2 \mid reg_1 + reg_2 \mid reg^*$.

Every regular expression *reg* denotes a language $\mathcal{L}(reg)$.

Example: $(a^*.(b+bb).a^*$. The words with either 1 b or 2 consecutive b's.

Theorem: For every regular expression *reg* we can construct a language equivalent NFA of size $O(|reg|)$.

Theorem: For every DFA A we can construct a language equivalent regular expression $reg(A)$.

Content

Infinite Word Languages

Modeling infinite computations of reactive systems.

- An ω-word α over Σ is infinite sequence

  $$a_0,\ a_1,\ a_2 \ldots.$$

  Formally, $\alpha : \mathbb{N} \rightarrow \Sigma$.

  The set of all infinite words is denoted by Σ^ω.

- A ω-language L is collection of ω-words, i.e. $L \subseteq \Sigma^\omega$.

  Example All words over $\{a, b\}$ with infinitely many a's.

Notation

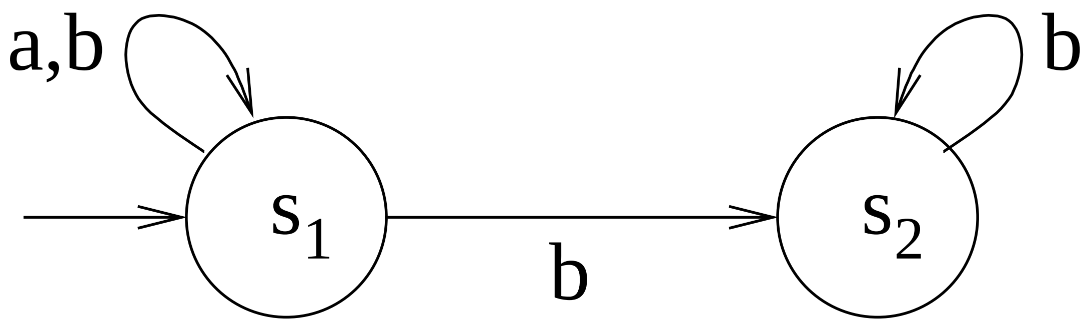
omega words $\alpha, \beta, \gamma \in \Sigma^\omega$.

omega-languages $L, L_1 \subseteq \Sigma^\omega$

For $u \in \Sigma^+$, let $u^\omega = u.u.u \ldots$

Omega-Automata

We consider automaton runs over infinite words.



Let $\alpha = aabbbb\ldots$. There are several possible runs.

Run $\rho_1 = s_1, s_1, s_1, s_1, s_2, s_2 \ldots$

Run $\rho_2 = s_1, s_1, s_1, s_1, s_1, s_1 \ldots$

Acceptance Conditions Büchi, (Muller, Rabin, Street).

Acceptance is based on states occurring infinitely often

Notation Let $\rho \in Q^{\omega}$. Then,

$$Inf(\rho) = \{s \in Q \mid \exists^{\infty} i \in \mathbb{N}.\ \rho(i) = s\}.$$

Büchi Automata

Nondeterministic Büchi Automaton

$A = (Q, \Sigma, \delta, I, F)$, where $F \subseteq Q$ is the set of accepting states.

- A run ρ of A on omega word α is an infinite sequence $\rho = q_o, q_1, q_2, \ldots$ s.t. $q_0 \in I$ and $q_i \xrightarrow{a_i} q_{i+1}$ for $0 \leq i$.
- The run ρ is accepting if

  $$Inf(\rho) \cap F \neq \emptyset.$$

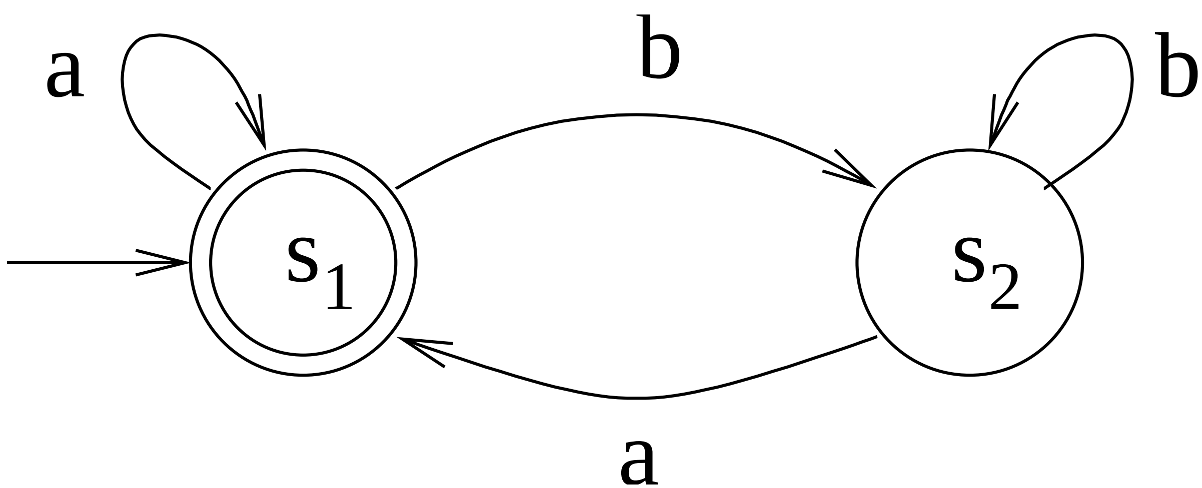
- The language accepted by A

  $\mathcal{L}(A) = \{\alpha \in \Sigma^\omega \mid A \text{ has an accepting run on } \alpha\}$

Büchi Automaton: Example

Let $\Sigma = \{a, b\}$.

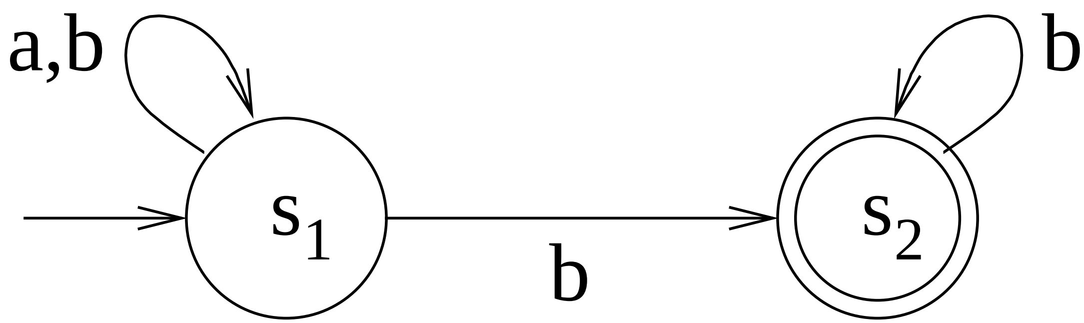
Let a Deterministic Büchi Automaton (DBA) A_1 be



- With $F = \{s_1\}$ the automaton recognizes words with infinitely many a's.
- With $F = \{s_2\}$ the automaton recognizes words with infinitely many b's.

Büchi Automaton: Example (2)

Let a Nondeterministic Büchi Automaton (NBA) A_2 be



With $F = \{s_2\}$, automaton A_2 recognizes words with finitely many a.Thus, $\mathcal{L}(A_2) = \overline{\mathcal{L}(A_1)}$.

Deterministic vs. Nondeterministic Büchi Automata

Theorem DBA's are strictly less powerful than NBA's.

Closure Properties

Theorem (union, intersection)

For the NBA's A_1, A_2 we can construct

– the NBA A s.t. $\mathcal{L}(A) = \mathcal{L}(A_1) \cup \mathcal{L}(A_2)$. $|A| = |A_1| + |A_2|$

– the NBA A s.t. $\mathcal{L}(A) = \mathcal{L}(A_1) \cap \mathcal{L}(A_2)$. $|A| = |A_1| \cdot |A_2| \cdot 2$.

Union of two NBA's

Two NBA's $A_1 = (Q_1, \Sigma_1, \delta_1, I_1, F_1)$, $A_2 = (Q_2, \Sigma_2, \delta_2, I_2, F_2)$, $A = A_1 \cup A_2 = (Q, \Sigma, \delta, I, F)$ is defined as follows

- ▷ $Q := Q_1 \cup Q_2$, $I := I_1 \cup I_2$, $F := F_1 \cup F_2$
- ▷ $R(s, s') := \begin{cases} R_1(s, s') \; if \; s \in Q_1 \\ R_2(s, s') \; if \; s \in Q_2 \end{cases}$

$\Longrightarrow$ A is an automaton which just runs nondeterministically either A_1 or A_2

- ▷ $\mathcal{L}(A) \; = \; \mathcal{L}(A_1) \cup \mathcal{L}(A_2)$
- ▷ $|A| = |A_1| + |A_2|$
- ▷ (same construction as with ordinary automata)

Synchronous Product of NBA's

Let $A_1 = (Q_1, \Sigma, \delta_1, I_1, F_1)$ and $A_2 = (Q_2, \Sigma, \delta_2, I_2, F_2)$.

Then, $A_1 \times A_2 = (Q, \Sigma, \delta, I, F)$, where

$$Q = Q_1 \times Q_2 \times \{1, 2\}.$$

$$I = I_1 \times I_2 \times \{1\}.$$

$$F = F_1 \times Q_2 \times \{1\}.$$

$< p, q, 1 > \xrightarrow{a} < p', q', 1 >$ iff $p \xrightarrow{a} p'$ and $q \xrightarrow{a} q'$ and $p \not\in F_1$.

$< p, q, 1 > \xrightarrow{a} < p', q', 2 >$ iff $p \xrightarrow{a} p'$ and $q \xrightarrow{a} q'$ and $p \in F_1$.

$< p, q, 2 > \xrightarrow{a} < p', q', 2 >$ iff $p \xrightarrow{a} p'$ and $q \xrightarrow{a} q'$ and $q \not\in F_2$.

$< p, q, 2 > \xrightarrow{a} < p', q', 1 >$ iff $p \xrightarrow{a} p'$ and $q \xrightarrow{a} q'$ and $q \in F_2$.

Theorem $\mathcal{L}(A_1 \times A_2) = \mathcal{L}(A_1) \cap \mathcal{L}(A_2)$.

Product of NBA's: Intuition

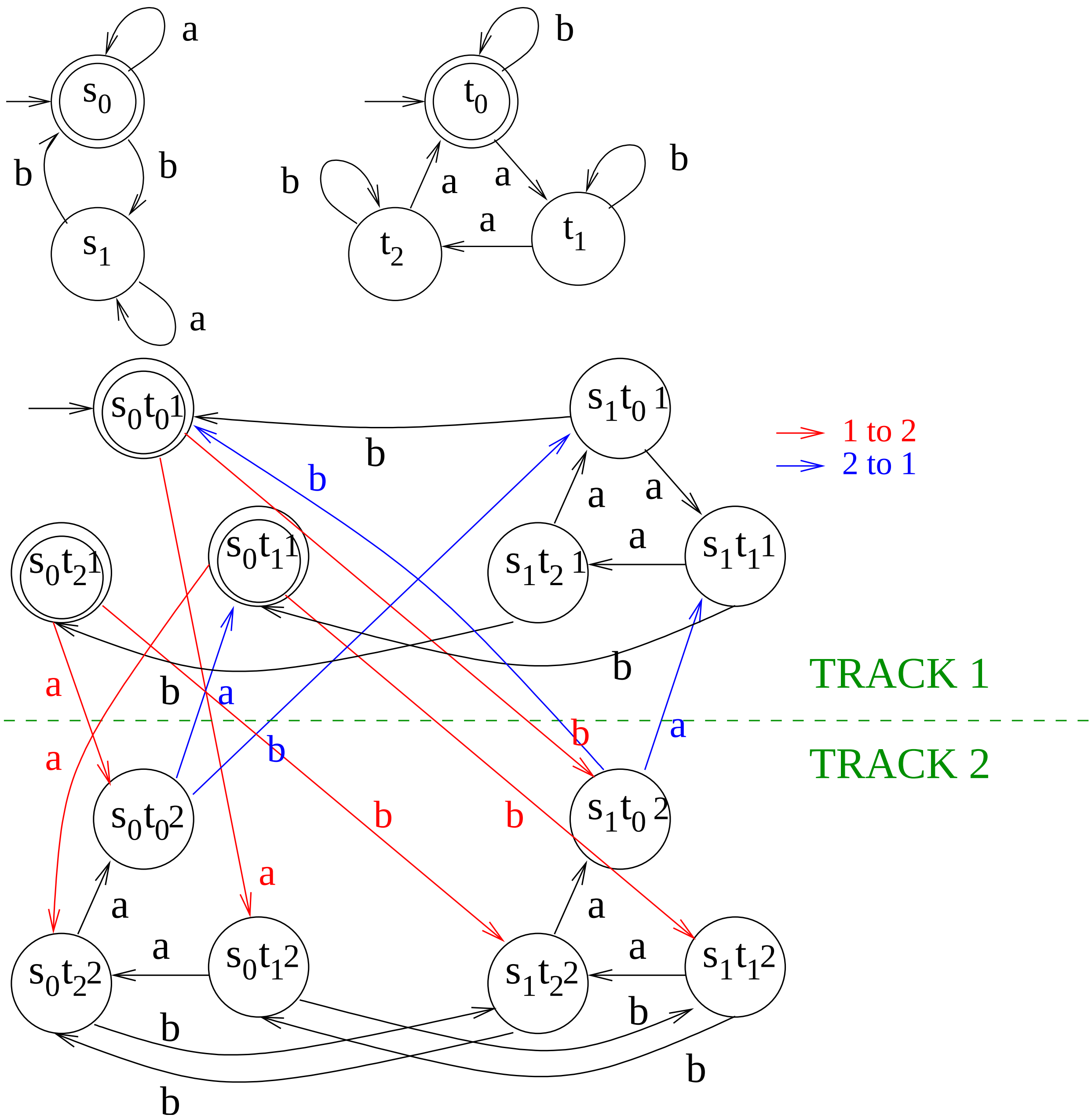
- ▷ The automaton remembers two tracks, one for each source NBA, and it points to one of the two tracks
- ▷ As soon as it goes through an accepting state of the current track, it switches to the other track

$\Longrightarrow$ to visit infinitely often a state in F (i.e., F_1), it must visit infinitely often some state also in F_2

- ▷ Important subcase: If $F_2 = Q_2$, then

$$Q = Q_1 \times Q_2.$$
$$I = I_1 \times I_2.$$
$$F = F_1 \times Q_2.$$

Product of NBA's: Example



Closure Properties (2)

Theorem (complementation)

For the NBA A_1 we can construct an NBA A_2 such that $\mathcal{L}(A_2) = \overline{\mathcal{L}(A_1)}$.

$|A_2| = O(2^{|A_1| \cdot \log(|A_1|)})$.

Method: (hint)

(1) convert a Büchi automaton into a Non-Deterministic Rabin automaton.

(2) Determinize and Complement the Rabin automaton

(3) convert the Rabin automaton into a Büchi automaton

Generalized Büchi Automaton

A Generalized Büchi Automaton is $A := (Q, \Sigma, \delta, I, FT)$ where $FT = < F_1, F_2, \ldots, F_k >$ with $F_i \subseteq Q$.

A run ρ of A is accepting if $Inf(\rho) \cap F_i \neq \emptyset$ for each $1 \leq i \leq k$.

Theorem For every Generalized Büchi Automaton (A, FT) we can construct a language equivalent Büchi Automaton (A', G').

Construction (Hint) Let $Q' = Q \times \{1, \ldots, k\}$.
Automaton remains in i phase till it visits a state in F_i. Then, it moves to $i+1$ mode. After phase k it moves to phase 1.

Size: $|A'| \leq |A| \cdot k$.

Omega Regular Expressions

A language is called ω-regular if it has the form $\cup_{i=1}^{n}\ U_i.(V_i)^{\omega}$ where U_i, V_i are regular languages.

Theorem A language L is ω-regular iff it is NBA-recognizable.

Decision Problem

Emptiness For a NBA A, it is decidable whether $\mathcal{L}(A) = \emptyset$.

Method

- ▷ Find the maximal strongly connected components (MSCC) in the graph of A (disregarding the edge labels).
- ▷ A MSCC C is called non-trivial if $C \cap F \neq \emptyset$ and C has at least one edge.
- ▷ Find all nodes from which there is a path to a non-trivial SCC. Call the set of these nodes as N.
- ▷ $\mathcal{L}(A) = \emptyset$ iff $N \cap I = \emptyset$.

Time Complexity: $O(|Q| + |\delta|)$.

Content

Computing a NBA A_M from a Kripke Structure M

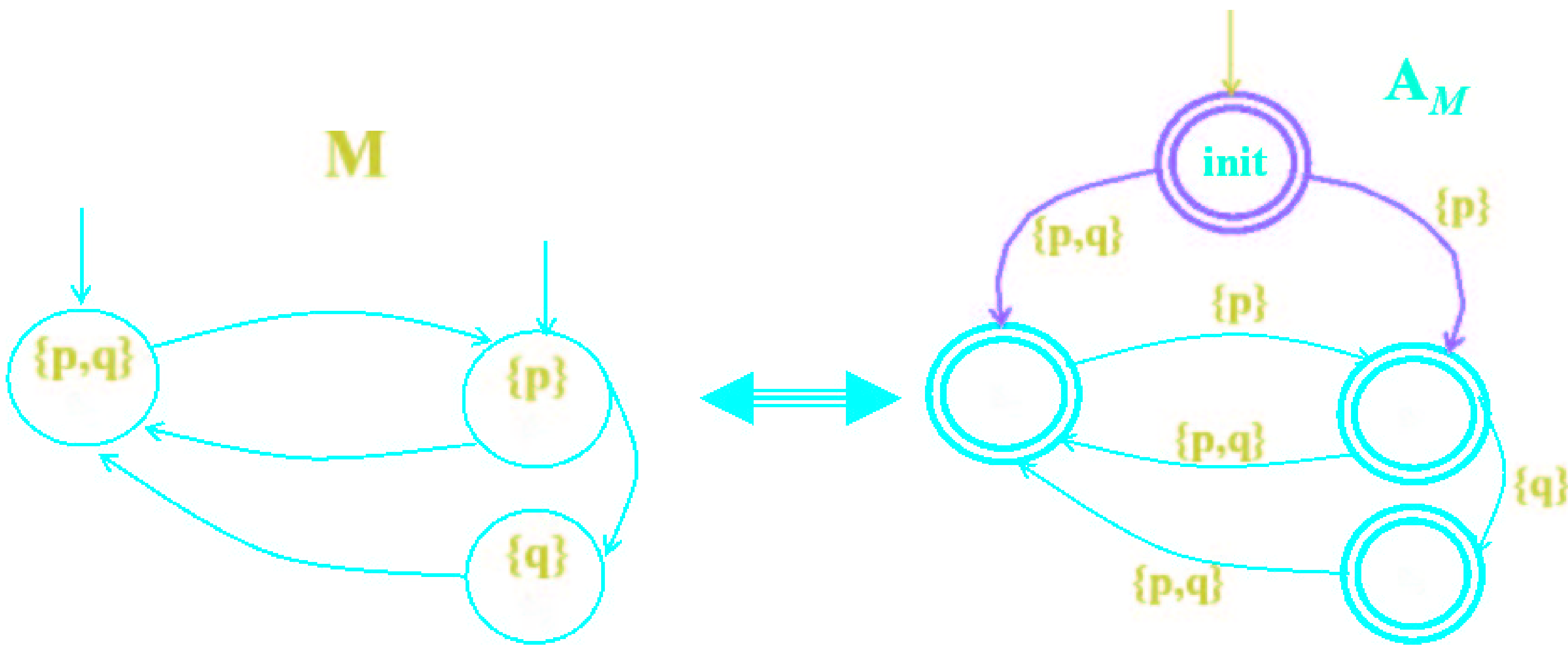
- ▷ Transforming a K.S. $M = \langle S, S_0, R, L, AP \rangle$ into an NBA $A_M = \langle Q, \Sigma, \delta, I, F \rangle$ s.t.:
  - States: $Q := S \cup \{init\}$, *init* being a new initial state
  - Alphabet: $\Sigma := 2^{AP}$
  - Initial State: $I := \{init\}$
  - Accepting States: $F := Q = S \cup \{init\}$
  - Transitions:

$$\delta: \quad q \xrightarrow{a} q' \text{ iff } (q, q') \in R \; and \; L(q') = a$$

$$init \xrightarrow{a} q \text{ iff } q \in S_0 \; and \; L(q') = a$$

- ▷ $\mathcal{L}(A_M) = \mathcal{L}(M)$
- ▷ $|A_M| = |M| + 1$

Computing a NBA A_M from a Kripke Structure M: Example



$\Longrightarrow$Substantially, add one initial state, move labels from states to incoming edges, set all states as accepting states

Computing a NBA A_M from a Fair Kripke Structure M

- Transforming a fair K.S. $M = \langle S, S_0, R, L, AP, FT \rangle$, $FT = \{F_1, ..., F_n\}$, into an NBA $A_M = \langle Q, \Sigma, \delta, I, F \rangle$ s.t.:
  - States: $Q := S \cup \{init\}$, $init$ being a new initial state
  - Alphabet: $\Sigma := 2^{AP}$
  - Initial State: $I := \{init\}$
  - Accepting States: $F := FT$
  - Transitions:

$$\delta : \quad q \xrightarrow{a} q' \text{ iff } (a, a') \in R \; and \; L(q') = a$$

$$init \xrightarrow{a} q \text{ iff } q \in S_0 \; and \; L(q') = a$$

- $\mathcal{L}(A_M) = \mathcal{L}(M)$
- $|A_M| = |M| + 1$

Content

Paths as ω-words

Let φ be an LTL formula.

- $Var(\varphi)$ denotes the set of free variables of φ.

  E.g. $Var(p \wedge (\neg q \mathbf{U} q)) = \{p, q\}$.

- Let $\Sigma := 2^{Var(\varphi)}$.

  $\Longrightarrow$ a model for φ is an ω-word $\alpha = a_0, a_1, \ldots$ in Σ^ω.

- We can define $\alpha, i \models \varphi$. Also, $\alpha \models \varphi$ iff $\alpha, 0 \models \varphi$.

  Example A model of $p \wedge (\neg q \mathbf{U} q)$ is the ω-word

  $$\{p\}, \{\}, \{q\}, \{p, q\}^\omega.$$

- N.B.: correspondence between ω-words and paths in Kripke structures:

  $$\alpha, i \models \varphi \Longleftrightarrow \pi, s_i \models \varphi, \quad \alpha, 0 \models \varphi \Longleftrightarrow \pi, s_0 \models \varphi$$

Automata for LTL model checking

Let $Mod(\varphi)$ denote the set of models of φ.

Theorem For any LTL formula φ, the set $Mod(\varphi)$ is omega-regular.

$\Longrightarrow$ Technique: Construct a (Generalized) NBA A_φ such that $Mod(\varphi) \; = \; \mathcal{L}(A_\varphi)$.

Closures

Closure Given $\varphi \in LTL$, let $CL'(\varphi)$ be the smallest set s.t.

- $\varphi \in CL'(\varphi)$.
- If $\neg\varphi_1 \in CL'(\varphi)$ then $\varphi_1 \in CL'(\varphi)$.
- If $\varphi_1 \vee \varphi_2 \in CL'(\varphi)$ then $\varphi_1, \varphi_2 \in CL'(\varphi)$.
- If $X\varphi_1 \in CL'(\varphi)$ then $\varphi_1 \in CL'(\varphi)$.
- If $(\varphi_1 \mathbf{U} \varphi_2) \in CL'(\varphi)$ then $\varphi_1, \varphi_2 \in CL'(\varphi)$ and $X(\varphi_1 \mathbf{U} \varphi_2) \in CL'(\varphi)$

$CL(\varphi) := \{\varphi_1, \neg\varphi_1 \mid \varphi_1 \in CL'(\varphi)\}$ (we identify $\neg\neg\varphi_1$ with φ_1.)

N.B.: $|CL(\varphi)| = O(|\varphi|)$.

Atoms

An Atom is a maximal consistent subset of $CL(\varphi)$.

▷ Definition A set $A \subseteq CL(\varphi)$ is called an atom if

- For all $\varphi_1 \in CL(\varphi)$, we have $\varphi_1 \in A$ iff $\neg\varphi_1 \notin A$.
- For all $\varphi_1 \vee \varphi_2 \in CL(\varphi)$, we have $\varphi_1 \vee \varphi_2 \in A$ iff $\varphi_1 \in A$ or $\varphi_2 \in A$ (or both).
- For all $(\varphi_1 \mathbf{U} \varphi_2) \in CL(\varphi)$, we have $(\varphi_1 \mathbf{U} \varphi_2) \in A$ iff $\varphi_2 \in A$ or ($\varphi_1 \in A$ and $X(\varphi_1 \mathbf{U} \varphi_2) \in A$).

▷ In practice, an atom is a consistent truth assignment to the elementary subformulas of φ', φ' being the result of applying the tableau expansion rules to φ

▷ We call $Atoms(\varphi)$ the set of all atoms of φ.

Definition of A_φ

For an LTL formula φ, we construct a Generalized NBA $A_\varphi = (Q, \Sigma, \delta, Q_0, FT)$ as follows:

- ▷ $\Sigma = 2^{vars(\varphi)}$
- ▷ $Q = Atoms(\varphi)$, the set of atoms.
- ▷ δ is s.t., for $q, q' \in Atoms(\varphi)$ and $a \in \Sigma$, $q \xrightarrow{a} q'$ holds in δ iff
  - $q' \cap Var(\varphi) = a$,
  - for all $X\varphi_1 \in CL(\varphi)$, we have $X\varphi_1 \in q$ iff $\varphi_1 \in q'$.
- ▷ $Q_0 = \{q \in Atoms(\varphi) \mid \varphi \in q\}$.
- ▷ $FT = (F_1, F_2, \ldots, F_k)$ where, for all $(\psi_i \mathbf{U} \varphi_i)$ occurring positively in φ,

$$F_i = \{q \in Atoms(\varphi) \mid (\psi_i \mathbf{U} \varphi_i) \notin q \ \ or\ \ \varphi_i \in q\}.$$

Definition of A_φ [cont.]

Theorem Let $\alpha = a_0, a_1, \ldots \in \Sigma^\omega$. Then,

$\alpha \models \varphi$ iff $\alpha \in \mathcal{L}(A_\varphi)$.

Size: $|A_\varphi| = O(2^{|\varphi|})$.

LTL Negative Normal Form (NNF)

- Every LTL formula φ can be written as equivalent formula φ' using only the operators $\neg$, $\vee$ X and $\mathbf{U}$.

- We can further push negations down to literal level:

$$\neg(\varphi_1 \vee \varphi_2) \Longrightarrow (\neg\varphi_1 \wedge \neg\varphi_2)$$
$$\neg(\varphi_1 \wedge \varphi_2) \Longrightarrow (\neg\varphi_1 \vee \neg\varphi_2)$$
$$\neg\mathbf{X}\varphi_1 \Longrightarrow \mathbf{X}\neg\varphi_1$$
$$\neg(\varphi_1\mathbf{U}\varphi_2) \Longrightarrow (\neg\varphi_1\mathbf{R}\neg\varphi_2)$$

$\Longrightarrow$the resulting formula is expressed in terms of $\vee$, $\wedge$, X, $\mathbf{U}$, $\mathbf{R}$ and literals (Negative Normal Form, NNF).

- In the construction of A_φ we now assume that φ is in NNF.

Construction of A_φ (Schema)

Apply recursively the following steps:

Step 1: Apply the tableau expansion rules to φ

$$\psi_1 \mathbf{U} \psi_2 \Longrightarrow \psi_2 \vee (\psi_1 \wedge \mathbf{X}(\psi_1 \mathbf{U} \psi_2))$$

$$\psi_1 \mathbf{R} \psi_2 \Longrightarrow \psi_2 \wedge (\psi_1 \vee \mathbf{X}(\psi_1 \mathbf{R} \psi_2))$$

until we get a boolean combination of elementary subformulas of φ

Construction of A_φ (Schema) [cont.]

Step 2: Convert all formulas into Disjunctive Normal Form:

$$\bigvee_i (\bigwedge_j l_{ij} \wedge \bigwedge_k \mathbf{X}\psi_{ik})$$

- Each disjunct $(\overbrace{\bigwedge_j l_{ij}}^{labels} \wedge \overbrace{\bigwedge_k \mathbf{X}\psi_{ik}}^{next\ \ part})$ represents a state:
  - the conjunction of literals $\bigwedge_j l_{ij}$ represents a set of labels in Σ (e.g., if $Vars(\varphi) = \{p, q, r\}$, $p \wedge \neg q$ represents the two labels $\{p, \neg q, r\}$ and $\{p, \neg q, \neg r\}$)
  - $\bigwedge_k \mathbf{X}\psi_{ik}$ represents the next part of the state (obbligations for the successors)
- N.B., if no next part occurs, $\mathbf{X}\top$ is implicitly assumed

Construction of A_φ (Schema) [cont.]

Step 3: For every state represented by $(\bigwedge_j l_{ij} \wedge \bigwedge_k \mathbf{X}\psi_{ik})$

- ▷ draw an edge to all states which satisfy $\bigwedge_k \psi_{ik}$
- ▷ label the incoming edges with $\bigwedge_j l_{ij}$

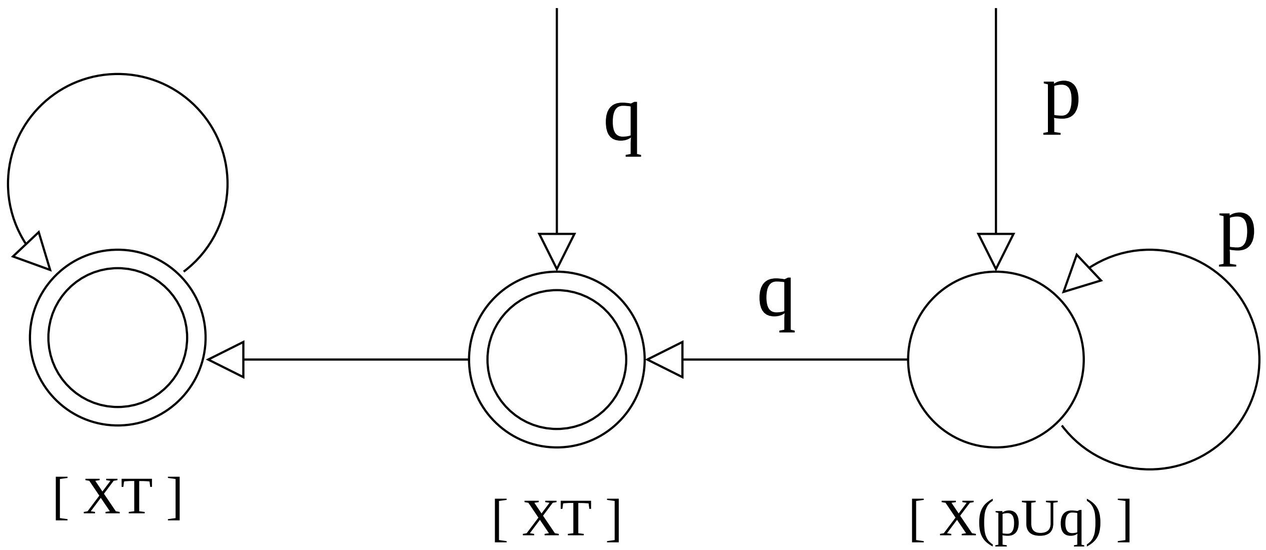
N.B., if no next part occurs, $\mathbf{X}\top$ is implicitly assumed, so that an edge to a “true” node is drawn

Construction of A_φ (Schema) [cont.]

Step 4: For every $\psi_i \mathbf{U} \varphi_i$, for every state q_j, mark q_j with F_i iff $(\psi_i \mathbf{U} \varphi_i) \notin q_j$ *or* $\varphi_i \in q_j$

Example: $p\mathbf{U}q$

$$\begin{aligned}\varphi &= p\mathbf{U}q \\ &= q \vee (p \wedge \mathbf{X}(p\mathbf{U}q)) \\ &= (q \wedge \mathbf{X}\top) \vee (p \wedge \mathbf{X}(p\mathbf{U}q))\end{aligned}$$



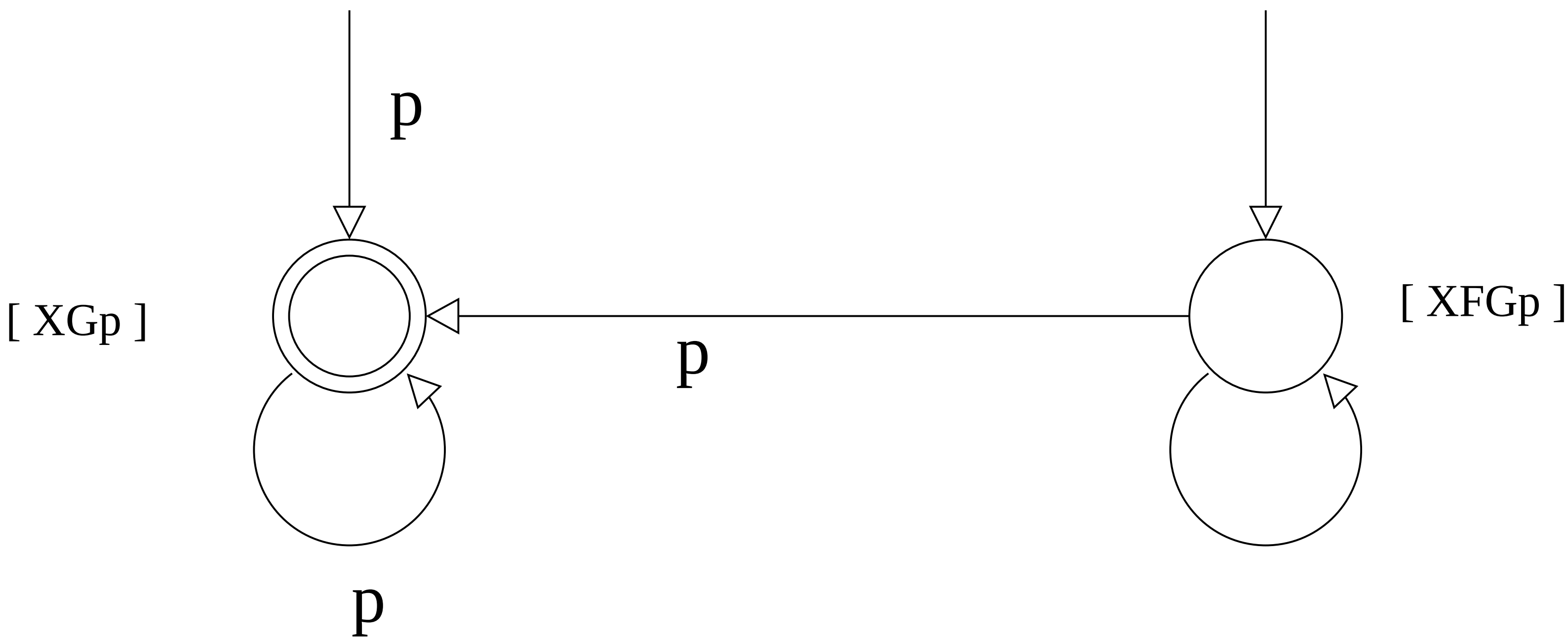
N.B.: e.g.,

"... $\xrightarrow{p}$ ..." here equivalent to ... $\xrightarrow{\{\{p,q\},\{p,\neg q\}\}}$ ...,

"... $\longrightarrow$ ..." here equivalent to ... $\xrightarrow{\{\{p,q\},\{p,\neg q\},\{\neg p,q\},\{\neg p,\neg q\}\}}$ ...

Example: $\mathbf{FG}p$

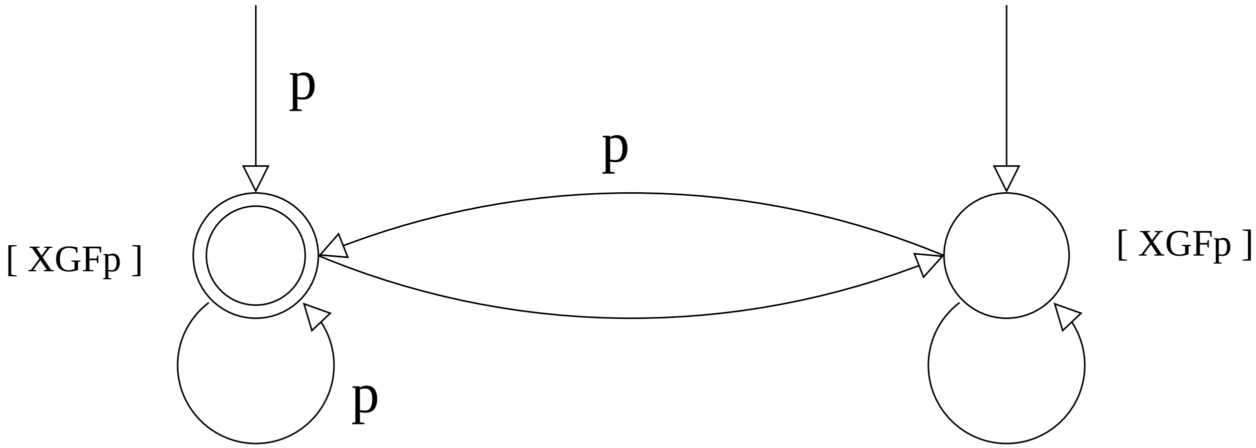
$$
\begin{aligned}
\varphi &= \mathbf{FG}p \\
&= \top\mathbf{U}(\bot\mathbf{R}p) \\
&= \bot\mathbf{R}p \vee \mathbf{X}\varphi \\
&= \underbrace{(p \wedge \mathbf{X}(\bot\mathbf{R}p))}_{\bot\mathbf{R}p} \vee \mathbf{X}\varphi
\end{aligned}
$$



Example: $\mathbf{G}F\,p$

$$\begin{aligned}
\varphi &= \mathbf{GF}p \\
&= \bot\mathbf{R}(\top\mathbf{U}p) \\
&= \top\mathbf{U}p \wedge \mathbf{X}\varphi \\
&= (p \vee \mathbf{X}(\mathbf{F}p)) \wedge \mathbf{X}\varphi \\
&= (p \wedge \mathbf{X}\varphi) \vee (\mathbf{X}\varphi \wedge \mathbf{XF}p) \\
&= (p \wedge \mathbf{X}\varphi) \vee \mathbf{X}(\varphi \wedge \mathbf{F}p) \\
&= (p \wedge \mathbf{X}\varphi) \vee \mathbf{X}\varphi
\end{aligned}$$

N.B.: $(\varphi \wedge \mathbf{F}p) = \varphi$



Content

Automata-Theoretic LTL Model Checking

Four steps:

1. Compute A_M
2. Compute A_φ
3. Compute the product $A_M \times A_\varphi$
4. Check the emptiness of $\mathcal{L}(A_M \times A_\varphi)$

Automata-Theoretic LTL Model Checking: complexity

Four steps:

1. Compute A_M: $|A_M| = O(|M|)$
2. Compute A_φ
3. Compute the product $A_M \times A_\varphi$
4. Check the emptiness of $\mathcal{L}(A_M \times A_\varphi)$

Automata-Theoretic LTL Model Checking: complexity [cont.]

Four steps:

1. Compute A_M: $|A_M| = O(|M|)$
2. Compute A_φ: $|A_\varphi| = O(2^{|\varphi|})$
3. Compute the product $A_M \times A_\varphi$
4. Check the emptiness of $\mathcal{L}(A_M \times A_\varphi)$

Automata-Theoretic LTL Model Checking: complexity [cont.]

Four steps:

1. Compute A_M: $|A_M| = O(|M|)$
2. Compute A_φ: $|A_\varphi| = O(2^{|\varphi|})$
3. Compute the product $A_M \times A_\varphi$:
   $|A_M \times A_\varphi| = |A_M| \cdot |A_\varphi| = O(|M| \cdot 2^{|\varphi|})$
4. Check the emptiness of $\mathcal{L}(A_M \times A_\varphi)$

Automata-Theoretic LTL Model Checking: complexity [cont.]

Four steps:

1. Compute A_M: $|A_M| = O(|M|)$
2. Compute A_φ: $|A_\varphi| = O(2^{|\varphi|})$
3. Compute the product $A_M \times A_\varphi$:
   $|A_M \times A_\varphi| = |A_M| \cdot |A_\varphi| = O(|M| \cdot 2^{|\varphi|})$
4. Check the emptiness of $\mathcal{L}(A_M \times A_\varphi)$:
   $O(|A_M \times A_\varphi|) = O(|M| \cdot 2^{|\varphi|})$

$\Longrightarrow$the complexity of LTL M.C. grows linearly wrt. the size of the model M and exponentially wrt. the size of the property φ

Final Remarks

- ▷ Büchi automata are in general more expressive than LTL!

$\Longrightarrow$ Some tools (e.g., Spin, ObjectGEODE) allow specifications to be expressed directly as NBA's

$\Longrightarrow$ complementation of NBA important!

- ▷ for every LTL formula, there are many possible equivalent NBA's

$\Longrightarrow$ lots of research for finding "the best" conversion algorithm

- ▷ performing the product and checking emptiness very relevant

$\Longrightarrow$ lots of techniques developed (e.g., partial order reduction)

$\Longrightarrow$ lots on ongoing research