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 ........................................... 2
• AUTOMATA ON FINITE WORDS ..................... 7
• AUTOMATA ON INFINITE WORDS ................... 25
• FROM Kripke STRUCTURES TO BüCHI AUT. .... 41
• FROM LTL FORMULAS TO BüCHI AUTOMATA .. 45
• AUTOMATA-THEORETIC LTL MODEL CHECKING 60
The problem

▷ Given a Kripke structure $M$ and an LTL specification $\psi$, does $M$ satisfy $\psi$?

\[ M \models \psi \]

▷ Equivalent to the CTL$^*$ M.C. problem:

\[ M \models \Box \psi \]

▷ Dual CTL$^*$ M.C. problem:

\[ M \models \Diamond \neg \psi \]
Automata-Theoretic LTL Model Checking

\[ M \models A\psi \quad \text{CTL}^* \]

\[ M \models \psi \quad \text{LTL} \]

\[ \mathcal{L}(M) \subseteq \mathcal{L}(\psi) \]

\[ \mathcal{L}(M) \cap \overline{\mathcal{L}(\psi)} = \{\} \]

\[ \mathcal{L}(A_M) \cap \mathcal{L}(A_{\neg\psi}) = \{\} \]

\[ \mathcal{L}(A_M \times A_{\neg\psi}) = \{\} \]

\( A_M \) is a B"uchi Automaton equivalent to \( M \) (which represents all and only the executions of \( M \))

\( A_{\neg\psi} \) is a B"uchi Automaton which represents all and only the paths that satisfy \( \neg\psi \) (do not satisfy \( \psi \))

\( A_M \times A_{\neg\psi} \) represents all and only the paths appearing in \( M \) and not in \( \psi \).
Automata-Theoretic LTL M.C. (dual version)

$\triangleright M \models E \phi$

$\iff M \not\models A \neg \phi$

$\iff \ldots$

$\iff \mathcal{L}(A_M \times A_{\phi}) \neq \{\}$

$\triangleright A_M$ is a Büchi Automaton equivalent to M (which represents all and only the executions of M)

$\triangleright A_{\phi}$ is a Büchi Automaton which represents all and only the paths that satisfy $\phi$

$\implies A_M \times A_{\phi}$ represents all and only the paths appearing in both $A_M$ and $A_{\phi}$. 
Automata-Theoretic LTL Model Checking

Four steps:

1. Compute $A_M$

2. Compute $A_\varphi$

3. Compute the product $A_M \times A_\varphi$

4. Check the emptiness of $\mathcal{L}(A_M \times A_\varphi)$
Content

- THE PROBLEM ...................................................... 2
- AUTOMATA ON FINITE WORDS ............................... 7
- AUTOMATA ON INFINITE WORDS ......................... 25
- FROM KRIPEK STRUCTURES TO BÜCHI AUT. ............ 41
- FROM LTL FORMULAS TO BÜCHI AUTOMATA .......... 45
- AUTOMATA-THEORETIC LTL MODEL CHECKING ......... 60
Finite Word Languages

- An Alphabet $\Sigma$ 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 $\Sigma^*$.
- 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.

\( \lambda \) 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 $A$
  \[ L(A) = \{ u \in \Sigma^* \mid A \text{ has an accepting run on } u \} \]

- 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 $b$.

![DFA A1 diagram]

NFA $A_2$. Recognizes words which end in $b$.

![NFA A2 diagram]
Determinisation

Theorem (determinisation) Given a NFA $A$ we can construct a DFA $A'$ s.t. $L(A) = L(A')$. 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 $\Sigma$ we can construct NFA $A$ over $\Sigma$ s.t.

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

- $Q := Q_1 \cup Q_2$, $I := I_1 \cup I_2$, $F := F_1 \cup F_2$

- $R(s, s') := \begin{cases} R_1(s, s') & \text{if } s \in Q_1 \\ R_2(s, s') & \text{if } s \in Q_2 \end{cases}$

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

- $L(A) = L(A_1) \cup L(A_2)$

- $|A| = |A_1| + |A_2|$
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

\[ Q = Q_1 \times Q_2, \quad I = I_1 \times I_2, \quad F = F_1 \times F_2. \]

\[ \langle p, q \rangle \xrightarrow{a} \langle p', q' \rangle \text{ iff } p \xrightarrow{a} p' \text{ and } q \xrightarrow{a} q'. \]

**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,

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

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

- $\langle p, q \rangle \xrightarrow{a} \langle p', q' \rangle$ if $a \in \Sigma_1 \cap \Sigma_2$ and $p \xrightarrow{a} p'$ and $q \xrightarrow{a} q'$.
- $\langle p, q \rangle \xrightarrow{a} \langle p', q \rangle$ if $a \in \Sigma_1$, $a \notin \Sigma_2$ and $p \xrightarrow{a} p'$.
- $\langle p, q \rangle \xrightarrow{a} \langle p, q' \rangle$ if $a \notin \Sigma_1$, $a \in \Sigma_2$ and $q \xrightarrow{a} q'$. 
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,

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

where

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

$\triangleright <p,q> \xrightarrow{a} <p',q>$ if $a \in \Sigma_1$ and $p \xrightarrow{a} p'$.

$\triangleright <p,q> \xrightarrow{a} <p,q'>$ if $a \in \Sigma_2$ and $q \xrightarrow{a} q'$. 
Decision Problems

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

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 $L(A_1) \subseteq L(A_2)$.

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

N.B. Model Checking:
Typically, $L(A_1 \times A_2 \times \ldots \times A_n) \subseteq L(A_{prop})$. 
Regular Expressions

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

Every regular expression $reg$ denotes a language $L(reg)$.

**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

- THE PROBLEM .................................................. 2
- AUTOMATA ON FINITE WORDS ............................ 7
- AUTOMATA ON INFINITE WORDS ....................... 25
- FROM K Ripke STRUCTURES TO BÜCHI AUT. ........ 41
- FROM LTL FORMULAS TO BÜCHI AUTOMATA ........ 45
- AUTOMATA-THEORETIC LTL MODEL CHECKING .......... 60
Infinite Word Languages

Modeling infinite computations of reactive systems.

- An $\omega$-word $\alpha$ over $\Sigma$ 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 $\Sigma^\omega$.

- A $\omega$-language $L$ is collection of $\omega$-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 $u^\omega = u.u.u.u\ldots$
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,

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

Nondeterministic Büchi Automaton

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

\( \triangleright \) A run \( \rho \) of \( A \) on omega word \( \alpha \) is an infinite sequence

\[ \rho = q_0, q_1, q_2, \ldots \text{ s.t. } q_0 \in I \text{ and } q_i \xrightarrow{a_i} q_{i+1} \text{ for } 0 \leq i. \]

\( \triangleright \) The run \( \rho \) is accepting if

\[ \text{Inf}(\rho) \cap F \neq \emptyset. \]

\( \triangleright \) 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.
Let a Nondeterministic Büchi Automaton (NBA) $A_2$ be

$$\begin{array}{c}
\text{a, b} \\
\text{-----------}
\end{array}$$

\[
\begin{array}{c}
s_1 \\
\downarrow b \\
\text{-----------} \\
\end{array} \quad \begin{array}{c}
s_2 \\
\downarrow b \\
\text{-----------} \\
\end{array}
\]

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


Deterministic vs. Nondeterministic Büchi Automata

Theorem *DBA*’s are strictly less powerful than *NBA*’s.
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

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

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

- $L(A) = L(A_1) \cup L(A_2)$
- $|A| = |A_1| + |A_2|$
- (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

$Q = Q_1 \times Q_2 \times \{1, 2\}$.
$I = I_1 \times I_2 \times \{1\}$.
$F = F_1 \times Q_2 \times \{1\}$.

\[
< p, q, 1 > \xrightarrow{a} < p', q', 1 > \text{ iff } p \xrightarrow{a} p' \text{ and } q \xrightarrow{a} q' \text{ and } p \notin F_1.
\]
\[
< p, q, 1 > \xrightarrow{a} < p', q', 2 > \text{ iff } p \xrightarrow{a} p' \text{ and } q \xrightarrow{a} q' \text{ and } p \in F_1.
\]
\[
< p, q, 2 > \xrightarrow{a} < p', q', 2 > \text{ iff } p \xrightarrow{a} p' \text{ and } q \xrightarrow{a} q' \text{ and } q \notin F_2.
\]
\[
< p, q, 2 > \xrightarrow{a} < p', q', 1 > \text{ iff } p \xrightarrow{a} p' \text{ and } q \xrightarrow{a} q' \text{ and } q \in F_2.
\]

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

\[\text{to visit infinitely often a state in } F \text{ (i.e., } F_1), \text{ it must visit infinitely often some state also in } F_2\]

- 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
Theorem (complementation)
For the NBA $A_1$ we can construct an NBA $A_2$ such that
\[ L(A_2) = \overline{L(A_1)}. \]
\[ |A_2| = O(2|A_1| \cdot \log(|A_1|)). \]

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 $\rho$ of $A$ is accepting if $\text{Inf}(\rho) \cap F_i \neq \emptyset$ 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 $\omega$-regular if it has the form $\bigcup_{i=1}^{n} U_i(V_i)^\omega$ where $U_i, V_i$ are regular languages.

Theorem A language $L$ is $\omega$-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

✓ • THE PROBLEM .............................................. 2
✓ • AUTOMATA ON FINITE WORDS .................. 7
✓ • AUTOMATA ON INFINITE WORDS ............. 25
⇒ • FROM Kripke STRUCTURES TO Büchi AUT. 41
• FROM LTL FORMULAS TO Büchi AUTOMATA . 45
• AUTOMATA-THEORETIC LTL MODEL CHECKING 60
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:
    \[
    \delta : \quad q \xrightarrow{a} q' \text{ iff } (q, q') \in R \text{ and } L(q') = a
    \]
    \[
    init \xrightarrow{a} q \text{ iff } q \in S_0 \text{ and } L(q') = a
    \]

- $L(A_M) = L(M)$
- $|A_M| = |M| + 1$
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. $M = \langle S, S_0, R, L, AP, FT \rangle$, $FT = \{F_1, \ldots, F_n\}$, into an NBA $A_M = \langle Q, \Sigma, \delta, I, F \rangle$ s.t.:
  - States: $Q := S \cup \{\text{init}\}$, init being a new initial state
  - Alphabet: $\Sigma := 2^{AP}$
  - Initial State: $I := \{\text{init}\}$
  - Accepting States: $F := FT$
  - Transitions:

$$\delta : \quad q \xrightarrow{a} q' \text{ iff } (a, a') \in R \text{ and } L(q') = a$$
$$\text{init} \xrightarrow{a} q \text{ iff } q \in S_0 \text{ and } L(q') = a$$

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

✓ • THE PROBLEM .................................................. 2
✓ • AUTOMATA ON FINITE WORDS ..................... 7
✓ • AUTOMATA ON INFINITE WORDS ................. 25
✓ • FROM Kripke STRUCTURES TO BüCHI Aut. .. 41
⇒ • FROM LTL FORMULAS TO BüCHI AUTOMATA . 45
● AUTOMATA-THEORETIC LTL MODEL CHECKING 60
Paths as $\omega$-words

Let $\varphi$ be an LTL formula.

- $\text{Var}(\varphi)$ denotes the set of free variables of $\varphi$.
  
  E.g. $\text{Var}(p \land (\neg q \mathcal{U} q)) = \{p, q\}$.

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

  $\implies$ a model for $\varphi$ is an $\omega$-word $\alpha = a_0, a_1, \ldots$ in $\Sigma^\omega$.

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

  Example A model of $p \land (\neg q \mathcal{U} q)$ is the $\omega$-word
  
  $\{p\}, \{\}, \{q\}, \{p, q\}^\omega$.

- N.B.: correspondence between $\omega$-words and paths in Kripke structures:

  $\alpha, i \models \varphi \iff \pi, s_i \models \varphi$, $\alpha, 0 \models \varphi \iff \pi, s_0 \models \varphi$.
Automata for LTL model checking

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

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

$\implies$ **Technique:** Construct a (Generalized) NBA $A_\phi$ such that $Mod(\phi) = \mathcal{L}(A_\phi)$. 
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 $\varphi_1 \lor \varphi_2 \in CL'(\varphi)$ then $\varphi_1, \varphi_2 \in CL'(\varphi)$.
- If $X \varphi_1 \in CL'(\varphi)$ then $\varphi_1 \in CL'(\varphi)$.
- If $(\varphi_1 U \varphi_2) \in CL'(\varphi)$ then $\varphi_1, \varphi_2 \in CL'(\varphi)$ and $X(\varphi_1 U \varphi_2) \in CL'(\varphi)$

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

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 $\varphi_1 \lor \varphi_2 \in CL(\varphi)$, we have $\varphi_1 \lor \varphi_2 \in A$ iff $\varphi_1 \in A$ or $\varphi_2 \in A$ (or both).
  - For all $(\varphi_1 \lor \varphi_2) \in CL(\varphi)$, we have $(\varphi_1 \lor \varphi_2) \in A$ iff $\varphi_2 \in A$ or $(\varphi_1 \in A$ and $X(\varphi_1 \lor \varphi_2) \in A)$.

- In practice, an atom is a consistent truth assignment to the elementary subformulas of $\varphi'$, $\varphi'$ being the result of applying the tableau expansion rules to $\varphi$.

- We call $Atoms(\varphi)$ the set of all atoms of $\varphi$. 
Definition of $A_\varphi$

For an LTL formula $\varphi$, we construct a Generalized NBA

$$A_\varphi = (Q, \Sigma, \delta, Q_0, FT)$$

as follows:

1. $\Sigma = 2^{\text{vars}(\varphi)}$
2. $Q = \text{Atoms}(\varphi)$, the set of atoms.
3. $\delta$ is s.t., for $q, q' \in \text{Atoms}(\varphi)$ and $a \in \Sigma$, $q \xrightarrow{a} q'$ holds in $\delta$ iff
   - $q' \cap \text{Var}(\varphi) = a$,
   - for all $X\varphi_1 \in \text{CL}(\varphi)$, we have $X\varphi_1 \in q$ iff $\varphi_1 \in q'$.
4. $Q_0 = \{q \in \text{Atoms}(\varphi) \mid \varphi \in q\}$.
5. $FT = (F_1, F_2, \ldots, F_k)$ where, for all $(\psi_i \mathbf{U} \varphi_i)$ occurring positively in $\varphi$,
   $$F_i = \{q \in \text{Atoms}(\varphi) \mid (\psi_i \mathbf{U} \varphi_i) \notin q \text{ or } \varphi_i \in q\}.$$
Definition of $A_\varphi$ [cont.]

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

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

**Size:** $|A_\varphi| = O(2^{|\varphi|})$. 
LTL Negative Normal Form (NNF)

- Every LTL formula $\varphi$ can be written as equivalent formula $\varphi'$ using only the operators $\neg$, $\lor$, $X$ and $U$.

- We can further push negations down to literal level:

  - $\neg(\varphi_1 \lor \varphi_2) \implies (\neg \varphi_1 \land \neg \varphi_2)$
  - $\neg(\varphi_1 \land \varphi_2) \implies (\neg \varphi_1 \lor \neg \varphi_2)$
  - $\neg X \varphi_1 \implies X \neg \varphi_1$
  - $\neg(\varphi_1 U \varphi_2) \implies (\neg \varphi_1 R \neg \varphi_2)$

  The resulting formula is expressed in terms of $\lor$, $\land$, $X$, $U$, $R$ and literals (Negative Normal Form, NNF).

- In the construction of $A_\varphi$ we now assume that $\varphi$ is in NNF.
Construction of $A_\varphi$ (Schema)

Apply recursively the following steps:

**Step 1:** Apply the tableau expansion rules to $\varphi$

$\psi_1 U \psi_2 \iff \psi_2 \lor (\psi_1 \land X(\psi_1 U \psi_2))$

$\psi_1 R \psi_2 \iff \psi_2 \land (\psi_1 \lor X(\psi_1 R \psi_2))$

until we get a boolean combination of elementary subformulas of $\varphi$
Step 2: Convert all formulas into Disjunctive Normal Form:

\[ \bigvee \left( \bigwedge l_{ij} \land \bigwedge X\psi_{ik} \right) \]

- Each disjunct \( \left( \bigwedge l_{ij} \land \bigwedge X\psi_{ik} \right) \) represents a state:
  - the conjunction of literals \( \bigwedge_{j} l_{ij} \) represents a set of labels in \( \Sigma \) (e.g., if \( Vars(\varphi) = \{p, q, r\} \), \( p \land \neg q \) represents the two labels \( \{p, \neg q, r\} \) and \( \{p, \neg q, \neg r\} \))
  - \( \bigwedge_{k} X\psi_{ik} \) represents the next part of the state (obligations for the successors)

- N.B., if no next part occurs, \( X\top \) is implicitly assumed
Construction of $A_\varphi$ (Schema) [cont.]

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

- 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, $X^T$ is implicitly assumed, so that an edge to a “true” node is drawn
Construction of $A_\varphi$ (Schema) [cont.]

**Step 4:** For every $\psi_i \cup \varphi_i$, for every state $q_j$, mark $q_j$ with $F_i$ iff $(\psi_i \cup \varphi_i) \notin q_j$ or $\varphi_i \in q_j$
Example: $pUq$

$\varphi = pUq$

$= q \lor (p \land X(pUq))$

$= (q \land X^\top) \lor (p \land X(pUq))$
Example: $FG\ p$

\[
\phi = FG\ p \\
= \top U (\perp R\ p) \\
= \perp R\ p \lor X\phi \\
= (p \land X (\perp R\ p)) \lor X\phi
\]
Example: $GF_p$

$\varphi = GF_p$

$= \perp_R(\top U_p)$

$= \top U_p \land X\varphi$

$= (p \lor X(F_p)) \land X\varphi$

$= (p \land X\varphi) \lor (X\varphi \land XF_p)$

$= (p \land X\varphi) \lor X(\varphi \land F_p)$

$= (p \land X\varphi) \lor X\varphi$  \hspace{1cm} N.B.: $(\varphi \land F_p) = \varphi$

\[ \text{[ XGFp ]} \]
<table>
<thead>
<tr>
<th>Content</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>√ ● THE PROBLEM</td>
<td>2</td>
</tr>
<tr>
<td>√ ● AUTOMATA ON FINITE WORDS</td>
<td>7</td>
</tr>
<tr>
<td>√ ● AUTOMATA ON INFINITE WORDS</td>
<td>25</td>
</tr>
<tr>
<td>√ ● FROM KRIPKE STRUCTURES TO BÜCHI AUT.</td>
<td>41</td>
</tr>
<tr>
<td>√ ● FROM LTL FORMULAS TO BÜCHI AUTOMATA</td>
<td>45</td>
</tr>
<tr>
<td>⇒ ● AUTOMATA-THEORETIC LTL MODEL CHECKING</td>
<td>60</td>
</tr>
</tbody>
</table>
Automata-Theoretic LTL Model Checking

Four steps:

1. Compute $A_M$
2. Compute $A_\varphi$
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

Four steps:

1. Compute $A_M$: $|A_M| = O(|M|)$
2. Compute $A_\phi$
3. Compute the product $A_M \times A_\phi$
4. Check the emptiness of $\mathcal{L}(A_M \times A_\phi)$
Automata-Theoretic LTL Model Checking: complexity [cont.]

Four steps:

1. Compute $A_M$: $|A_M| = O(|M|)$

2. Compute $A_\phi$: $|A_\phi| = O(2^{\phi})$

3. Compute the product $A_M \times A_\phi$

4. Check the emptiness of $\mathcal{L}(A_M \times A_\phi)$
Automata-Theoretic LTL Model Checking: complexity [cont.]

Four steps:

1. Compute $A_M$: $|A_M| = O(|M|)$

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

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

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 $L(A_M \times A_\varphi)$:
   
   $O(|A_M \times A_\varphi|) = O(|M| \cdot 2^{|\varphi|})$

$\iff$ the complexity of LTL M.C. grows linearly wrt. the size of the model $M$ and exponentially wrt. the size of the property $\varphi$
Final Remarks

- Büchi automata are in general more expressive than LTL!

- Some tools (e.g., Spin, ObjectGEODE) allow specifications to be expressed directly as NBA’s

- Complementation of NBA important!

- For every LTL formula, there are many possible equivalent NBA’s

- Lots of research for finding “the best” conversion algorithm

- Performing the product and checking emptiness very relevant

- Lots of techniques developed (e.g., partial order reduction)

- Lots on ongoing research