1. LTL: a specification language for LT properties
2. Büchi automata: automata on infinite words
3. LTL model checking
1 LTL: a specification language for LT properties

2 Büchi automata: automata on infinite words

3 LTL model checking
Linear time semantics: a reminder

TS $\mathcal{T}$ with state labels $AP = \{a, b\}$ (state and action names are omitted).
From now on, we assume no terminal state.

- Linear time semantics deals with traces of executions.
  - The language of infinite words described by $\mathcal{T}$.
  - E.g., do all executions eventually reach $\{b\}$? No.
Different kinds of LT properties

Safety

TS for semaphore-based mutex [BK08] (Ch. 2).

Ensure that $\langle c_1, c_2, y = \ldots \rangle \notin \text{Reach}(\mathcal{T}(PG_1 \parallel PG_2))$ or equivalently that $\not\exists \pi \in \text{Paths}(\mathcal{T}), \langle c_1, c_2, y = \ldots \rangle \in \pi$.

$\rightarrow$ Satisfied.
Different kinds of LT properties

Safety

TS for semaphore-based mutex [BK08] (Ch. 2).

For model checking, we like to use labels and traces.

- $AP = \{crit_1, crit_2\}$, natural labeling.
- Ensure that $\exists \sigma \in Traces(T), \{crit_1, crit_2\} \in \sigma$. 
Different kinds of LT properties

Liveness

Beverage vending machine [BK08] (Ch. 2).

Ensure that the machine delivers a drink infinitely often.

- \( AP = \{ paid, drink \} \), natural labeling.

- \( \forall \sigma \in Traces(\mathcal{T}) \), for all position \( i \) along \( \sigma \), label drink must appear in the future.

⇒ Will be formalized thanks to LTL.

← Satisfied. Recall we consider infinite executions.
Different kinds of LT properties

Liveness

Beverage vending machine \([BK08]\) (Ch. 2).

What if we ask that the machine delivers a \textit{beer} infinitely often.

\begin{itemize}
  \item[>] \( AP = \{\text{paid, soda, beer}\} \), natural labeling.
  \item[>] \( \forall \sigma \in \text{Traces}(\mathcal{T}) \), for all position \( i \) along \( \sigma \), label \textit{beer} must appear in the future.
\end{itemize}

\( \leftrightarrow \) \textbf{Not satisfied.} E.g., \( \sigma = (\emptyset \{\text{paid}\} \{\text{paid, soda}\})^\omega \).
Different kinds of LT properties

Safety vs. liveness

Informally, safety means “something bad never happens.”

⇒ Can easily be satisfied by doing nothing!

⇒ Needs to be complemented with liveness, i.e., “something good will happen.”

Finite vs. infinite time

Safety is violated by finite executions (i.e., the prefix up to seeing a bad state) whereas liveness is violated by infinite ones (witnessing that the good behavior never occurs).

⇒ For more about the safety/liveness taxonomy, see the book.
Different kinds of LT properties

Persistence

Ensure that a property \textit{eventually} holds \textit{forever}.

\begin{itemize}
    \item E.g., from some point on, \(a\) holds but \(b\) does not.
    \item Satisfied. Indeed,
    \[Traces(T) = \{a\} \left[ \{a\}^{\omega} \mid (\{a\} \{a, c\})^{\omega} \mid \{a\}^{+} \{b\} (\{a, c\} \{a\})^{\omega}\right].\]
    \item Ultimately periodic traces where \(b\) is false and \(a\) is true, at all steps after some point.
\end{itemize}
Different kinds of LT properties

Fairness (1/4)

TS for semaphore-based mutex [BK08] (Ch. 2).

Ensure that both processes get *fair access* to the critical section.

What is fairness?
Different kinds of LT properties

Fairness (2/4)

Different types of fairness constraints.

- **Unconditional fairness.** E.g., “every process gets access infinitely often.”

- **Strong fairness.** E.g., “every process that requests access infinitely often gets access infinitely often.”

- **Weak fairness.** E.g., “every process that continuously requests access from some point on gets access infinitely often.”

Unconditional $\Rightarrow$ strong $\Rightarrow$ weak.

Converse not true in general.

$\Rightarrow$ All forms can be formalized in LTL.
Different kinds of LT properties

Fairness (3/4)

The semaphore-based mutex is not fair in any sense. We have seen that starvation is possible. E.g., execution

\[ \langle n_1, n_2, y = 1 \rangle \rightarrow (\langle w_1, n_2, y = 1 \rangle \rightarrow \langle w_1, w_2, y = 1 \rangle \rightarrow \langle w_1, c_2, y = 0 \rangle)^\omega \]

sees process 1 asking continuously but never getting access (hence not even weakly fair).
Different kinds of LT properties

Fairness (4/4)

\[
\begin{align*}
(n_1, n_2, x = 2) & \quad \quad \quad \quad \quad \quad \quad \quad (n_1, n_2, x = 1) \\
(c_1, n_2, x = 2) & \quad \quad \quad \quad \quad \quad \quad \quad (n_1, c_2, x = 1) \\
(w_1, n_2, x = 2) & \quad \quad \quad \quad \quad \quad \quad \quad (n_1, w_2, x = 1) \\
(w_1, w_2, x = 1) & \quad \quad \quad \quad \quad \quad \quad \quad (w_1, w_2, x = 2) \\
(c_1, w_2, x = 1) & \quad \quad \quad \quad \quad \quad \quad \quad (w_1, c_2, x = 2)
\end{align*}
\]

*TS for Peterson’s mutex* [BK08] (Ch. 2).

Peterson’s mutex is **strongly fair**. We saw that it has bounded waiting.

▷ A process requesting access waits at most one turn.

→ **Infinitely frequent requests** → **infinitely frequent access.**

→ **Strong fairness.**
Linear Temporal Logic

Essentially, a set of acceptable traces over $AP$.

- Often difficult to describe explicitly.
- Adequate formalism needed for model checking.

$\Rightarrow$ Linear Temporal Logic (LTL):
propositional logic + temporal operators.
LTL in a nutshell

- **Atomic propositions** $a \in AP$ (represented as $\{a\}$, $\{b\}$, etc).

- **Boolean combinations of formulae:** $\neg \phi$, $\phi \land \psi$, $\phi \lor \psi$.

- **Temporal operators.**

<table>
<thead>
<tr>
<th>Atomic prop. $a$</th>
<th>next $\bigcirc \phi$</th>
<th>until $\phi U \psi$</th>
<th>eventually $\Diamond \phi$</th>
<th>always $\Box \phi$</th>
</tr>
</thead>
<tbody>
<tr>
<td>${a}$</td>
<td>arbitrary</td>
<td>arbitrary</td>
<td>arbitrary</td>
<td>arbitrary</td>
</tr>
<tr>
<td>arbitrary</td>
<td>$\phi$</td>
<td>arbitrary</td>
<td>arbitrary</td>
<td>arbitrary</td>
</tr>
<tr>
<td>$\phi \land \neg \psi$</td>
<td>$\phi \land \neg \psi$</td>
<td>$\psi$</td>
<td>arbitrary</td>
<td>arbitrary</td>
</tr>
<tr>
<td>$\neg \phi$</td>
<td>$\neg \phi$</td>
<td>$\neg \phi$</td>
<td>$\phi$</td>
<td>arbitrary</td>
</tr>
<tr>
<td>$\phi$</td>
<td>$\phi$</td>
<td>$\phi$</td>
<td>$\phi$</td>
<td>$\phi$</td>
</tr>
<tr>
<td>$\phi$</td>
<td>$\phi$</td>
<td>$\phi$</td>
<td>$\phi$</td>
<td>$\phi$</td>
</tr>
</tbody>
</table>
LTL syntax

Core syntax

Given the set of atomic propositions $AP$, LTL formulae are formed according to the following grammar:

$$
\phi ::= \text{true} \mid a \mid \phi \land \psi \mid \neg \phi \mid \Box \phi \mid \phi \mathcal{U} \psi
$$

where $a \in AP$.

⚠️ $\phi \mathcal{U} \psi$ requires that $\psi$ holds at some point! (i.e., $\phi$ forever does not suffice)
LTL syntax

Derived operators

$$\phi \lor \psi \equiv \neg(\neg\phi \land \neg\psi)$$

$$\phi \rightarrow \psi \equiv \neg\phi \lor \psi \quad \text{*implication*}$$

$$\phi \leftrightarrow \psi \equiv (\phi \rightarrow \psi) \land (\psi \rightarrow \phi) \quad \text{*equivalence*}$$

$$\phi \oplus \psi \equiv (\phi \land \neg\psi) \lor (\neg\phi \land \psi) \quad \text{*exclusive or*}$$

false $\equiv \neg\text{true}$

$$\Diamond \phi \equiv \text{true} U \phi \quad \text{*eventually (or finally)*}$$

$$\Box \phi \equiv \neg\Diamond\neg\phi \quad \text{*always (or globally)*}$$

$$\phi W \psi \equiv (\phi U \psi) \lor \Box \phi \quad \text{*weak until*}$$

$$\phi R \psi \equiv \neg(\neg\phi U \neg\psi) \quad \text{*release*}$$

▷ Weak until $\rightsquigarrow$ until that does not require $\psi$ to be reached.

▷ Release $\rightsquigarrow \psi$ must hold up to the point where $\phi$ releases it, or forever if $\phi$ never holds.
LTL syntax

Precedence order

Precedence order:

- unary operators before binary ones,
- ¬ and ⋄ equally strong,
- U before ∧, ∨ and →.
Formalizing LT properties in LTL

Safety

TS for semaphore-based mutex [BK08] (Ch. 2).

- $AP = \{\text{crit}_1, \text{crit}_2\}$, natural labeling.
- Ensure that $\not\exists \sigma \in \text{Traces}(T), \{\text{crit}_1, \text{crit}_2\} \in \sigma$.
  $\iff \neg \lozenge (\text{crit}_1 \land \text{crit}_2)$ or equivalently $\square (\neg \text{crit}_1 \lor \neg \text{crit}_2)$. 
Formalizing LT properties in LTL

Liveness

$\forall \sigma \in Traces(\mathcal{T})$, for all position $i$ along $\sigma$, label $\text{drink}$ must appear in the future.

$\neg \neg \Diamond \text{drink}$. 

$\Rightarrow$ “infinitely often”
Formalizing LT properties in LTL

Persistence

Ensure that a property eventually holds forever.

- E.g., from some point on, $a$ holds but $b$ does not.

$\diamond\Box (a \land \neg b)$.

$\implies$ “eventually always”
Formalizing LT properties in LTL

Fairness

Assume $k$ processes and $AP = \{wait_1, \ldots, wait_k, crit_1, \ldots, crit_k\}$.

- **Unconditional fairness.** E.g., “every process gets access infinitely often.”

  $\leftarrow \bigwedge_{1 \leq i \leq k} \Box \Diamond crit_i.$

- **Strong fairness.** E.g., “every process that requests access infinitely often gets access infinitely often.”

  $\leftarrow \bigwedge_{1 \leq i \leq k} (\Box \Diamond wait_i \rightarrow \Box \Diamond crit_i).$

- **Weak fairness.** E.g., “every process that continuously requests access from some point on gets access infinitely often.”

  $\leftarrow \bigwedge_{1 \leq i \leq k} (\Diamond \Box wait_i \rightarrow \Box \Diamond crit_i).$
LTL semantics

Over words (1/2)

Given propositions $AP$ and LTL formula $\phi$, the associated LT property is the language of words:

$$Words(\phi) = \{ \sigma = A_0A_1A_2\ldots \in (2^{AP})^\omega \mid \sigma \models \phi \}$$

where $\models$ is the smallest relation satisfying:

- $\sigma \models \text{true}$
- $\sigma \models a$ iff $a \in A_0$
- $\sigma \models \phi \land \psi$ iff $\sigma \models \phi$ and $\sigma \models \psi$
- $\sigma \models \neg \phi$ iff $\sigma \not\models \phi$
- $\sigma \models \Diamond \phi$ iff $\sigma[1..] = A_1A_2\ldots \models \phi$
- $\sigma \models \phi \cup \psi$ iff $\exists j \geq 0, \sigma[j..] \models \psi$ and $\forall 0 \leq i < j, \sigma[i..] \models \phi$
LTL semantics

Over words (2/2)

Other common operators:

\[
\begin{align*}
\sigma & \models \Diamond \phi & \text{iff} & \exists j \geq 0, \; \sigma[j..] \models \phi \\
\sigma & \models \Box \phi & \text{iff} & \forall j \geq 0, \; \sigma[j..] \models \phi \\
\sigma & \models \Box \Diamond \phi & \text{iff} & \forall j \geq 0, \; \exists i \geq j, \; \sigma[i..] \models \phi \\
\sigma & \models \Diamond \Box \phi & \text{iff} & \exists j \geq 0, \; \forall i \geq j, \; \sigma[i..] \models \phi
\end{align*}
\]
LTL semantics

Over transition systems

Let $\mathcal{T} = (S, \text{Act}, \rightarrow, I, \text{AP}, L)$ be a TS and $\phi$ an LTL formula over $\text{AP}$.

- For $\pi \in \text{Paths}(\mathcal{T})$, $\pi \models \phi$ iff $\text{trace}(\pi) \models \phi$.
- For $s \in S$, $s \models \phi$ iff $\forall \pi \in \text{Paths}(s), \pi \models \phi$.
- TS $\mathcal{T}$ satisfies $\phi$, denoted $\mathcal{T} \models \phi$ iff $\text{Traces}(\mathcal{T}) \subseteq \text{Words}(\phi)$.

It follows that $\mathcal{T} \models \phi$ iff $\forall s_0 \in I, s_0 \models \phi$. 
Example

Notice the added initial state.

\[
\begin{align*}
\mathcal{T} \not\models \Box a & \quad \mathcal{T} \models \Diamond \Box a & \quad \mathcal{T} \models \Box (a \land \neg c) \\
\mathcal{T} \not\models \Diamond b & \quad \mathcal{T} \not\models a \cup b & \quad \mathcal{T} \models \Box (c \rightarrow \Box a) \\
\mathcal{T} \models a \text{W} b & \quad \mathcal{T} \not\models b \text{R} a & \quad \mathcal{T} \models \Box \neg c \rightarrow \neg \Diamond b \\
\mathcal{T} \models \Box (b \rightarrow \Box \Diamond c) & \quad \mathcal{T} \models b \rightarrow \Box c & \quad \mathcal{T} \not\models \Box \Box (b \lor c) \lor \Box a
\end{align*}
\]

\[\implies \text{Blackboard solution.}\]
Semantics of negation

Paths

**Negation for paths**

For \( \pi \in Paths(\mathcal{T}) \) and an LTL formula \( \phi \) over \( AP \),

\[
\pi \not\models \phi \iff \pi \models \neg \phi
\]

because \( Words(\neg \phi) = (2^{AP})^\omega \setminus Words(\phi) \).
Semantics of negation

Transition systems

Negation for TSs

For TS $T = (S, Act, \rightarrow, I, AP, L)$ and an LTL formula $\phi$ over $AP$:

$$T \not\models \phi$$

$$\Downarrow \Uparrow$$

$$T \models \neg \phi$$

We have that $T \not\models \phi$ iff $\text{Traces}(T) \not\subseteq \text{Words}(\phi)$

iff $\text{Traces}(T) \setminus \text{Words}(\phi) \neq \emptyset$

iff $\text{Traces}(T) \cap \text{Words}(\neg \phi) \neq \emptyset$

But it may be the case that $T \not\models \phi$ and $T \not\models \neg \phi$ if

$$\text{Traces}(T) \cap \text{Words}(\neg \phi) \neq \emptyset \text{ and } \text{Traces}(T) \cap \text{Words}(\phi) \neq \emptyset.$$
Semantics of negation

Example

We saw that $\mathcal{T} \not\models \Diamond b$.

Do we have $\mathcal{T} \models \neg \Diamond b \equiv \Box \neg b$?

$$\implies \textbf{No.} \text{ Because trace } \sigma = \{a\}^2 \{b\}(\{a, c\}\{a\})^\omega \text{ satisfies } \Diamond b.$$
Equivalence of LTL formulae

Definition

\[ \text{LTL formulae } \phi \text{ and } \psi \text{ are equivalent, denoted } \phi \equiv \psi, \text{ if} \]

\[ \text{Words}(\phi) = \text{Words}(\psi). \]

\[ \implies \text{ Let us review some computational rules.} \]
Equivalence of LTL formulae

Duality, idempotence, absorption

■ Duality.

\[
\neg \Box \phi \equiv \Diamond \neg \phi \\
\neg \Diamond \phi \equiv \Box \neg \phi \\
\neg \bigcirc \phi \equiv \bigcirc \neg \phi
\]

■ Idempotence.

\[
\Box \Box \phi \equiv \Box \phi \\
\Diamond \Diamond \phi \equiv \Diamond \phi \\
\phi U (\phi U \psi) \equiv \phi U \psi \\
(\phi U \psi) U \psi \equiv \phi U \psi
\]

■ Absorption.

\[
\Diamond \Box \Diamond \phi \equiv \Box \Diamond \phi \\
\Box \Diamond \Box \phi \equiv \Diamond \Box \phi
\]
Equivalence of LTL formulae

Distribution

- **Distribution.**

\[
\begin{align*}
\Diamond (\phi \lor \psi) & \equiv \Diamond \phi \lor \Diamond \psi \\
\Box (\phi \land \psi) & \equiv \Box \phi \land \Box \psi \\
\Diamond (\phi \land \psi) & \equiv (\Diamond \phi) \lor (\Diamond \psi)
\end{align*}
\]

- **But...**

\[
\begin{align*}
\Diamond (\phi \land \psi) & \neq \Diamond \phi \land \Diamond \psi \\
\Box (\phi \lor \psi) & \neq \Box \phi \lor \Box \psi
\end{align*}
\]

\[
\begin{align*}
\mathcal{T} |\models \Diamond a \land \Diamond b & \quad \text{but} \quad \mathcal{T} \not|\models \Diamond (a \land b) \\
\mathcal{T} |\models \Box (a \lor b) & \quad \text{but} \quad \mathcal{T} \not|\models \Box a \lor \Box b
\end{align*}
\]
Equivalence of LTL formulae

Expansion laws

- Expansion laws (recursive equivalence).

  \( \phi \mathbf{U} \psi \equiv \psi \lor (\phi \land \Diamond (\phi \mathbf{U} \psi)) \)

  \( \Diamond \phi \equiv \phi \lor \Diamond \Diamond \phi \)

  \( \Box \phi \equiv \phi \land \Box \Box \phi \)

\(\implies\) Blackboard proof for until.
Positive normal form (PNF)

Weak-until PNF

Goal
Retain the full expressiveness of LTL but permit only negations of atomic propositions.

Weak-until PNF for LTL
Given atomic propositions $AP$, LTL formulae in weak-until positive normal form are given by:

$$\phi ::= \text{true} \mid \text{false} \mid a \mid \neg a \mid \phi \land \psi \mid \phi \lor \psi \mid \Box \phi \mid \phi U \psi \mid \phi W \psi$$

where $a \in AP$.

$\implies$ Gives a normal form for formulae.
Positive normal form (PNF)

Rewriting to weak-until PNF

To rewrite any LTL formula into weak-until PNF, we push negations inside:

\[
\begin{align*}
\neg \text{true} & \leadsto \text{false} & \neg \text{false} & \leadsto \text{true} \\
\neg \neg \phi & \leadsto \phi & \neg (\phi \land \psi) & \leadsto \neg \phi \lor \neg \psi \\
\neg \bigcirc \phi & \leadsto \bigcirc \neg \phi & \neg (\phi \lor \psi) & \leadsto \neg \phi \land \neg \psi \\
\neg \lozenge \phi & \leadsto \Box \neg \phi & \neg \Box \phi & \leadsto \lozenge \neg \phi \\
\neg (\phi \lor \psi) & \leadsto (\phi \land \neg \psi) \lor (\neg \phi \land \neg \psi) & \equiv \ (\phi \land \neg \psi) \lor (\neg \phi \land \neg \psi) \lor (\phi \land \neg \psi) \\
\neg (\phi \land \psi) & \leadsto (\phi \land \neg \psi) \lor (\neg \phi \land \neg \psi) \\
\end{align*}
\]

\[\iff \text{Blackboard example: } \neg \Box((a \lor b) \lor \bigcirc c).\]

\[\iff \text{Solution: } \lozenge((a \land \neg b) \lor (\neg a \land \neg b) \land \bigcirc \neg c).\]
Positive normal form (PNF)

Release PNF

Problem
Rewriting to weak-until PNF may induce an exponential blowup in the size of the formula (number of operators) because of the rewrite rule for until.

Solution: release PNF for LTL
Given atomic propositions \( AP \), LTL formulae in *release positive normal form* are given by:

\[
\phi ::= \text{true} \mid \text{false} \mid a \mid \neg a \mid \phi \land \psi \mid \phi \lor \psi \mid \lozenge \phi \mid \phi \mathbf{U} \psi \mid \phi \mathbf{R} \psi
\]

where \( a \in AP \).

We use the rule: \( \neg(\phi \mathbf{U} \psi) \sim \neg\phi \mathbf{R} \neg\psi \).

\[\implies \text{linear increase in the size of the formula.}\]
Back to fairness constraints

Reminder

Let $\phi, \psi$ be LTL formulae representing that “something is enabled” ($\phi$) and that “something is granted” ($\psi$). Recall the three types of fairness.

- **Unconditional** fairness constraint

  $$ufair = \square \diamond \psi.$$

- **Strong** fairness constraint

  $$sfair = \square \diamond \phi \rightarrow \square \diamond \psi.$$

- **Weak** fairness constraint

  $$wfair = \diamond \square \phi \rightarrow \square \diamond \psi.$$
Fairness assumptions

Let \( fair \) denote a conjunction of such assumptions. It is sometimes useful to check that all \textbf{fair executions} of a TS satisfy a formula (in contrast to all of them).

**Fair satisfaction**

Let \( \phi \) be an LTL formula and \( fair \) an LTL fairness assumption. We have that \( \mathcal{T} \models_{fair} \phi \) iff

\[
\forall \sigma \in \text{Traces}(\mathcal{T}) \text{ such that } \sigma \models fair, \sigma \models \phi.
\]
Example: randomized arbiter for mutex

**Mutual exclusion with a randomized arbiter** [BK08].

The arbiter chooses who gets access by tossing a coin: probabilities are abstracted by non-determinism.

Can process 1 access the section infinitely often?

→ **No,** \( T_1 \parallel Arbiter \parallel T_2 \not\models \lozenge \lozenge req_1 \rightarrow \square \lozenge crit_1 \) because the arbiter can always choose *tails.*
Example: randomized arbiter for mutex

Mutual exclusion with a randomized arbiter \([\text{BK08}]\).

Intuitively, this is \textit{unfair}: a real coin would lead to this with probability zero.

\[ \Rightarrow \] LTL fairness assumption: \(\Box \Diamond \text{heads} \land \Box \Diamond \text{tails} \).

\[ \iff \] The property is verified on fair executions, i.e.,
\[ \mathcal{T}_1 \parallel \text{Arbiter} \parallel \mathcal{T}_2 \models_{\text{fair}} \bigwedge_{i \in \{1,2\}} (\Box \Diamond \text{req}_i \rightarrow \Box \Diamond \text{crit}_i). \]
Handling fairness assumptions

Given a formula $\phi$ and a fairness assumption $\text{fair}$, we can reduce $\models_{\text{fair}}$ to the classical satisfaction $\models$.

From $\models_{\text{fair}}$ to $\models$

$$\mathcal{T} \models_{\text{fair}} \phi \iff \mathcal{T} \models (\text{fair} \rightarrow \phi).$$

$\implies$ The classical model checking algorithm will suffice.
1. LTL: a specification language for LT properties

2. Büchi automata: automata on infinite words

3. LTL model checking
Why?

Goal
Express languages of *infinite* words (e.g., $Words(\phi)$) using a *finite* automaton.

$\implies$ Will be essential to the model checking algorithm for LTL.
Finite-state automata

Reminder

Automata describing languages of \textit{finite} words.

\textbf{Definition: non-deterministic finite-state automaton (NFA)}

Tuple $\mathcal{A} = (Q, \Sigma, \delta, Q_0, F)$ with

- $Q$ a finite set of states,
- $\Sigma$ a finite alphabet,
- $\delta: Q \times \Sigma \rightarrow 2^Q$ a transition function,
- $Q_0 \subseteq Q$ a set of initial states,
- $F \subseteq Q$ a set of accept (or final) states.
Finite-state automata

Example

\[ Q = \{ q_1, q_2, q_3 \}, \Sigma = \{ A, B \}, \; Q_0 = \{ q_1 \}, \; F = \{ q_3 \}. \]

- This automaton is non-deterministic: see letter A on state \( q_1 \).

Language?

- Finite word \( \sigma = A_0 A_1 \ldots A_n \in \Sigma^* \). A run for \( \sigma \) is a sequence \( q_0 q_1 \ldots q_{n+1} \) such that \( q_0 \in Q_0 \) and for all \( 0 \leq i \leq n \), \( q_{i+1} \in \delta(q_i, A_i) \).

- \( \sigma \in L(A) \) if there exists a run \( q_0 q_1 \ldots q_{n+1} \) for \( \sigma \) such that \( q_{n+1} \in F \).

\( \hookrightarrow \) Here, \( L(A) = (A \mid B)^* A B \), i.e., all words ending by “AB.”
Finite-state automata

Regular expressions

Recall that NFAs correspond to regular languages, which can be described by regular expressions.

Syntax

Regular expressions over letters $A \in \Sigma$ are formed by

\[ E ::= \emptyset | \varepsilon | A | E + E' | E.E' | E^*. \]

Semantics

For regular expression $E$, language $L(E) \subseteq \Sigma^*$ obtained by

\[ L(\emptyset) = \emptyset, \quad L(\varepsilon) = \{\varepsilon\}, \quad L(A) = \{A\}, \quad L(E^*) = L(E)^*, \]
\[ L(E + E') = L(E) \cup L(E'), \quad L(E.E') = L(E).L(E'), \quad L(E.\emptyset) = \emptyset. \]

Syntactic sugar: we often write $E | E'$ for $E + E'$, $E^+$ for $E.E^*$ and we drop the concatenation operator, i.e., $EE'$ instead of $E.E'$. 

Finite-state automata

DFAs vs. NFAs

**Expressiveness**

Deterministic FAs (DFAs) are *expressively equivalent* to NFAs, i.e., for any NFA, there exists a DFA recognizing the same language.

⇒ One can determinize any NFA through subset construction.

⇒ With a potentially exponential blowup!

⇒ Blackboard illustration.
ω-regular languages

Definition

Intuitively, extension of regular languages to infinite words.

Syntax

An ω-regular expression $G$ over $Σ$ has the form

$$G = E_1.F_1^ω + \ldots + E_n.F_n^ω$$

for $n > 0$ where $E_i, F_i$ are regular expressions over $Σ$ with $ε \notin L(F_i)$.

Semantics

For $L \subseteq Σ^*$, let $L^ω = \{w_1w_2w_3\ldots | \forall i \geq 1, \ w_i \in L\}$.

For $G = E_1.F_1^ω + \ldots + E_n.F_n^ω$, $L^ω(G) \subseteq Σ^ω$ is given by

$$L^ω(G) = L(E_1).L(F_1)^ω \cup \ldots \cup L(E_n).L(F_n)^ω.$$
ω-regular languages

Examples

A language $L$ is ω-regular if $L = L_\omega(G)$ for some ω-regular expression $G$.

Examples for $\Sigma = \{A, B\}$.

- Words with infinitely many $A$’s: $(B^* A)^\omega$.
- Words with finitely many $A$’s: $(A \mid B)^* B^\omega$.
- Empty language: $\emptyset^\omega$ (OK because $\emptyset$ is a valid regular expression).

Properties of ω-regular languages

They are closed under union, intersection and complementation.
ω-regular languages

Counter-example

Not all languages on infinite words are ω-regular.

E.g., \( \mathcal{L} = \{ \text{words on } \Sigma = \{A, B\} \text{ such that } A \text{ appears infinitely often with increasingly many } B\text{'s between occurrences of } A \} \) is not.
Link with LTL?

We know that every LTL formula $\phi$ describes a language of infinite words $Words(\phi) \subseteq (2^{AP})^\omega$.

$\implies$ We will see that for every LTL formula $\phi$, $Words(\phi)$ is an $\omega$-regular language.

**The converse is false!**

There exist $\omega$-regular languages that cannot be expressed in LTL. E.g.,

$$L = \left\{ A_0 A_1 A_2 \ldots \in (2^\{a\})^\omega \mid \forall i \geq 0, \ a \in A_{2i} \right\},$$

the language of infinite words over $2^\{a\}$ where $a$ must hold in all even positions.

- $\omega$-regular expression $G = (\{a\} (\{a\} | \emptyset))^\omega$.
- Not expressible in LTL. Intuitively, LTL can count up to $k \in \mathbb{N}$ (e.g., words with at most $k$ occurrences of “a”) but not modulo $k$ (e.g., words with “a” every $k$ steps).
Büchi automata

Definition

Automata describing languages of infinite words.

- ω-regular languages.

Definition: non-deterministic Büchi automaton (NBA)

Tuple $A = (Q, \Sigma, \delta, Q_0, F)$ with

- $Q$ a finite set of states,
- $\Sigma$ a finite alphabet,
- $\delta: Q \times \Sigma \to 2^Q$ a transition function,
- $Q_0 \subseteq Q$ a set of initial states,
- $F \subseteq Q$ a set of accept (or final) states.

Same as before?
Büchi automata

Acceptance condition

\[ \Rightarrow \text{The automaton is identical, but the acceptance condition is different!} \]

**Run**

A run for an infinite word \( \sigma = A_0A_1 \ldots \in \Sigma^\omega \) is a sequence \( q_0q_1 \ldots \) of states such that \( q_0 \in Q_0 \) and for all \( i \geq 0 \), \( q_{i+1} \in \delta(q_i, A_i) \).

**Accepting run**

A run is accepting if \( q_i \in F \) for infinitely many indices \( i \in \mathbb{N} \).

**Accepted language of \( \mathcal{A} \)**

\[ \mathcal{L}_{\omega}(\mathcal{A}) = \{ \sigma \in \Sigma^\omega \mid \text{there is an accepting run for } \sigma \text{ in } \mathcal{A} \}. \]
Büchi automata

Examples

- Words with infinitely many A’s: \((B^* A)^\omega\).

Deterministic Büchi automaton (DBA).

- Words with finitely many A’s: \((A \mid B)^* B^\omega\).

Non-deterministic Büchi automaton (NBA).

Is there an equivalent DBA?

\[\Rightarrow\] We will see that no!

- Empty language: \(\emptyset^\omega\).
Büchi automata

Modeling an $\omega$-regular property

**Liveness property:** “once a request is provided, eventually a response shall occur.”

- $\{\text{req, resp}\} \subseteq AP$ for the TS.
- NBA $\mathcal{A}$ uses alphabet $2^{AP}$.
  - Succinct representation of multiple transitions using propositional logic. E.g., for $AP = \{a, b\}$, $q \xrightarrow{a\lor b} q'$ stands for $q \xrightarrow{a} q'$, $q \xrightarrow{b} q'$, and $q \xrightarrow{\{a,b\}} q'$. 

![Diagram](image-url)
**Theorem**

The class of languages accepted by NBAs agrees with the class of \(\omega\)-regular languages.

\[\implies\text{ For any } \omega\text{-regular property, we can build a corresponding NBA.}\]

\[\implies\text{ For any NBA } A, \text{ the language } L_\omega(A) \text{ is } \omega\text{-regular.}\]
From $\omega$-regular expressions to NBAs

Idea

Reminder

An $\omega$-regular expression $G$ over $\Sigma$ has the form

$$G = E_1.F_1^\omega + \ldots + E_n.F_n^\omega$$

for $n > 0$ where $E_i, F_i$ are regular expressions over $\Sigma$ with $\varepsilon \not\in \mathcal{L}(F_i)$.

Construction scheme

Use operators on NBAs mimicking operators on $\omega$-regular expressions:

- union of NBAs ($E_1.F_1^\omega + E_2.F_2^\omega$),
- $\omega$-operator for NFA ($F^\omega$),
- concatenation of an NFA and an NBA ($E.F^\omega$).
From $\omega$-regular expressions to NBAs

Union of NBAs (sketch)

Goal
Mimic $E_1.F_1^\omega + E_2.F_2^\omega$.

Let $A^1 = (Q^1, \Sigma, \delta^1, Q^1_0, F^1)$ and $A^2 = (Q^2, \Sigma, \delta^2, Q^2_0, F^2)$ be two NBAs over the same alphabet with disjoint state spaces.

Union
$A^1 + A^2 = (Q^1 \cup Q^2, \Sigma, \delta, Q^1_0 \cup Q^2_0, F^1 \cup F^2)$ with $\delta(q, A) = \delta^i(q, A)$ if $q \in Q^i$.

$\implies$ A word is accepted by $A^1 + A^2$ iff it is accepted by (at least) one of the automata.

$\implies L_\omega(A^1 + A^2) = L_\omega(A^1) \cup L_\omega(A^2)$. 
From $\omega$-regular expressions to NBAs

$\omega$-operator for NFA (sketch 1/2)

**Goal**

Mimic $F^{\omega}$.

Let $A = (Q, \Sigma, \delta, Q_0, F)$ be an NFA with $\epsilon \not\in \mathcal{L}(A)$.

Example: NFA accepting $A^*B$.

**Step 1.** If some initial states of $A$ have incoming transitions or $Q_0 \cap F \neq \emptyset$.

- Introduce new initial state $q_{new} \not\in F$.
- Add $q_{new} \xrightarrow{A} q$ iff $q_0 \xrightarrow{A} q$ for some $q_0 \in Q_0$.
- Keep all other transitions of $A$.
- New $Q_0 = \{q_{new}\}$. 

Graphical representation of the NFA and the modified NFA after applying the step.
From $\omega$-regular expressions to NBAs

$\omega$-operator for NFA (sketch 2/2)

**Step 2.** Build the NBA $A'$ as follows.

- If $q \xrightarrow{A} q' \in F$, then add $q \xrightarrow{A} q_0$ for all $q_0 \in Q_0$.
- Keep all other transitions of $A$.
- $Q'_0 = Q_0$ and $F' = Q_0$.

$\xrightarrow{\text{In practice, state } q_2 \text{ is now useless and can be removed.}}$

$\xrightarrow{\text{This NBA recognizes } (A^* B)\omega}$.
From $\omega$-regular expressions to NBAs

Concatenation of an NFA and an NBA (1/2)

**Goal**

Mimic $E . F^\omega$.

Let $A^1 = (Q^1, \Sigma, \delta^1, Q^1_0, F^1)$ be an NFA and $A^2 = (Q^2, \Sigma, \delta^2, Q^2_0, F^2)$ be an NBA, both over the same alphabet and with disjoint state spaces.

Example: NFA $A^1$ with $L(A^1) = (A \cdot B)^*$ and NBA $A^2$ with $L_\omega(A^2) = (A \mid B)^* B A^\omega$. 
From $\omega$-regular expressions to NBAs

Concatenation of an NFA and an NBA (2/2)

Construction of NBA $A = (Q = Q^1 \cup Q^2, \Sigma, \delta, Q_0, F = F^2)$.

- $Q_0 = \begin{cases} Q^1_0 & \text{if } Q^1_0 \cap F^1 = \emptyset \\ Q^1_0 \cup Q^2_0 & \text{otherwise} \end{cases}$

- $\delta(q, A) = \begin{cases} \delta^1(q, A) & \text{if } q \in Q^1 \text{ and } \delta^1(q, A) \cap F^1 = \emptyset \\ \delta^1(q, A) \cup Q^2_0 & \text{if } q \in Q^1 \text{ and } \delta^1(q, A) \cap F^1 \neq \emptyset \\ \delta^2(q, A) & \text{if } q \in Q^2 \end{cases}$

$\implies \mathcal{L}(A) = \mathcal{L}(A^1) \cdot \mathcal{L}(A^2)$, i.e., this NBA recognizes $(A B)^*(A | B)^* B A^\omega$. 
Checking non-emptiness

**Criterion for non-emptiness**

Let \( A \) be an NBA. Then,

\[
\mathcal{L}_\omega(A) \neq \emptyset
\]

\[
\iff \exists q_0 \in Q_0, \exists q \in F, \exists w \in \Sigma^*, \exists v \in \Sigma^+,\]
\[
q \in \delta^*(q_0, w) \land q \in \delta^*(q, v),
\]

i.e., there is reachable accept state on a cycle.

\[\implies\text{Can be checked in \textit{linear time} by computing reachable strongly connected components (SCCs).}\]

\[\implies\text{Important tool for LTL model checking.}\]
NBAs vs. DBAs

Recall that **DFAs are as expressive as NFAs**. What about DBAs w.r.t. NBAs?

**NBAs are strictly more expressive than DBAs**

There exists no DBA $A$ such that $L_\omega(A) = L_\omega((A \mid B)^* B^\omega)$.

\[
\begin{array}{c}
A, B \\
\downarrow \quad \downarrow \\
q_1 & \to & q_2 \\
B & \to \\
\end{array}
\]

*Words with finitely many A’s.*

→ See the book for the proof. Intuition: by contradiction, if a DBA would exist, then we show that it would accept some words with infinitely many A’s by exploiting determinism to construct corresponding accepting runs.
Is non-determinism really useful for model checking?

Yes. Consider a persistence property of the form “eventually forever”, i.e., LTL formula $\phi = \Diamond \square a$ for $AP = \{a\}$.

- $Words(\phi) = \mathcal{L}_\omega((\emptyset | \{a\})^* \{a\}^\omega)$.
- I.e., exactly $\mathcal{L}_\omega((A | B)^* B^\omega)$ for $A = \emptyset$ and $B = \{a\}$.

$q_1 \xrightarrow{a} q_2$

$\Rightarrow$ Not expressible with a DBA.
Generalized Büchi automata

- NBAs describe $\omega$-regular languages.
- Several equally expressive variants exist, with different acceptance conditions: Muller, Rabin, Streett, parity and generalized Büchi automata (GNBAs).

$\implies$ Will help us for LTL model checking.
Definition: non-det. generalized Büchi automaton (GNBA)

Tuple $G = (Q, \Sigma, \delta, Q_0, \mathcal{F})$ with

- $Q$ a finite set of states,
- $\Sigma$ a finite alphabet,
- $\delta: Q \times \Sigma \rightarrow 2^Q$ a transition function,
- $Q_0 \subseteq Q$ a set of initial states,
- $\mathcal{F} = \{F_1, \ldots, F_k\} \subseteq 2^Q$ ($k \geq 0$ and $\forall 0 \leq i \leq k$, $F_i \subseteq Q$).

**Intuition:** A GNBA requires to visits each set $F_i$ infinitely often.
Generalized Büchi automata

Acceptance condition

Accepting run

A run \( q_0q_1\ldots \) is accepting if for all \( F \in \mathcal{F} \), \( q_i \in F \) for infinitely many indices \( i \in \mathbb{N} \).

Accepted language of \( \mathcal{G} \)

\[
\mathcal{L}_\omega(\mathcal{G}) = \{ \sigma \in \Sigma^\omega \mid \text{there is an accepting run for } \sigma \text{ in } \mathcal{G} \}.
\]

For \( k = 0 \), all runs are accepting. For \( k = 1 \), \( \mathcal{G} \) is a simple NBA.

\( \triangle \) Observe the difference between \( F = \emptyset \) for an NBA (i.e., no run is accepting) and \( \mathcal{F} = \emptyset \) for a GNBA (i.e., all runs are accepting). In fact, \( \mathcal{F} = \emptyset \) is equivalent to having \( \mathcal{F} = \{ Q \} \).
Generalized Büchi automata
Modeling an $\omega$-regular property

**Liveness property:** “both processes are infinitely often in their critical section.”

$\{crit_1, crit_2\} \subseteq AP$ for the TS.

$\mathcal{F} = \{\{q_2\}, \{q_3\}\}$. *Both must be visited infinitely often!*
GNBAs vs. NBAs

From GNBA to NBA

For any GNBA $G$, there exists an equivalent NBA $A$ (i.e., $L_\omega(G) = L_\omega(A)$) of size $|A| = O(|G| \cdot |F|)$.

Construction scheme starting from $G$ with $F = \{F_1, \ldots, F_k\}$.

1. Make $k$ copies of $Q$ arranged in $k$ levels.
2. At level $i \in \{1, \ldots, k\}$, keep all transitions leaving states $q \not\in F_i$.
3. At level $i \in \{1, \ldots, k\}$, redirect transitions leaving states $q \in F_i$ to level $i + 1$ (level $k + 1 :=$ level 1).
4. $Q'_0 = \{\langle q_0, 1 \rangle \mid q_0 \in Q_0\}$, i.e., initial states in level 1; and $F' = \{\langle q, 1 \rangle \mid q \in F_1\}$, i.e., final states in level 1.

Works because by construction, $F'$ can only be visited infinitely often if the accept states ($F_i$) at every level $i$ are visited infinitely often.
GNBAs vs. NBAs

Example

\[
\begin{array}{c}
\text{true} \\
\text{crit}_1 \\
\text{true} \\
\end{array}
\quad
\begin{array}{c}
\text{true} \\
\text{crit}_2 \\
\text{true} \\
\end{array}
\]

\[
\begin{array}{c}
\langle q_1, 1 \rangle \\
\langle q_2, 1 \rangle \\
\langle q_3, 1 \rangle \\
\end{array}
\quad
\begin{array}{c}
\langle q_1, 2 \rangle \\
\langle q_2, 2 \rangle \\
\langle q_3, 2 \rangle \\
\end{array}
\]

\rightarrow \text{Blackboard illustration.}
1. LTL: a specification language for LT properties
2. Büchi automata: automata on infinite words
3. LTL model checking
Back to LTL model checking

Decision problem

**Definition: LTL model checking problem**
Given a TS \( \mathcal{T} \) and an LTL formula \( \phi \), decide if \( \mathcal{T} \models \phi \) or not.

\[ + \text{ if } \mathcal{T} \not\models \phi \text{ we would like a counter-example (trace witnessing it).} \]

\[ \implies \text{ Model checking algorithm via automata-based approach (Vardi and Wolper, 1986).} \]

**Intuition.**

- Represent \( \phi \) as an NBA.
- Use it to try to find a path \( \pi \) in \( \mathcal{T} \) such that \( \pi \not\models \phi \).
- If one is found, a prefix of it is an *error trace*. Otherwise, \( \mathcal{T} \models \phi \).
Back to LTL model checking

Key observation

\[ \mathcal{T} \models \phi \iff \text{Traces}(\mathcal{T}) \subseteq \text{Words}(\phi) \]
\[ \iff \text{Traces}(\mathcal{T}) \cap ((2^\mathcal{AP})^\omega \setminus \text{Words}(\phi)) = \emptyset \]
\[ \iff \text{Traces}(\mathcal{T}) \cap \text{Words}(\neg \phi) = \emptyset \]
\[ \iff \text{Traces}(\mathcal{T}) \cap \mathcal{L}_\omega(\mathcal{A}_{\neg \phi}) = \emptyset \]
\[ \iff \mathcal{T} \otimes \mathcal{A}_{\neg \phi} \models \lozenge \square \neg F \]

Line 3 uses negation for paths.

Line 4 uses the existence of an NBA for any \( \omega \)-regular language and the fact that all LTL formulae describe \( \omega \)-regular languages.

\[ \implies \text{We will see it in the following.} \]

Line 5 reduces the language intersection problem to the satisfaction of a persistence property over the product TS \( \mathcal{T} \otimes \mathcal{A}_{\neg \phi} \). The idea is to check that no trace yielded by \( \mathcal{T} \) will satisfy the acceptance condition of the NBA \( \mathcal{A}_{\neg \phi} \).
Overview of the automata-based approach for LTL model checking [BK08].
From LTL to GNBA

Examples

- NBA for $\square (req \rightarrow \Diamond resp)$.

- NBA for $\Diamond \square a$.

- GNBA for $\square \Diamond crit_1 \land \square \Diamond crit_2$.
From LTL to GNBA

Intuition of the construction (1/3)

**Goal**

For an LTL formula $\phi$, build GNBA $G_\phi$ over alphabet $2^{AP}$ such that $L_\omega(G_\phi) = \text{Words}(\phi)$.

- Assume $\phi$ only contains core operators $\land, \neg, \bigcirc, \bigwedge$ (w.l.o.g., see core syntax) and $\phi \neq \text{true}$ (otherwise, trivial GNBA).

- **What will be the states of $G_\phi$?**

  - Let $\sigma = A_0A_1A_2 \ldots \in \text{Words}(\phi)$. Idea: “expand” the sets $A_i \subseteq AP$ with subformulae $\psi$ of $\phi$.

  - Obtain $\overline{\sigma} = B_0B_1B_2 \ldots$ such that
    
    $$
    \psi \in B_i \iff A_i A_{i+1} A_{i+2} \ldots \models \psi.
    $$

  - $\overline{\sigma}$ will be a run for $\sigma$ in the GNBA $G_\phi$. 

---

Chapter 3: Linear temporal logic

Mickael Randour

74 / 102
From LTL to GNBA

Intuition of the construction (2/3)

- Let $\phi = a \cup (\neg a \land b)$ and $\sigma = \{a\} \{a, b\} \{b\} \ldots$
  - Letters $B_i$ are subsets of $\{a, \neg a, b, \neg a \land b, \phi\} \cup \{\neg b, \neg (\neg a \land b), \neg \phi\}$.
    - Letters $B_i$ are subsets of subformulae of $\phi$ and their negation.
  - Negations also considered for technical reasons.

- $A_0 = \{a\}$ is extended with $\neg b$, $\neg (\neg a \land b)$ and $\phi$ as they hold in $\sigma$ and no other subformula holds.

- $A_1 = \{a, b\}$ with $\neg (\neg a \land b)$ and $\phi$ as they hold in $\sigma[1..]$ and no others.

- $A_2 = \{b\}$ with $\neg a$, $\neg a \land b$ and $\phi$ as they hold in $\sigma[2..]$ and no others. Etc.

$\bar{\sigma} = \{a, \neg b, \neg (\neg a \land b), \phi\} \{a, b, \neg (\neg a \land b), \phi\} \{\neg a, b, \neg a \land b, \phi\} \ldots$

$\Rightarrow$ In practice, this is not done on words, but on the automaton.
From LTL to GNBA

Intuition of the construction (3/3)

- Sets $B_i$ will be the states of GNBA $G_\phi$.
- $\bar{\sigma} = B_0 B_1 B_2 \ldots$ is a run for $\sigma$ in $G_\phi$ by construction.
- Accepting condition chosen such that $\bar{\sigma}$ is accepting if and only if $\sigma \models \phi$.

- How do we encode the meaning of the logical operators?
  - \(\land, \neg\) and true impose consistent formula sets $B_i$ in the states (e.g., $a$ and $\neg a$ is not possible).
  - $\bigcirc$ encoded in the transition relation (must be consistent).
  - $\mathsf{U}$ split according to the expansion law into local condition (encoded in states) and next-step one (encoded in transitions).
  - Meaning of $\mathsf{U}$ is the least solution of the expansion law (see book) reflected in the choice of acceptance sets for $G_\phi$. 
From LTL to GNBA

Closure of a formula

**Definition: closure of φ**

Set $\text{closure}(φ)$ consisting of all sub-formulae $ψ$ of $φ$ and their negation $¬ψ$.

E.g., for $φ = a \cup (¬a \land b)$,

\[
\text{closure}(φ) = \{ a, ¬a, b, ¬b, ¬a \land b, ¬(¬a \land b), φ, ¬φ \}.
\]

\[|\text{closure}(φ)| = O(|φ|)\]

Sets $B_i$ are subsets of $\text{closure}(φ)$.

**But not all subsets are interesting!**

$\implies$ Restriction to **elementary sets**.

**Intuition:** a set $B$ is *elementary* if there is a path $π$ such that $B$ is the set of **all** formulae $ψ \in \text{closure}(φ)$ with $π \models ψ$. 
From LTL to GNBA

Elementary sets of formulae

Definition: elementary set

A set of sub-formulae $B \subseteq \text{closure}(\phi)$ is \textit{elementary} if:

1. $B$ is \textbf{logically consistent}, i.e., for all $\phi_1 \land \phi_2, \psi \in \text{closure}(\phi)$,
   - $\phi_1 \land \phi_2 \in B \iff \phi_1 \in B \land \phi_2 \in B$,
   - $\psi \in B \implies \neg \psi \notin B$,
   - $\text{true} \in \text{closure}(\phi) \implies \text{true} \in B$.

2. $B$ is \textbf{locally consistent}, i.e., for all $\phi_1 \mathbin{U} \phi_2 \in \text{closure}(\phi)$,
   - $\phi_2 \in B \implies \phi_1 \mathbin{U} \phi_2 \in B$,
   - $\phi_1 \mathbin{U} \phi_2 \in B \land \phi_2 \notin B \implies \phi_1 \in B$.

3. $B$ is \textbf{maximal}, i.e., for all $\psi \in \text{closure}(\phi)$,
   - $\psi \notin B \implies \neg \psi \in B$. 
From LTL to GNBA

Elementary sets: examples (1/2)

Let $\phi = a \cup (\neg a \land b)$:

$$\text{closure}(\phi) = \{a, \neg a, b, \neg b, \neg a \land b, \neg(\neg a \land b), \phi, \neg \phi\}.$$  

**Is $B = \{a, b, \phi\} \subset \text{closure}(\phi)$ elementary?**

$\hookrightarrow$ **No.** Logically and locally consistent but **not maximal** because $\neg a \land b \in \text{closure}(\phi)$, yet $\neg a \land b \notin B$ and $\neg(\neg a \land b) \notin B$.

**Is $B = \{a, b, \neg a \land b, \phi\} \subset \text{closure}(\phi)$ elementary?**

$\hookrightarrow$ **No.** It is **not logically consistent** because $a \in B$ and $\neg a \land b \in B$.

**Is $B = \{\neg a, \neg b, \neg(\neg a \land b), \phi\} \subset \text{closure}(\phi)$ elementary?**

$\hookrightarrow$ **No.** Logically consistent but **not locally consistent** because $\phi = a \cup (\neg a \land b) \in B$ and $\neg a \land b \notin B$ but $a \notin B$. 

From LTL to GNBA

Elementary sets: examples (2/2)

Let $\phi = a \lor (\neg a \land b)$:

$$\text{closure}(\phi) = \{a, \neg a, b, \neg b, \neg a \land b, \neg(\neg a \land b), \phi, \neg \phi\}.$$

All elementary sets?

$\Rightarrow$ Blackboard construction.

All elementary sets:

$$B_1 = \{a, b, \neg(\neg a \land b), \phi\},$$
$$B_2 = \{a, b, \neg(\neg a \land b), \neg \phi\},$$
$$B_3 = \{a, \neg b, \neg(\neg a \land b), \phi\},$$
$$B_4 = \{a, \neg b, \neg(\neg a \land b), \neg \phi\},$$
$$B_5 = \{\neg a, \neg b, \neg(\neg a \land b), \neg \phi\},$$
$$B_6 = \{\neg a, b, \neg a \land b, \phi\}.$$
From LTL to GNBA

Construction of $G_{\phi} (1/2)$

For formula $\phi$ over $AP$, let $G_{\phi} = (Q, \Sigma = 2^AP, \delta, Q_0, F)$ where:

- $Q = \{ B \subseteq \text{closure}(\phi) \mid B \text{ is elementary}\}$,
- $Q_0 = \{ B \in Q \mid \phi \in B\}$,
- $F = \{ F_{\phi_1 \cup \phi_2} \mid \phi_1 \cup \phi_2 \in \text{closure}(\phi)\}$ with
  
  $F_{\phi_1 \cup \phi_2} = \{ B \in Q \mid \phi_1 \cup \phi_2 \not\in B \lor \phi_2 \in B\}$.

**Intuition:** for any run $B_0 B_1 B_2 \ldots$, if $\phi_1 \cup \phi_2 \in B_0$, then $\phi_2$ must eventually become true ($\Rightarrow$ ensured by the acceptance condition).

Observe that $F = \emptyset$ if no until in $\phi$.

$\implies$ All runs are accepting in this case.
From LTL to GNBA

Construction of $G_{\phi}$ (2/2)

The transition relation $\delta: Q \times 2^{AP} \rightarrow 2^{Q}$ is given by:

- For $A \in 2^{AP}$ and $B \in Q$, if $A \neq B \cap AP$, then $\delta(B, A) = \emptyset$.

  *Intuition: transitions only exist for the set of propositions that are true in $B$, i.e., $B \cap AP$ is the only readable letter at state $B$."

- If $A = B \cap AP$, then $\delta(B, A)$ is the set of all elementary sets of formulae $B'$ satisfying
  
  (i) for every $\bigcirc \psi \in \text{closure}(\phi)$, $\bigcirc \psi \in B \iff \psi \in B'$, and

  (ii) for every $\phi_1 U \phi_2 \in \text{closure}(\phi)$,

  $\phi_1 U \phi_2 \in B \iff \left( \phi_2 \in B \lor (\phi_1 \in B \land \phi_1 U \phi_2 \in B') \right)$.

  *Intuition: (i) and (ii) reflect the semantics of $\bigcirc$ and $U$ operators, (ii) is based on the expansion law."
From LTL to GNBA

Example: $\phi = \Diamond a$

- $\text{closure}(\phi) = \{a, \neg a, \Diamond a, \neg \Diamond a\}$.

$\implies$ Blackboard construction of the GNBA + proof.

$Q = \{\{a, \Diamond a\}, \{a, \neg \Diamond a\}, \{\neg a, \Diamond a\}, \{\neg a, \neg \Diamond a\}\}$,

$Q_0 = \{\{a, \Diamond a\}, \{\neg a, \Diamond a\}\}$,

$F = \emptyset$. 
From LTL to GNBA

Example: $\phi = a \mathbin{U} b$ (1/3)

- $\text{closure}(\phi) = \{ a, \neg a, b, \neg b, a \mathbin{U} b, \neg (a \mathbin{U} b) \}$.

$\rightarrow$ Blackboard construction of the GNBA.
From LTL to GNBA

Example: $\phi = a U b$ (2/3)

Some explanations (see blackboard for more).

Let $B_1 = \{a, b, a U b\}$, $B_2 = \{\neg a, b, a U b\}$, $B_3 = \{a, \neg b, a U b\}$, $B_4 = \{\neg a, \neg b, \neg (a U b)\}$ and $B_5 = \{a, \neg b, \neg (a U b)\}$.

- $Q = \{B_1, B_2, B_3, B_4, B_5\}$, $Q_0 = \{B_1, B_2, B_3\}$.
- $\mathcal{F} = \{F_{a U b}\} = \{\{B_1, B_2, B_4, B_5\}\}$.
- $G_\phi$ is actually a simple NBA.

- Labels omitted for readability (recall label is $B \cap AP$).
- From $B_1$ (resp. $B_2$), we can go anywhere because $a U b$ is already fulfilled by $b \in B_1$ (resp. $B_2$).
- From $B_3$, we need to go where $a U b$ holds: $B_1$, $B_2$ or $B_3$.
- From $B_4$, we can go anywhere because $\neg (a U b)$ is already fulfilled by $\neg a, \neg b \in B_4$.
- From $B_5$, we need to go where $\neg (a U b)$ holds: $B_4$ or $B_5$. 
From LTL to GNBA

Example: $\phi = a U b$ (3/3)

Sample words/runs:

- $\sigma = \{a\} \{a\} \{b\}^\omega \in \text{Words}(\phi)$ has accepting run $\bar{\sigma} = B_3 B_3 B_2^\omega$ in $G_\phi$.

- $\sigma = \{a\}^\omega \not\in \text{Words}(\phi)$ has only one run $\bar{\sigma} = B_3^\omega$ in $G_\phi$ and it is not accepting since $B_3 \not\in F_{a U b}$. 
From LTL to... NBA

Construction

Idea: LTL ⇔ GNBA ⇔ NBA.

Theorem: LTL to NBA

For any LTL formula $\phi$ over propositions $AP$, there exists an NBA $A_\phi$ with $\text{Words}(\phi) = \mathcal{L}_\omega(A_\phi)$ which can be constructed in time and space $2^{O(|\phi|)}$.

Sketch

1. Construct the GNBA $G_\phi$.
   - $|\text{closure}(\phi)| = O(|\phi|)$ and $|Q| \leq 2 |\text{closure}(\phi)| = 2^{O(|\phi|)}$.
   - $\#$ accepting sets of $G_\phi = \#$ until-operators in $\phi \leq O(|\phi|)$.

2. Construct the NBA $A_\phi$.
   - $\#$ states of $A_\phi = |Q| \times \#$ accepting sets of $G_\phi$.
   - $\#$ states of $A_\phi$
     $\leq 2^{O(|\phi|)} \cdot O(|\phi|) = 2^{O(|\phi|)} \cdot 2^{\log(O(|\phi|))} = 2^{O(|\phi|)}$. 
The algorithm presented here is conceptually simple but may lead to unnecessary large GNBAs (and thus NBAs).

Example: the right NBA also recognizes $\Diamond a$ but is smaller.
From LTL to... NBA

Can we do better? (2/3)

Example: the right NBA also recognizes $a \cup b$ but is much smaller.

Can we always do better?
In practice, there exist more efficient (but more complex) algorithms in the literature.

Still, the exponential blowup cannot be avoided in the worst-case!

**Theorem: lower bound for NBA from LTL formula**

There exists a family of LTL formulae $\phi_n$ with $|\phi_n| = O(poly(n))$ such that every NBA $A_{\phi_n}$ for $\phi_n$ has at least $2^n$ states.

⇒ Proof in the next slides.
From LTL to... NBA

Lower bound proof (1/2)

Let \( AP \) be arbitrary and non-empty, i.e., \(|2^AP| \geq 2\). Let

\[
L_n = \left\{ A_1 \ldots A_n A_1 \ldots A_n \sigma \mid A_i \subseteq AP \land \sigma \in (2^AP)^\omega \right\}
\]

for \( n \geq 0 \).

This language is expressible in LTL, i.e., \( L_n = Words(\phi_n) \) for

\[
\phi_n = \bigwedge_{a \in AP} \bigwedge_{0 \leq i < n} (\Box^i a \leftrightarrow \Box^{n+i} a).
\]

Polynomial length: \( |\phi_n| = \mathcal{O}(|AP| \cdot n^2) \).

Claim: any NBA \( A \) with \( L_\omega(A) = L_n \) has at least \( 2^n \) states.
From LTL to... NBA

Lower bound proof (2/2)

Assume $A$ is such an automaton. Words $A_1 \ldots A_n A_1 \ldots A_n \emptyset^\omega$ belong to $L_n$, hence are accepted by $A$.

- For every word $A_1 \ldots A_n$ of length $n$, $A$ has a state $q(A_1 \ldots A_n)$ which can be reached after consuming $A_1 \ldots A_n$.
- From $q(A_1 \ldots A_n)$, it is possible to visit an accept state infinitely often by reading the suffix $A_1 \ldots A_n \emptyset^\omega$.
- If $A_1 \ldots A_n \neq A'_1 \ldots A'_n$, then $A_1 \ldots A_n A'_1 \ldots A'_n \emptyset^\omega \not\in L_n = L^\omega(A)$.

Therefore, states $q(A_1 \ldots A_n)$ are all pairwise different.

- Since each $A_i$ can take $2^{|AP|}$ different values, the number of different sequences $A_1 \ldots A_n$ of length $n$ is $(2^{|AP|})^n \geq 2^n$ (by non-emptiness of $AP$).

- Hence, the NBA has at least $2^n$ states.
LTL vs. NBAs

What have we learned?

Corollary
Every LTL formula expresses an $\omega$-regular property, i.e., for all LTL formula $\phi$, $\text{Words}(\phi)$ is an $\omega$-regular language.

Why? Because LTL can be transformed to NBA and NBAs coincide with $\omega$-regular languages.

The converse is false!
Recall $\mathcal{L} = \left\{ A_0A_1A_2 \ldots \in (2\{a\})^\omega \mid \forall i \geq 0, a \in A_{2i} \right\}$.

$\implies$ There are $\omega$-regular properties not expressible in LTL.
It remains to consider the last line.

Two remaining questions:

1. How to compute the product $T \otimes A_{\neg \phi}$?
2. How to check persistence, i.e., $T \otimes A_{\neg \phi} \models \Diamond \Box \neg F$?
Product of TS and NBA

Definition

**Definition: product of TS and NBA**

Let $\mathcal{T} = (S, Act, \rightarrow, I, AP, L)$ be a TS without terminal states and $\mathcal{A} = (Q, \Sigma = 2^{AP}, \delta, Q_0, F')$ a non-blocking NBA. Then, $\mathcal{T} \otimes \mathcal{A}$ is the following TS:

$$
\mathcal{T} \otimes \mathcal{A} = (S', Act, \rightarrow', I', AP', L')
$$

where

- $S' = S \times Q$, $AP' = Q$ and $L'(\langle s, q \rangle) = \{q\}$,
- $\rightarrow'$ is the smallest relation such that if $s \xrightarrow{\alpha} t$ and $q \xrightarrow{L(t)} p$, then $\langle s, q \rangle \xrightarrow{\alpha'} \langle t, p \rangle$,
- $I' = \{\langle s_0, q \rangle \mid s_0 \in I \land \exists q_0 \in Q_0, q_0 \xrightarrow{L(s_0)} q\}$. 

Chapter 3: Linear temporal logic

Mickael Randour
Product of TS and NBA

Example: simple traffic light

Simple traffic light with two modes: red and green. LTL formula to check $\phi = \Box \Diamond \neg green$.

TS $T$ for the traffic light.

NBA $A_{\neg \phi}$ for $\neg \phi = \Diamond \Box \neg green$.

$\implies$ Blackboard construction of $T \otimes A_{\neg \phi}$.
Persistence checking

Illustration (1/2)

It remains to check $\mathcal{T} \otimes \mathcal{A}_{\neg \phi} \models \diamond \Box \neg F$ to see that $\mathcal{T} \models \phi$.

Here, $\mathcal{T} \otimes \mathcal{A}_{\neg \phi} \models \diamond \Box \neg F$ with $F = \{q_2\}$.

Yes! State $\langle s_1, q_2 \rangle$ can be seen at most once, and state $\langle s_2, q_2 \rangle$ is not reachable.

$\Rightarrow$ There is no common trace between $\mathcal{T}$ and $\mathcal{A}_{\neg \phi}$.

$\Rightarrow \mathcal{T} \models \phi$. 
Persistence checking

Illustration (2/2)

*Slightly revised traffic light*: can switch off to save energy. Same formula $\phi$ (hence same NBA $A_{\neg\phi}$).

Here, $T \otimes A_{\neg\phi} \not\models \Diamond \Box \neg F$ with $F = \{q_2\}$. See for example path $\langle s_1, q_1 \rangle (\langle s_3, q_2 \rangle \langle s_1, q_2 \rangle)^\omega$ that visits $q_2$ infinitely often.

$\implies$ Path $\pi = (s_1s_3)^\omega$ of $T$ gives trace $\sigma = (\{\text{red}\} \emptyset)^\omega$ which is accepted by $A_{\neg\phi}$ (run $q_1(q_2)^\omega$), i.e., $\sigma \not\models \phi$. 
Persistence checking
Algorithm: cycle detection

As for checking non-emptiness, we reduce the problem to a cycle detection problem.

Persistence checking and cycle detection

Let \( \mathcal{T} \) be a TS without terminal states over \( AP \) and \( \Phi \) a propositional formula over \( AP \), then

\[
\mathcal{T} \not\vDash \Diamond \Box \Phi
\]

\[\iff\]

\[\exists s \in \text{Reach}(\mathcal{T}), \ s \not\vDash \Phi \text{ and } s \text{ is on a cycle in the graph of } \mathcal{T}.
\]

In particular, it holds for \( \Phi = \neg F \) as needed for LTL model checking (with \( F \) the acceptance set of the NBA \( A_{\neg \phi} \)).
Persistence checking

Algorithmic solutions for cycle detection

1. Compute the reachable SCCs and check if one contains a state satisfying $\neg \Phi$.
   \[ \leftrightarrow \text{Linear time but requires to construct entirely the product } \mathcal{T} \otimes \mathcal{A}_{\neg \phi} \text{ which may be very large (exponential).} \]

2. Another solution: on-the-fly algorithms.
   \[ \triangleright \text{Construct } \mathcal{T} \text{ and } \mathcal{A}_{\neg \phi} \text{ in parallel and simultaneously construct the reachable fragment of } \mathcal{T} \otimes \mathcal{A}_{\neg \phi} \text{ via nested depth-first search.} \]
   \[ \leftrightarrow \text{Construction of the product “on demand”.} \]
   \[ \leftrightarrow \text{More efficient in practice (used in software solutions such as Spin).} \]
   \[ \implies \text{See the book for more.} \]

Still, the complexity of LTL model checking remains high!
Wrap-up of the automata-based approach

\[ \mathcal{T} \models \phi \quad \text{iff} \quad \text{Traces}(\mathcal{T}) \subseteq \text{Words}(\phi) \]
\[ \text{iff} \quad \text{Traces}(\mathcal{T}) \cap ((2^\mathcal{AP})^\omega \setminus \text{Words}(\phi)) = \emptyset \]
\[ \text{iff} \quad \text{Traces}(\mathcal{T}) \cap \text{Words}(\neg \phi) = \emptyset \]
\[ \text{iff} \quad \text{Traces}(\mathcal{T}) \cap \mathcal{L}_\omega(\mathcal{A}_{\neg \phi}) = \emptyset \]
\[ \text{iff} \quad \mathcal{T} \otimes \mathcal{A}_{\neg \phi} \models \Diamond \Box \neg F \]

Complexity of this approach

The time and space complexity is \( \mathcal{O}(|\mathcal{T}|) \cdot 2^{\mathcal{O}(|\phi|)} \).
Complexity of LTL model checking

The LTL model checking problem is PSPACE-complete.

⇒ See the book for a proof by reduction from the membership problem for polynomial-space deterministic Turing machines.

Recall that bisimulation and simulation quotienting (Ch. 2) preserve LTL properties while being computable in polynomial time: interesting to do before model checking!
References I

C. Baier and J.-P. Katoen.  
Principles of model checking.  