Formal Methods for System Design

Chapter 3: Linear temporal logic

Mickael Randour

Mathematics Department, UMONS

September 2023

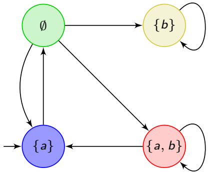
UMONS
Université de Mons

1 LTL: a specification language for LT properties

2 Büchi automata: automata on infinite words

3 LTL model checking

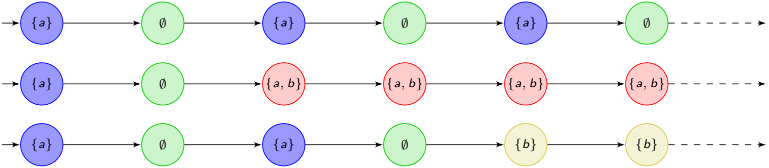
Linear time semantics: a reminder



TS $\mathcal{T}$ with state labels $AP = \{a, b\}$ (state and action names are omitted).

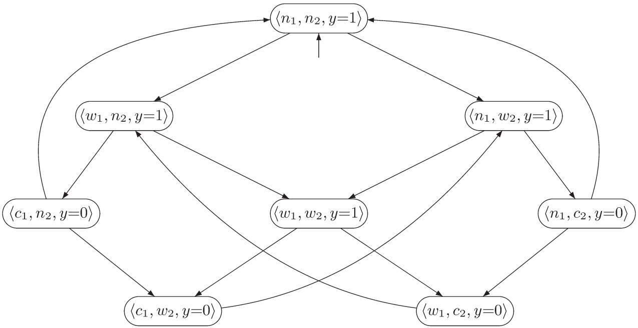
From now on, we assume **no terminal state**.

- **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



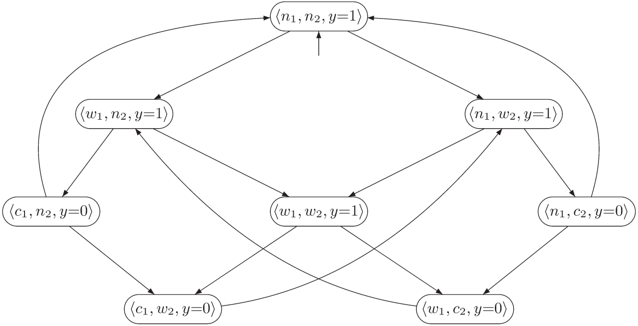
TS for semaphore-based mutex [BK08] (Ch. 2).

Ensure that $\langle c_1, c_2, y = \ldots\rangle \notin Reach(\mathcal{T}(PG_1 \parallel\!\mid PG_2))$ or equivalently that $\nexists\, \pi \in Paths(\mathcal{T}),\ \langle c_1, c_2, y = \ldots\rangle \in \pi$.

$\hookrightarrow$ **Satisfied.**

Different kinds of LT properties

Safety



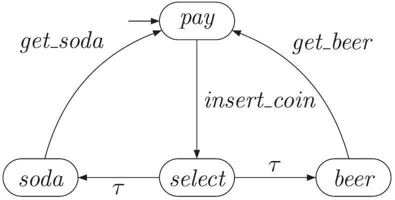
TS for semaphore-based mutex [BK08] (Ch. 2).

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

- $AP = \{crit_1, crit_2\}$, natural labeling.
- Ensure that $\nexists\, \sigma \in Traces(\mathcal{T}),\ \{crit_1, crit_2\} \in \sigma$.

Different kinds of LT properties

Liveness



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

Ensure that the machine delivers a *drink* infinitely often.

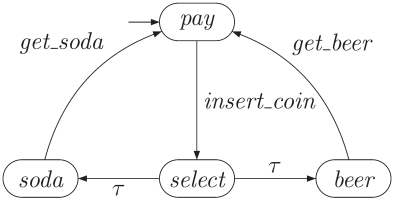
- ▷ $AP = \{paid, drink\}$, natural labeling.
- ▷ $\forall \sigma \in Traces(\mathcal{T})$, **for all position** *i* along σ, label *drink* **must appear in the future**.

$\Longrightarrow$ **Will be formalized thanks to LTL.**

$\hookrightarrow$ **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 *beer* infinitely often.

- ▷ $AP = \{paid, soda, beer\}$, natural labeling.
- ▷ $\forall\, \sigma \in Traces(\mathcal{T})$, for all position *i* along σ, label *beer* must appear in the future.

↪ **Not satisfied.** E.g., $\sigma = (\emptyset\, \{paid\}\, \{paid, soda\})^{\omega}$.

Different kinds of LT properties

Safety vs. liveness

Informally, safety means "something bad never happens."

$\Longrightarrow$ Can easily be satisfied by doing nothing!

$\Longrightarrow$ Needs to be complemented with liveness, i.e., "something good **will** happen."

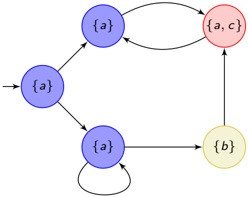
Finite vs. infinite time

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).

$\Longrightarrow$ **For more about the safety/liveness taxonomy, see the book.**

Different kinds of LT properties

Persistence



Ensure that a property **eventually** holds **forever**.

▷ E.g., from some point on, *a* holds but *b* does not.

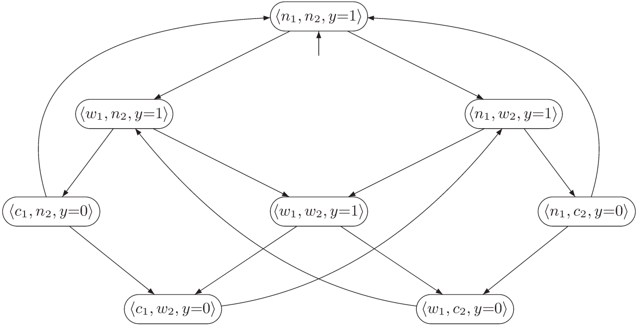
↪ **Satisfied.** Indeed,

$$Traces(\mathcal{T}) = \{a\} \left[\{a\}^\omega \mid (\{a\}\,\{a,c\})^\omega \mid \{a\}^+\,\{b\}\,(\{a,c\}\,\{a\})^\omega\right].$$

⟹ **Ultimately periodic traces where *b* is false and *a* is true, at all steps after some point.**

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.

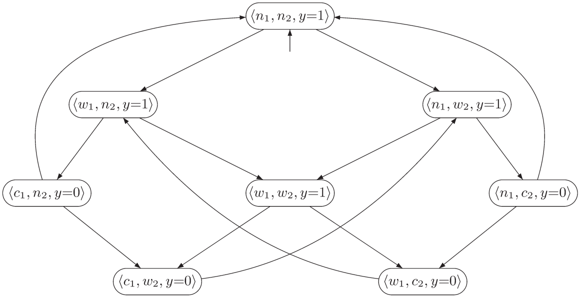
- **Unconditional fairness.** E.g., "every process gets access infinitely often."
- **Strong fairness.** E.g., "every process that requests access infinitely often gets access infinitely often."
- **Weak fairness.** E.g., "every process that continuously requests access from some point on gets access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

Unconditional $\Longrightarrow$ strong $\Longrightarrow$ weak.
Converse not true in general.

$\Longrightarrow$ **All forms can be formalized in LTL.**

Different kinds of LT properties

Fairness (3/4)



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

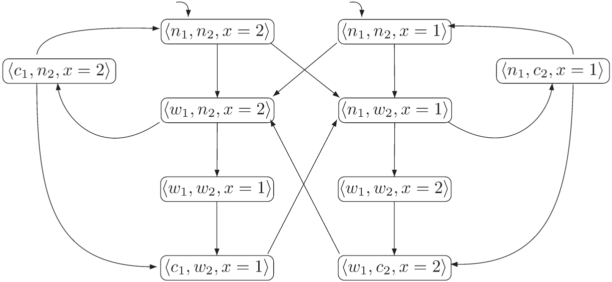
The semaphore-based mutex is **not fair** in any sense. We have seen that *starvation* is possible. E.g., execution

$$\langle n_1, n_2, y = 1\rangle \longrightarrow (\langle w_1, n_2, y = 1\rangle \longrightarrow \langle w_1, w_2, y = 1\rangle \longrightarrow \langle w_1, c_2, y = 0\rangle)^\omega$$

sees process 1 asking continuously but never getting access (hence not even weakly fair).

Different kinds of LT properties

Fairness (4/4)



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

LT property

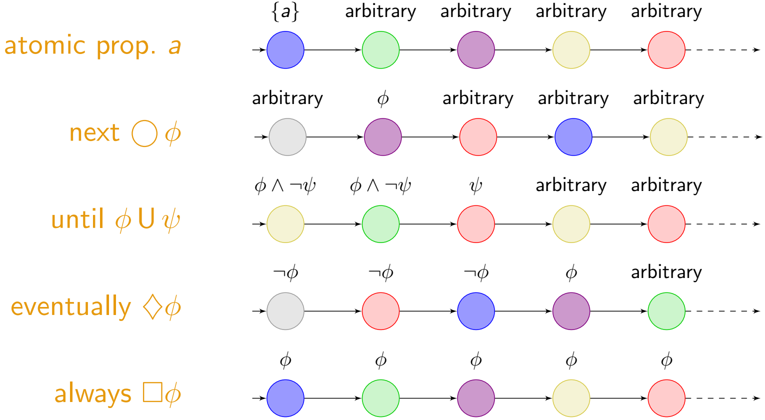
Essentially, a set of acceptable traces over AP.

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

$\Longrightarrow$ **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**: $\neg\phi$, $\phi \wedge \psi$, $\phi \vee \psi$.
- **Temporal operators**.



LTL syntax

Core syntax

LTL syntax

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

$$\phi ::= \mathsf{true} \mid a \mid \phi \wedge \psi \mid \neg\phi \mid \bigcirc \phi \mid \phi \, \mathsf{U} \, \psi$$

where $a \in AP$.

⚠ **$\phi \, \mathsf{U} \, \psi$ requires that ψ holds at some point! (i.e., ϕ forever does not suffice)**

LTL syntax

Derived operators

$$\phi \vee \psi \equiv \neg(\neg\phi \wedge \neg\psi)$$

$$\phi \rightarrow \psi \equiv \neg\phi \vee \psi$$ *implication*

$$\phi \leftrightarrow \psi \equiv (\phi \rightarrow \psi) \wedge (\psi \rightarrow \phi)$$ *equivalence*

$$\phi \oplus \psi \equiv (\phi \wedge \neg\psi) \vee (\neg\phi \wedge \psi)$$ *exclusive or*

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

$$\Diamond\phi \equiv \mathsf{true}\,\mathsf{U}\,\phi$$ ***eventually (or finally)***

$$\Box\phi \equiv \neg\Diamond\neg\phi$$ ***always (or globally)***

$$\phi\,\mathsf{W}\,\psi \equiv (\phi\,\mathsf{U}\,\psi) \vee \Box\phi$$ ***weak until***

$$\phi\,\mathsf{R}\,\psi \equiv \neg(\neg\phi\,\mathsf{U}\,\neg\psi)$$ ***release***

- Weak until $\rightsquigarrow$ until that does not require ψ to be reached.
- Release $\rightsquigarrow$ ψ must hold up to the point where ϕ releases it, or forever if ϕ never holds.

LTL syntax

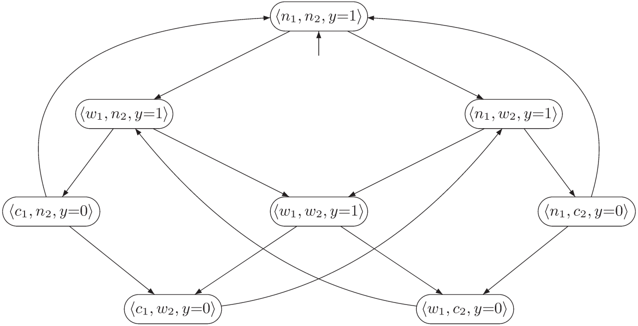
Precedence order

Precedence order:

- unary operators before binary ones,
- $\neg$ and $\bigcirc$ equally strong,
- U before $\wedge$, $\vee$ and $\rightarrow$.

Formalizing LT properties in LTL

Safety



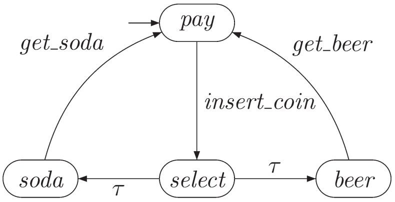
TS for semaphore-based mutex [BK08] (Ch. 2).

- ▷ $AP = \{crit_1, crit_2\}$, natural labeling.
- ▷ Ensure that $\nexists\, \sigma \in Traces(\mathcal{T}),\ \{crit_1, crit_2\} \in \sigma$.

$\hookrightarrow$ $\neg\Diamond\,(crit_1 \wedge crit_2)$ or equivalently $\Box\,(\neg crit_1 \vee \neg crit_2)$.

Formalizing LT properties in LTL

Liveness



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

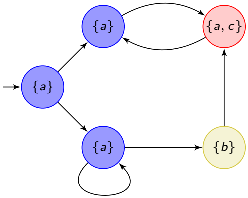
- ▷ $AP = \{paid, drink\}$, natural labeling.
- ▷ $\forall\, \sigma \in \mathit{Traces}(\mathcal{T})$, for all position *i* along σ, label *drink* must appear in the future.

↪ $\Box\Diamond drink$.

⟹ **"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.

$\hookrightarrow$ $\Diamond\Box(a \land \neg b)$.

$\Longrightarrow$ **"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."

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

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

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

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

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

LTL semantics

Over words (1/2)

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

$$Words(\phi) = \{\sigma = A_0 A_1 A_2 \ldots \in (2^{AP})^\omega \mid \sigma \models \phi\}$$

where $\models$ is the smallest relation satisfying:

$\sigma \models \text{true}$	*Recall letters are subsets of AP*	
$\sigma \models a$	iff	$a \in A_0$
$\sigma \models \phi \wedge \psi$	iff	$\sigma \models \phi$ and $\sigma \models \psi$
$\sigma \models \neg\phi$	iff	$\sigma \not\models \phi$
$\sigma \models \bigcirc \phi$	iff	$\sigma[1..] = A_1 A_2 \ldots \models \phi$
$\sigma \models \phi \, \mathsf{U} \, \psi$	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:

$$
\begin{aligned}
\sigma &\models \Diamond\phi && \text{iff} \quad \exists\, j \geq 0,\ \sigma[j..] \models \phi \\
\sigma &\models \Box\phi && \text{iff} \quad \forall\, j \geq 0,\ \sigma[j..] \models \phi \\
\sigma &\models \Box\Diamond\phi && \text{iff} \quad \forall\, j \geq 0,\ \exists\, i \geq j,\ \sigma[i..] \models \phi \\
\sigma &\models \Diamond\Box\phi && \text{iff} \quad \exists\, j \geq 0,\ \forall\, i \geq j,\ \sigma[i..] \models \phi
\end{aligned}
$$

LTL semantics

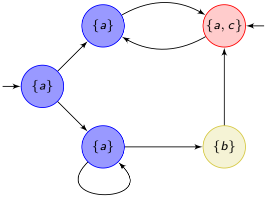
Over transition systems

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

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

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

Example



Notice the added initial state.

$\mathcal{T} \not\models \Box a$ $\quad$ $\mathcal{T} \models \Diamond\Box a$ $\quad$ $\mathcal{T} \models \bigcirc (a \land \neg c)$

$\mathcal{T} \not\models \Diamond b$ $\quad$ $\mathcal{T} \not\models a \,\mathsf{U}\, b$ $\quad$ $\mathcal{T} \models \Box(c \rightarrow \bigcirc a)$

$\mathcal{T} \models a \,\mathsf{W}\, b$ $\quad$ $\mathcal{T} \not\models b \,\mathsf{R}\, a$ $\quad$ $\mathcal{T} \models \Box\neg c \rightarrow \neg\Diamond b$

$\mathcal{T} \models \Box(b \rightarrow \Box\Diamond c)$ $\quad$ $\mathcal{T} \models b \rightarrow \Box c$ $\quad$ $\mathcal{T} \not\models \bigcirc \bigcirc (b \lor c) \lor \Box a$

$\Longrightarrow$ **Blackboard solution.**

Semantics of negation

Paths

Negation for paths

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

$$\pi \not\models \phi \Longleftrightarrow \pi \models \neg\phi$$

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

Semantics of negation

Transition systems

Negation for TSs

For TS $\mathcal{T} = (S, Act, \longrightarrow, I, AP, L)$ and an LTL formula ϕ over AP:

$$\mathcal{T} \not\models \phi$$
$$\not\Downarrow \; \Uparrow$$
$$\mathcal{T} \models \neg\phi$$

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

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

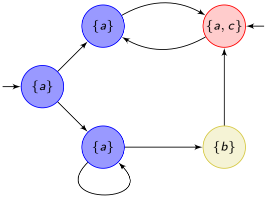
iff $Traces(\mathcal{T}) \cap Words(\neg\phi) \neq \emptyset$

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

$Traces(\mathcal{T}) \cap Words(\neg\phi) \neq \emptyset$ and $Traces(\mathcal{T}) \cap Words(\phi) \neq \emptyset$.

Semantics of negation

Example



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

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

$\Longrightarrow$ **No.** Because trace $\sigma = \{a\}^2\{b\}(\{a, c\}\{a\})^\omega$ satisfies $\Diamond b$.

Equivalence of LTL formulae

Definition

Equivalence of LTL formulae

LTL formulae ϕ and ψ are *equivalent*, denoted $\phi \equiv \psi$, if

$$Words(\phi) = Words(\psi).$$

$\Longrightarrow$ **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.**

$$\Box\Box\phi \equiv \Box\phi$$
$$\Diamond\Diamond\phi \equiv \Diamond\phi$$
$$\phi\,\mathsf{U}\,(\phi\,\mathsf{U}\,\psi) \equiv \phi\,\mathsf{U}\,\psi$$
$$(\phi\,\mathsf{U}\,\psi)\,\mathsf{U}\,\psi \equiv \phi\,\mathsf{U}\,\psi$$

- **Absorption.**

$$\Diamond\Box\Diamond\phi \equiv \Box\Diamond\phi$$
$$\Box\Diamond\Box\phi \equiv \Diamond\Box\phi$$

Equivalence of LTL formulae

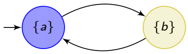
Distribution

- **Distribution.**

$$\bigcirc(\phi \,\mathsf{U}\, \psi) \equiv (\bigcirc\phi) \,\mathsf{U}\, (\bigcirc\psi)$$
$$\Diamond(\phi \vee \psi) \equiv \Diamond\phi \vee \Diamond\psi$$
$$\Box(\phi \wedge \psi) \equiv \Box\phi \wedge \Box\psi$$

- **But...**

$$\Diamond(\phi \wedge \psi) \not\equiv \Diamond\phi \wedge \Diamond\psi$$
$$\Box(\phi \vee \psi) \not\equiv \Box\phi \vee \Box\psi$$



$\mathcal{T} \models \Diamond a \wedge \Diamond b$ but $\mathcal{T} \not\models \Diamond(a \wedge b)$

$\mathcal{T} \models \Box(a \vee b)$ but $\mathcal{T} \not\models \Box a \vee \Box b$

Equivalence of LTL formulae

Expansion laws

- **Expansion laws (recursive equivalence).**

$$\begin{aligned}
\phi \,\mathsf{U}\, \psi &\equiv \psi \vee (\phi \wedge \bigcirc (\phi \,\mathsf{U}\, \psi)) \\
\Diamond \phi &\equiv \phi \vee \bigcirc \Diamond \phi \\
\Box \phi &\equiv \phi \wedge \bigcirc \Box \phi
\end{aligned}$$

$\Longrightarrow$ **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:

$$\phi ::= \mathsf{true} \mid \mathsf{false} \mid a \mid \neg a \mid \phi \wedge \psi \mid \phi \vee \psi \mid \bigcirc \phi \mid \phi \,\mathsf{U}\, \psi \mid \phi \,\mathsf{W}\, \psi$$

where $a \in AP$.

$\Longrightarrow$ **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:

$$\begin{array}{rclrcl}
\neg\text{true} & \rightsquigarrow & \text{false} & \neg\text{false} & \rightsquigarrow & \text{true} \\
\neg\neg\phi & \rightsquigarrow & \phi & \neg(\phi \wedge \psi) & \rightsquigarrow & \neg\phi \vee \neg\psi \\
\neg \bigcirc \phi & \rightsquigarrow & \bigcirc\neg\phi & \neg(\phi \vee \psi) & \rightsquigarrow & \neg\phi \wedge \neg\psi \\
\neg\Diamond\phi & \rightsquigarrow & \Box\neg\phi & \neg\Box\phi & \rightsquigarrow & \Diamond\neg\phi
\end{array}$$

$$\begin{array}{rcl}
\neg(\phi \,\mathsf{U}\, \psi) & \rightsquigarrow & (\phi \wedge \neg\psi) \,\mathsf{W}\, (\neg\phi \wedge \neg\psi) \\
 & \equiv & (\phi \wedge \neg\psi) \,\mathsf{U}\, (\neg\phi \wedge \neg\psi) \vee \Box(\phi \wedge \neg\psi) \\
\neg(\phi \,\mathsf{W}\, \psi) & \rightsquigarrow & (\phi \wedge \neg\psi) \,\mathsf{U}\, (\neg\phi \wedge \neg\psi)
\end{array}$$

$\Longrightarrow$ **Blackboard example:** $\neg\Box((a \,\mathsf{U}\, b) \vee \bigcirc c)$.

$\Longrightarrow$ **Solution:** $\Diamond((a \wedge \neg b) \,\mathsf{W}\, (\neg a \wedge \neg b) \wedge \bigcirc\neg c)$.

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:

$$\phi ::= \mathsf{true} \mid \mathsf{false} \mid a \mid \neg a \mid \phi \wedge \psi \mid \phi \vee \psi \mid \bigcirc \phi \mid \phi \,\mathsf{U}\, \psi \mid \phi \,\mathsf{R}\, \psi$$

where $a \in AP$.

We use the rule: $\neg(\phi \,\mathsf{U}\, \psi) \quad \rightsquigarrow \quad \neg\phi \,\mathsf{R}\, \neg\psi$.

$\Longrightarrow$ **linear increase in the size of the formula.**

Back to fairness constraints

Reminder

Let ϕ, ψ be LTL formulae representing that "something is enabled" (ϕ) and that "something is granted" (ψ). Recall the three types of fairness.

- *Unconditional* fairness constraint

$$\mathit{ufair} = \Box\Diamond\psi.$$

- *Strong* fairness constraint

$$\mathit{sfair} = \Box\Diamond\phi \rightarrow \Box\Diamond\psi.$$

- *Weak* fairness constraint

$$\mathit{wfair} = \Diamond\Box\phi \rightarrow \Box\Diamond\psi.$$

Fairness assumptions

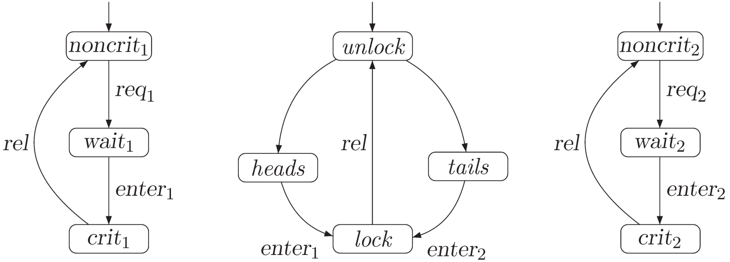
Let *fair* denote a conjunction of such assumptions. It is sometimes useful to check that all **fair executions** of a TS satisfy a formula (in contrast to all of them).

Fair satisfaction

Let ϕ be an LTL formula and *fair* an LTL fairness assumption. We have that $\mathcal{T} \models_{\mathit{fair}} \phi$ iff

$$\forall \sigma \in \mathit{Traces}(\mathcal{T}) \text{ such that } \sigma \models \mathit{fair},\ \sigma \models \phi.$$

Example: randomized arbiter for mutex



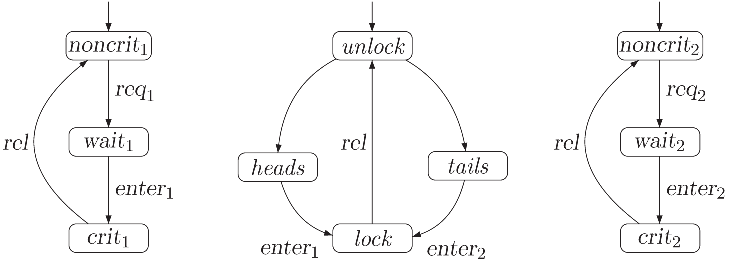
Mutual exclusion with a randomized arbiter [BK08].

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

Can process 1 access the section infinitely often?

$\hookrightarrow$ **No**, $\mathcal{T}_1 \mathbin{|||} \mathit{Arbiter} \mathbin{|||} \mathcal{T}_2 \not\models \Box\Diamond \mathit{req}_1 \rightarrow \Box\Diamond \mathit{crit}_1$ because the arbiter can always choose *tails*.

Example: randomized arbiter for mutex



Mutual exclusion with a randomized arbiter [BK08].

Intuitively, this is *unfair*: a real coin would lead to this with probability zero.

$\Longrightarrow$ LTL fairness assumption: $\square\diamondsuit heads \wedge \square\diamondsuit tails$.

$\hookrightarrow$ **The property is verified on fair executions**, i.e., $\mathcal{T}_1 \,|||\, Arbiter \,|||\, \mathcal{T}_2 \models_{fair} \bigwedge_{i\in\{1,2\}} (\square\diamondsuit req_i \rightarrow \square\diamondsuit crit_i)$.

Handling fairness assumptions

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

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

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

$\Longrightarrow$ **The classical model checking algorithm will suffice.**

Why?

Goal

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

$\Longrightarrow$ **Will be essential to the model checking algorithm for LTL.**

Finite-state automata

Reminder

Automata describing languages of *finite* words.

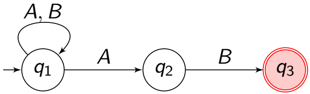
Definition: non-deterministic finite-state automaton (NFA)

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

- Q a finite set of states,
- Σ a finite alphabet,
- $\delta\colon Q \times \Sigma \rightarrow 2^Q$ 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 $\sigma = A_0 A_1 \dots A_n \in \Sigma^*$. A run for σ is a sequence $q_0 q_1 \dots q_{n+1}$ such that $q_0 \in Q_0$ and for all $0 \leq i \leq n$, $q_{i+1} \in \delta(q_i, A_i)$.
  - ▷ $\sigma \in \mathcal{L}(\mathcal{A})$ if there exists a run $q_0 q_1 \dots q_{n+1}$ for σ such that $q_{n+1} \in F$.

↪ Here, $\mathcal{L}(\mathcal{A}) = (A \mid B)^* A\, B$, 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

$$E ::= \emptyset \mid \varepsilon \mid A \mid E + E' \mid E.E' \mid E^*.$$

Semantics

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

$$\mathcal{L}(\emptyset) = \emptyset, \quad \mathcal{L}(\varepsilon) = \{\varepsilon\}, \quad \mathcal{L}(A) = \{A\}, \quad \mathcal{L}(E^*) = \mathcal{L}(E)^*,$$

$$\mathcal{L}(E + E') = \mathcal{L}(E) \cup \mathcal{L}(E'),\ \mathcal{L}(E.E') = \mathcal{L}(E).\mathcal{L}(E'),\ \mathcal{L}(E.\emptyset) = \emptyset.$$

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

Finite-state automata

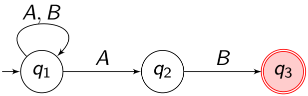
DFAs vs. NFAs

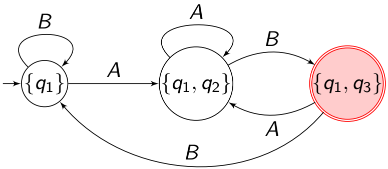
Expressiveness

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

$\implies$ **One can determinize any NFA through subset construction.**

$\implies$ **With a potentially exponential blowup!**





$\implies$ **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

$$G = E_1.F_1^\omega + \ldots + E_n.F_n^\omega \text{ for } n > 0$$

where E_i, F_i are regular expressions over Σ with $\varepsilon \notin \mathcal{L}(F_i)$.

Semantics

For $\mathcal{L} \subseteq \Sigma^*$, let $\mathcal{L}^\omega = \{w_1 w_2 w_3 \ldots \mid \forall\, i \geq 1,\ w_i \in \mathcal{L}\}$.

For $G = E_1.F_1^\omega + \ldots + E_n.F_n^\omega$, $\mathcal{L}_\omega(G) \subseteq \Sigma^\omega$ is given by

$$\mathcal{L}_\omega(G) = \mathcal{L}(E_1).\mathcal{L}(F_1)^\omega \cup \ldots \cup \mathcal{L}(E_n).\mathcal{L}(F_n)^\omega.$$

ω-regular languages

Examples

A language $\mathcal{L}$ is ω-regular if $\mathcal{L} = \mathcal{L}_\omega(G)$ 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., $\mathcal{L} = \big\{$words on $\Sigma = \{A, B\}$ such that A appears infinitely often with increasingly many B's between occurrences of $A\big\}$ is not.

Link with LTL?

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

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

The converse is false!

There exist ω-regular languages that cannot be expressed in LTL. E.g.,

$$\mathcal{L} = \left\{ A_0A_1A_2\ldots \in (2^{\{a\}})^\omega \mid \forall\, i \geq 0,\ a \in A_{2i} \right\},$$

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

- ω-regular expression $G = (\{a\}\,(\{a\} \mid \emptyset))^\omega$.
- 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 $\mathcal{A} = (Q, \Sigma, \delta, Q_0, F)$ with

- Q a finite set of states,
- Σ a finite alphabet,
- $\delta\colon Q \times \Sigma \to 2^Q$ 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

$\Longrightarrow$ The automaton is identical, but **the acceptance condition is different!**

Run

A run for an *infinite* word $\sigma = A_0 A_1 \ldots \in \Sigma^\omega$ is a sequence $q_0 q_1 \ldots$ 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 $i \in \mathbb{N}$.

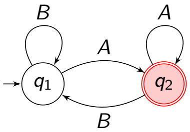
Accepted language of $\mathcal{A}$

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

Büchi automata

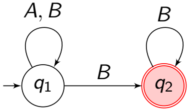
Examples

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



Deterministic Büchi automaton (DBA).

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

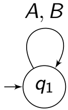


Non-deterministic Büchi automaton (NBA).

Is there an equivalent DBA?

$\Longrightarrow$ **We will see that there is not!**

- Empty language: $\emptyset^\omega$.



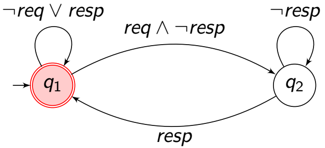
Büchi automata

Modeling an ω-regular property

Liveness property: "once a request is provided, eventually a response shall occur."

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

    $q \xrightarrow{a \vee b} q'$ stands for $q \xrightarrow{\{a\}} q'$, $q \xrightarrow{\{b\}} q'$, and $q \xrightarrow{\{a,b\}} q'$.



Büchi automata

NBAs and ω-regular languages

Theorem

The class of languages accepted by NBAs agrees with the class of ω-regular languages.

$\Longrightarrow$ **For any ω-regular property, we can build a corresponding NBA.**

$\Longrightarrow$ **For any NBA $\mathcal{A}$, the language $\mathcal{L}_\omega(\mathcal{A})$ is ω-regular.**

From ω-regular expressions to NBAs

Idea

Reminder

An ω-regular expression G over Σ has the form

$$G = E_1.F_1^\omega + \ldots + E_n.F_n^\omega \text{ for } n > 0$$

where E_i, F_i are regular expressions over Σ with $\varepsilon \notin \mathcal{L}(F_i)$.

Construction scheme

Use operators on NBAs mimicking operators on ω-regular expressions:

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

From ω-regular expressions to NBAs

Union of NBAs (sketch)

Goal

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

Let $\mathcal{A}^1 = (Q^1, \Sigma, \delta^1, Q_0^1, F^1)$ and $\mathcal{A}^2 = (Q^2, \Sigma, \delta^2, Q_0^2, F^2)$ be two NBAs over the same alphabet with disjoint state spaces.

Union

$\mathcal{A}^1 + \mathcal{A}^2 = (Q^1 \cup Q^2, \Sigma, \delta, Q_0^1 \cup Q_0^2, F^1 \cup F^2)$ with $\delta(q, A) = \delta^i(q, A)$ if $q \in Q^i$.

$\Longrightarrow$ **A word is accepted by $\mathcal{A}^1 + \mathcal{A}^2$ iff it is accepted by (at least) one of the automata.**

$$\Longrightarrow \mathcal{L}_\omega(\mathcal{A}^1 + \mathcal{A}^2) = \mathcal{L}_\omega(\mathcal{A}^1) \cup \mathcal{L}_\omega(\mathcal{A}^2).$$

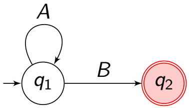
From ω-regular expressions to NBAs

ω-operator for NFA (sketch 1/2)

Goal

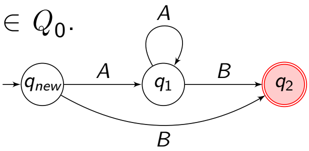
Mimic F^{ω}.

Let $\mathcal{A} = (Q, \Sigma, \delta, Q_0, F)$ be an NFA with $\varepsilon \notin \mathcal{L}(\mathcal{A})$.
Example: NFA accepting $A^* B$.



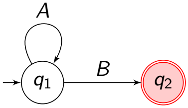
Step 1. If some initial states of $\mathcal{A}$ have incoming transitions or $Q_0 \cap F \neq \emptyset$.

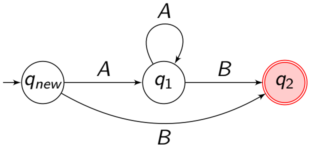
- Introduce new initial state $q_{new} \notin F$.
- Add $q_{new} \xrightarrow{A} q$ iff $q_0 \xrightarrow{A} q$ for some $q_0 \in Q_0$.
- Keep all other transitions of $\mathcal{A}$.
- New $Q_0 = \{q_{new}\}$.



From ω-regular expressions to NBAs

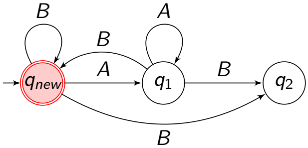
ω-operator for NFA (sketch 2/2)





Step 2. Build the NBA $\mathcal{A}'$ 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 $\mathcal{A}$.
- $Q'_0 = Q_0$ and $F' = Q_0$.



$\hookrightarrow$ *In practice, state q_2 is now useless and can be removed.*

$\implies \mathcal{L}_\omega(\mathcal{A}') = \mathcal{L}(\mathcal{A})^\omega$**, i.e., this NBA recognizes** $(A^*B)^\omega$**.**

From ω-regular expressions to NBAs

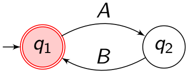
Concatenation of an NFA and an NBA (1/2)

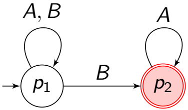
Goal

Mimic $E.F^\omega$.

Let $\mathcal{A}^1 = (Q^1, \Sigma, \delta^1, Q_0^1, F^1)$ be an NFA and $\mathcal{A}^2 = (Q^2, \Sigma, \delta^2, Q_0^2, F^2)$ be an NBA, both over the same alphabet and with disjoint state spaces.

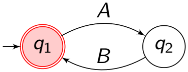
Example: NFA $\mathcal{A}^1$ with $\mathcal{L}(\mathcal{A}^1) = (A\,B)^*$ and NBA $\mathcal{A}^2$ with $\mathcal{L}_\omega(\mathcal{A}^2) = (A \mid B)^* B\, A^\omega$.

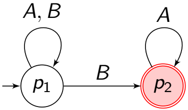




From ω-regular expressions to NBAs

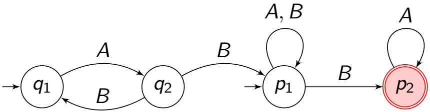
Concatenation of an NFA and an NBA (2/2)





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

- $Q_0 = \begin{cases} Q_0^1 & \text{if } Q_0^1 \cap F^1 = \emptyset \\ Q_0^1 \cup Q_0^2 & \text{otherwise} \end{cases}$
- $\delta(q, A) = \begin{cases} \delta^1(q, A) & \text{if } q \in Q^1 \text{ and } \delta^1(q, A) \cap F^1 = \emptyset \\ \delta^1(q, A) \cup Q_0^2 & \text{if } q \in Q^1 \text{ and } \delta^1(q, A) \cap F^1 \neq \emptyset \\ \delta^2(q, A) & \text{if } q \in Q^2 \end{cases}$



$\Longrightarrow \mathcal{L}_\omega(\mathcal{A}) = \mathcal{L}(\mathcal{A}^1).\mathcal{L}_\omega(\mathcal{A}^2)$, **i.e., this NBA recognizes** $(A\,B)^*(A \mid B)^*B\,A^\omega$**.**

Checking non-emptiness

Criterion for non-emptiness

Let $\mathcal{A}$ be an NBA. Then,

$$\mathcal{L}_\omega(\mathcal{A}) \neq \emptyset$$
$$\Updownarrow$$
$$\exists\, q_0 \in Q_0,\ \exists\, q \in F,\ \exists\, w \in \Sigma^*,\ \exists\, v \in \Sigma^+,$$
$$q \in \delta^*(q_0, w) \wedge q \in \delta^*(q, v),$$

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

$\Longrightarrow$ Can be checked in *linear time* by computing reachable strongly connected components (SCCs).

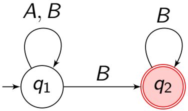
$\Longrightarrow$ **Important tool for LTL model checking.**

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 $\mathcal{A}$ such that $\mathcal{L}_\omega(\mathcal{A}) = \mathcal{L}_\omega((A \mid B)^* B^\omega)$.



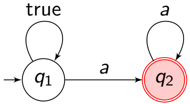
Words with finitely many A's.

$\Longrightarrow$ See the book for the proof. Intuition: by contradiction, if such a DBA existed, 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 $\phi = \Diamond\Box a$ for $AP = \{a\}$.

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



$\Longrightarrow$ **Not expressible with a DBA.**

Generalized Büchi automata

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

$\Longrightarrow$ **Will help us for LTL model checking.**

Generalized Büchi automata

Definition

Definition: non-det. generalized Büchi automaton (GNBA)

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

- Q a finite set of states,
- Σ a finite alphabet,
- $\delta \colon Q \times \Sigma \to 2^Q$ 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 visit **each set** F_i infinitely often.

Generalized Büchi automata

Acceptance condition

Accepting run

A run $q_0 q_1 \ldots$ 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.

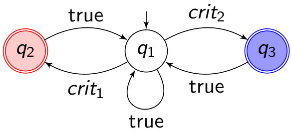
⚠ Observe the difference between $F = \emptyset$ for an NBA (i.e., no run is accepting) and $\mathcal{F} = \emptyset$ for a GNBA (i.e., all runs are accepting). In fact, $\mathcal{F} = \emptyset$ is equivalent to having $\mathcal{F} = \{Q\}$.

Generalized Büchi automata

Modeling an ω-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 $\mathcal{G}$, there exists an equivalent NBA $\mathcal{A}$ (i.e., $\mathcal{L}_\omega(\mathcal{G}) = \mathcal{L}_\omega(\mathcal{A})$) of size $|\mathcal{A}| = \mathcal{O}(|\mathcal{G}| \cdot |\mathcal{F}|)$.

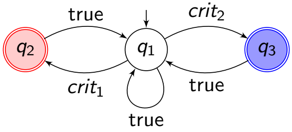
Construction scheme starting from $\mathcal{G}$ with $\mathcal{F} = \{F_1, \ldots, F_k\}$.

1. Make k copies of Q arranged in k levels.
2. At level $i \in \{1, \ldots, k\}$, keep all transitions leaving states $q \notin F_i$.
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.

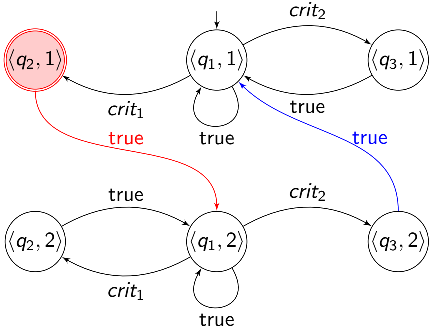
$\Longrightarrow$ **Works because by construction, F' can only be visited infinitely often if the accept states (F_i) at every level i are visited infinitely often.**

GNBAs vs. NBAs

Example



$\Longrightarrow$ **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 ϕ, decide if $\mathcal{T} \models \phi$ or not.

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

$\Longrightarrow$ Model checking algorithm via **automata-based approach** (Vardi and Wolper, 1986).

Intuition.

- Represent ϕ as an NBA.
- Use it to try to find a path π 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

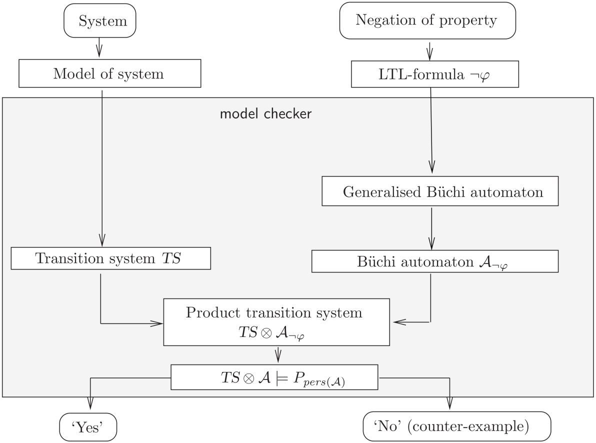
$$\begin{aligned}
\mathcal{T} \models \phi \quad & \text{iff} \quad Traces(\mathcal{T}) \subseteq Words(\phi) \\
& \text{iff} \quad Traces(\mathcal{T}) \cap ((2^{AP})^\omega \setminus Words(\phi)) = \emptyset \\
& \text{iff} \quad Traces(\mathcal{T}) \cap Words(\neg\phi) = \emptyset \\
& \text{iff} \quad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

Line 3 uses negation for paths.
Line 4 uses the existence of an NBA for any ω-regular language and the fact that **all LTL formulae describe ω-regular languages**.

$\Longrightarrow$ **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 algorithm

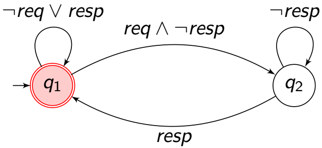


Overview of the automata-based approach for LTL model checking [BK08].

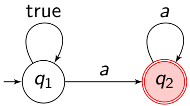
From LTL to GNBA

Examples

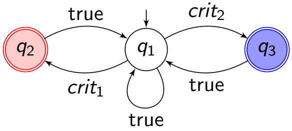
- NBA for $\square(req \rightarrow \lozenge resp)$.



- NBA for $\lozenge\square a$.



- GNBA for $\square\lozenge crit_1 \wedge \square\lozenge crit_2$.



From LTL to GNBA

Intuition of the construction (1/3)

Goal

For an LTL formula ϕ, build GNBA $\mathcal{G}_\phi$ over alphabet 2^{AP} such that $\mathcal{L}_\omega(\mathcal{G}_\phi) = Words(\phi)$.

- Assume ϕ only contains core operators $\wedge$, $\neg$, $\bigcirc$, U (w.l.o.g., see core syntax) and $\phi \neq$ true (otherwise, trivial GNBA).
- **What will be the states of $\mathcal{G}_\phi$?**
  - ▷ Let $\sigma = A_0A_1A_2\ldots \in Words(\phi)$. Idea: **"expand" the sets $A_i \subseteq AP$ with subformulae ψ of ϕ**.
  - ▷ Obtain $\overline{\sigma} = B_0B_1B_2\ldots$ such that

$$\psi \in B_i \iff A_iA_{i+1}A_{i+2}\ldots \models \psi.$$

  - ▷ $\overline{\sigma}$ will be a **run for σ** in the GNBA $\mathcal{G}_\phi$.

From LTL to GNBA

Intuition of the construction (2/3)

- Let $\phi = a \,\mathsf{U}\, (\neg a \wedge b)$ and $\sigma = \{a\}\,\{a, b\}\,\{b\}\dots$
  - ▷ Letters B_i are subsets of
    $$\underbrace{\{a, \neg a, b, \neg a \wedge b, \phi\}}_{\text{subformulae of } \phi} \cup \underbrace{\{\neg b, \neg(\neg a \wedge b), \neg\phi\}}_{\text{their negation}}.$$
  - ▷ Negations also considered for technical reasons.
- $A_0 = \{a\}$ is extended with $\neg b$, $\neg(\neg a \wedge b)$ and ϕ as they hold in σ and no other subformula holds.
- $A_1 = \{a, b\}$ with $\neg(\neg a \wedge b)$ and ϕ as they hold in $\sigma[1..]$ and no others.
- $A_2 = \{b\}$ with $\neg a$, $\neg a \wedge b$ and ϕ as they hold in $\sigma[2..]$ and no others. Etc.

$$\overline{\sigma} = \underbrace{\{a, \neg b, \neg(\neg a \wedge b), \phi\}}_{B_0}\,\underbrace{\{a, b, \neg(\neg a \wedge b), \phi\}}_{B_1}\,\underbrace{\{\neg a, b, \neg a \wedge b, \phi\}}_{B_2}\dots$$

$\Longrightarrow$ **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 $\mathcal{G}_\phi$.
- $\overline{\sigma} = B_0 B_1 B_2 \ldots$ is a run for σ in $\mathcal{G}_\phi$ by construction.
- Accepting condition chosen such that $\overline{\sigma}$ is accepting if and only if $\sigma \models \phi$.
- **How do we encode the meaning of the logical operators?**
  - $\wedge$, $\neg$ 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)*.
  - U split according to the *expansion law* into *local condition (encoded in states)* and *next-step one (encoded in transitions)*.
  - Meaning of U is the *least solution* of the expansion law (see book) $\Longrightarrow$ reflected in the choice of *acceptance sets for $\mathcal{G}_\phi$*.

From LTL to GNBA

Closure of a formula

Definition: closure of ϕ

Set $closure(\phi)$ consisting of all sub-formulae ψ of ϕ and their negation $\neg\psi$.

E.g., for $\phi = a \, \mathsf{U} \, (\neg a \wedge b)$,

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

$\hookrightarrow$ $|closure(\phi)| = \mathcal{O}(|\phi|)$.

Sets B_i are subsets of $closure(\phi)$.

But not all subsets are interesting!

$\Longrightarrow$ 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 $\psi \in closure(\phi)$ with $\pi \models \psi$.

From LTL to GNBA

Elementary sets of formulae

Definition: elementary set

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

1. B is **logically consistent**, i.e., for all $\phi_1 \wedge \phi_2, \psi \in \mathit{closure}(\phi)$,
   - $\phi_1 \wedge \phi_2 \in B \iff \phi_1 \in B \;\wedge\; \phi_2 \in B$,
   - $\psi \in B \implies \neg\psi \notin B$,
   - $\mathsf{true} \in \mathit{closure}(\phi) \implies \mathsf{true} \in B$.
2. B is **locally consistent**, i.e., for all $\phi_1 \,\mathsf{U}\, \phi_2 \in \mathit{closure}(\phi)$,
   - $\phi_2 \in B \implies \phi_1 \,\mathsf{U}\, \phi_2 \in B$,
   - $\phi_1 \,\mathsf{U}\, \phi_2 \in B \;\wedge\; \phi_2 \notin B \implies \phi_1 \in B$.
3. B is **maximal**, i.e., for all $\psi \in \mathit{closure}(\phi)$,
   - $\psi \notin B \implies \neg\psi \in B$.

From LTL to GNBA

Elementary sets: examples (1/2)

Let $\phi = a \,\mathsf{U}\, (\neg a \wedge b)$:

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

- Is $B = \{a, b, \phi\} \subset closure(\phi)$ elementary?
  - ↪ **No.** Logically and locally consistent but **not maximal** because $\neg a \wedge b \in closure(\phi)$, yet $\neg a \wedge b \notin B$ and $\neg(\neg a \wedge b) \notin B$.
- Is $B = \{a, b, \neg a \wedge b, \phi\} \subset closure(\phi)$ elementary?
  - ↪ **No.** It is **not logically consistent** because $a \in B$ and $\neg a \wedge b \in B$.
- Is $B = \{\neg a, \neg b, \neg(\neg a \wedge b), \phi\} \subset closure(\phi)$ elementary?
  - ↪ **No.** Logically consistent but **not locally consistent** because $\phi = a \,\mathsf{U}\, (\neg a \wedge b) \in B$ and $\neg a \wedge b \notin B$ but $a \notin B$.

From LTL to GNBA

Elementary sets: examples (2/2)

Let $\phi = a \,\mathsf{U}\, (\neg a \wedge b)$:

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

All elementary sets?

$\Longrightarrow$ **Blackboard construction.**

All elementary sets:

$$B_1 = \{a, b, \neg(\neg a \wedge b), \phi\},$$
$$B_2 = \{a, b, \neg(\neg a \wedge b), \neg\phi\},$$
$$B_3 = \{a, \neg b, \neg(\neg a \wedge b), \phi\},$$
$$B_4 = \{a, \neg b, \neg(\neg a \wedge b), \neg\phi\},$$
$$B_5 = \{\neg a, \neg b, \neg(\neg a \wedge b), \neg\phi\},$$
$$B_6 = \{\neg a, b, \neg a \wedge b, \phi\}.$$

From LTL to GNBA

Construction of $\mathcal{G}_\phi$ (1/2)

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

- $Q = \{B \subseteq closure(\phi) \mid B \text{ is elementary}\}$,
- $Q_0 = \{B \in Q \mid \phi \in B\}$,
- $\mathcal{F} = \{F_{\phi_1 \,\mathsf{U}\, \phi_2} \mid \phi_1 \,\mathsf{U}\, \phi_2 \in closure(\phi)\}$ with

$$F_{\phi_1 \,\mathsf{U}\, \phi_2} = \{B \in Q \mid \phi_1 \,\mathsf{U}\, \phi_2 \notin B \vee \phi_2 \in B\}.$$

Intuition: for any run $B_0 B_1 B_2 \ldots$, if $\phi_1 \,\mathsf{U}\, \phi_2 \in B_0$, then ϕ_2 must eventually become true ($\rightsquigarrow$ ensured by the acceptance condition).

Observe that $\mathcal{F} = \emptyset$ if no until in ϕ.

$\Longrightarrow$ All runs are accepting in this case.

From LTL to GNBA

Construction of $\mathcal{G}_\phi$ (2/2)

The transition relation $\delta\colon Q \times 2^{AP} \to 2^{Q}$ 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 $\bigcirc \psi \in closure(\phi)$, $\bigcirc \psi \in B \iff \psi \in B'$, and
  - (ii) for every $\phi_1 \mathsf{U}\, \phi_2 \in closure(\phi)$,

$$\phi_1 \mathsf{U}\, \phi_2 \in B \iff \Big(\phi_2 \in B \vee (\phi_1 \in B \wedge \phi_1 \mathsf{U}\, \phi_2 \in B')\Big).$$

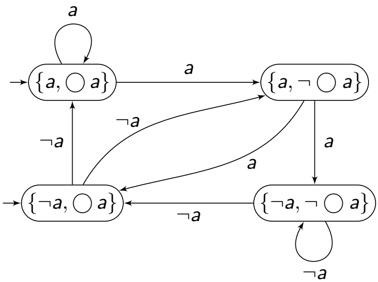
Intuition: (i) and (ii) reflect the semantics of $\bigcirc$ and U operators, (ii) is based on the expansion law.

From LTL to GNBA

Example: $\phi = \bigcirc a$

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

$\Longrightarrow$ **Blackboard construction of the GNBA + proof.**



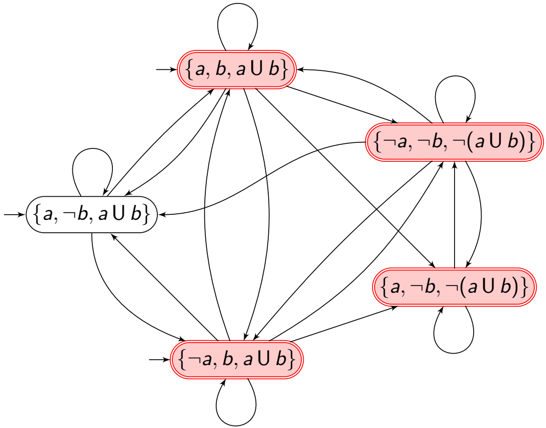
- $Q = \big\{\{a, \bigcirc a\}, \{a, \neg \bigcirc a\}, \{\neg a, \bigcirc a\}, \{\neg a, \neg \bigcirc a\}\big\}$,
- $Q_0 = \big\{\{a, \bigcirc a\}, \{\neg a, \bigcirc a\}\big\}$,
- $\mathcal{F} = \emptyset$.

From LTL to GNBA

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

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

$\Longrightarrow$ **Blackboard construction of the GNBA.**



From LTL to GNBA

Example: $\phi = a \, \mathsf{U} \, b$ (2/3)

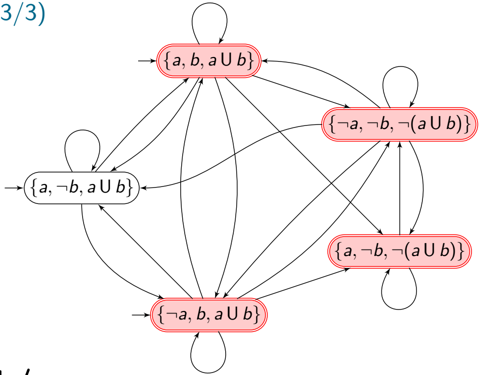
Some explanations (see blackboard for more).

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

- $Q = \{B_1, B_2, B_3, B_4, B_5\}$, $Q_0 = \{B_1, B_2, B_3\}$.
- $\mathcal{F} = \{F_{a \, \mathsf{U} \, b}\} = \big\{\{B_1, B_2, B_4, B_5\}\big\}$.
  - $\hookrightarrow$ $\mathcal{G}_\phi$ 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 $a \, \mathsf{U} \, b$ is already fulfilled by $b \in B_1$ (resp. B_2).
- From B_3, we need to go where $a \, \mathsf{U} \, b$ holds: B_1, B_2 or B_3.
- From B_4, we can go anywhere because $\neg(a \, \mathsf{U} \, b)$ is already fulfilled by $\neg a, \neg b \in B_4$.
- From B_5, we need to go where $\neg(a \, \mathsf{U} \, b)$ holds: B_4 or B_5.

From LTL to GNBA

Example: $\phi = a \,\mathsf{U}\, b$ (3/3)



Sample words/runs:

- $\sigma = \{a\}\,\{a\}\,\{b\}^\omega \in Words(\phi)$ has accepting run $\overline{\sigma} = B_3 B_3 B_2^\omega$ in $\mathcal{G}_\phi$.
- $\sigma = \{a\}^\omega \notin Words(\phi)$ has only one run $\overline{\sigma} = B_3^\omega$ in $\mathcal{G}_\phi$ and it is not accepting since $B_3 \notin F_{a \,\mathsf{U}\, b}$.

From LTL to... NBA

Construction

Idea: LTL $\rightsquigarrow$ GNBA $\rightsquigarrow$ NBA.

Theorem: LTL to NBA

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

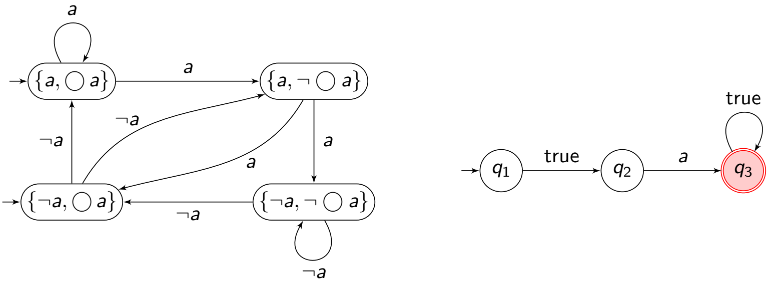
Sketch

1. Construct the GNBA $\mathcal{G}_\phi$.
   - $|closure(\phi)| = \mathcal{O}(|\phi|)$ and $|Q| \leq 2^{|closure(\phi)|} = 2^{\mathcal{O}(|\phi|)}$.
   - # accepting sets of $\mathcal{G}_\phi$ = # until-operators in $\phi \leq \mathcal{O}(|\phi|)$.
2. Construct the NBA $\mathcal{A}_\phi$.
   - # states of $\mathcal{A}_\phi = |Q| \times$ # accepting sets of $\mathcal{G}_\phi$.
   - # states of $\mathcal{A}_\phi$ $\leq 2^{\mathcal{O}(|\phi|)} \cdot \mathcal{O}(|\phi|) = 2^{\mathcal{O}(|\phi|)} \cdot 2^{\log(\mathcal{O}(|\phi|))} = 2^{\mathcal{O}(|\phi|)}$.

From LTL to... NBA

Can we do better? (1/3)

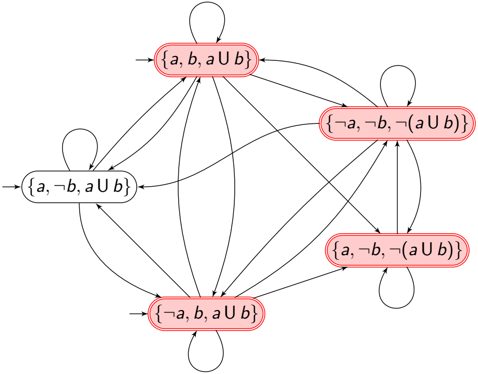
The algorithm presented here is conceptually simple but may lead to unnecessary large GNBAs (and thus NBAs).

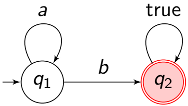


Example: the right NBA also recognizes $\bigcirc a$ but is *smaller*.

From LTL to... NBA

Can we do better? (2/3)





Example: the right NBA also recognizes $a \mathbin{\mathsf{U}} b$ but is *much smaller*.

Can we always do better?

From LTL to... NBA

Can we do better? (3/3)

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 ϕ_n with $|\phi_n| = \mathcal{O}(poly(n))$ such that every NBA $\mathcal{A}_{\phi_n}$ for ϕ_n has at least 2^n states.

$\Longrightarrow$ **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

$$\mathcal{L}_n = \left\{ A_1 \ldots A_n A_1 \ldots A_n \sigma \mid A_i \subseteq AP \wedge \sigma \in (2^{AP})^\omega \right\} \quad \text{for } n \geq 0.$$

This language is expressible in LTL, i.e., $\mathcal{L}_n = Words(\phi_n)$ for

$$\phi_n = \bigwedge_{a \in AP} \bigwedge_{0 \leq i < n} (\bigcirc^i a \longleftrightarrow \bigcirc^{n+i} a).$$

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

Claim: any NBA $\mathcal{A}$ with $\mathcal{L}_\omega(\mathcal{A}) = \mathcal{L}_n$ has at least 2^n states.

From LTL to... NBA

Lower bound proof (2/2)

Assume $\mathcal{A}$ is such an automaton. Words $A_1 \dots A_n A_1 \dots A_n \emptyset^\omega$ belong to $\mathcal{L}_n$, hence are accepted by $\mathcal{A}$.

- For every word $A_1 \dots A_n$ of length n, $\mathcal{A}$ has a state $q(A_1 \dots A_n)$ which can be reached after consuming $A_1 \dots A_n$.
- From $q(A_1 \dots A_n)$, it is possible to visit an accept state infinitely often by reading the suffix $A_1 \dots A_n \emptyset^\omega$.
- If $A_1 \dots A_n \neq A'_1 \dots A'_n$, then

$$A_1 \dots A_n A'_1 \dots A'_n \emptyset^\omega \notin \mathcal{L}_n = \mathcal{L}_\omega(\mathcal{A}).$$

- Therefore, **states $q(A_1 \dots A_n)$ are all pairwise different**.
- Since each A_i can take $2^{|AP|}$ different values, the number of different sequences $A_1 \dots A_n$ 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

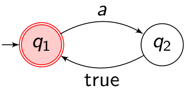
What have we learned?

Corollary

Every LTL formula expresses an ω-regular property, i.e., for all LTL formula ϕ, $Words(\phi)$ is an ω-regular language.

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

The converse is false!



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

$\Longrightarrow$ **There are ω-regular properties not expressible in LTL.**

Back to the model checking algorithm for LTL

What do we still need?

$$
\begin{aligned}
\mathcal{T} \models \phi \quad & \text{iff} \quad \mathit{Traces}(\mathcal{T}) \subseteq \mathit{Words}(\phi) \\
& \text{iff} \quad \mathit{Traces}(\mathcal{T}) \cap ((2^{AP})^\omega \setminus \mathit{Words}(\phi)) = \emptyset \\
& \text{iff} \quad \mathit{Traces}(\mathcal{T}) \cap \mathit{Words}(\neg\phi) = \emptyset \\
& \text{iff} \quad \mathit{Traces}(\mathcal{T}) \cap \mathcal{L}_\omega(\mathcal{A}_{\neg\phi}) = \emptyset \\
& \text{iff} \quad \mathcal{T} \otimes \mathcal{A}_{\neg\phi} \models \Diamond\Box\neg F
\end{aligned}
$$

It remains to consider the last line.

Two remaining questions:

1. How to compute the product TS $\mathcal{T} \otimes \mathcal{A}_{\neg\phi}$?
2. How to check persistence, i.e., $\mathcal{T} \otimes \mathcal{A}_{\neg\phi} \models \Diamond\Box\neg F$?

Product of TS and NBA

Definition

Definition: product of TS and NBA

Let $\mathcal{T} = (S, Act, \longrightarrow, I, AP, L)$ 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:

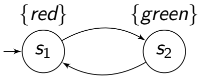
$$\mathcal{T} \otimes \mathcal{A} = (S', Act, \longrightarrow', I', AP', L') \quad \text{where}$$

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

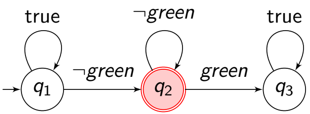
Product of TS and NBA

Example: simple traffic light

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

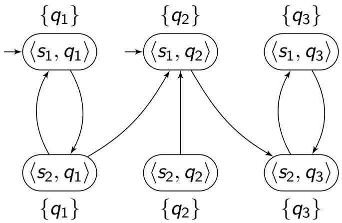


TS $\mathcal{T}$ for the traffic light.



NBA $\mathcal{A}_{\neg\phi}$ for $\neg\phi = \Diamond\Box\neg green$.

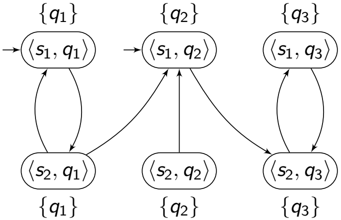
$\Longrightarrow$ Blackboard construction of $\mathcal{T} \otimes \mathcal{A}_{\neg\phi}$.



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, $\mathcal{T} \otimes \mathcal{A}_{\neg\phi} \overset{?}{\models} \Diamond\Box\neg F$ 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.

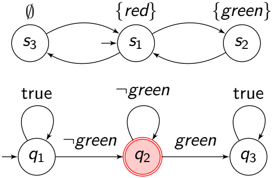
$\Longrightarrow$ There is no common trace between $\mathcal{T}$ and $\mathcal{A}_{\neg\phi}$.

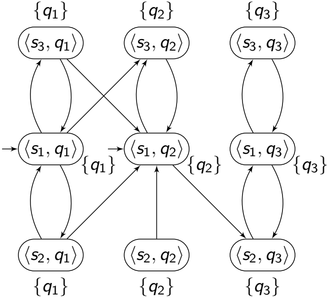
$\Longrightarrow \mathcal{T} \models \phi$.

Persistence checking

Illustration (2/2)

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





Here, $\mathcal{T} \otimes \mathcal{A}_{\neg\phi} \not\models \Diamond\Box\neg F$ 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.

$\Longrightarrow$ Path $\pi = (s_1 s_3)^\omega$ of $\mathcal{T}$ gives trace $\sigma = (\{\text{red}\}\,\emptyset)^\omega$ which is accepted by $\mathcal{A}_{\neg\phi}$ (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 Φ a *propositional* formula over AP, then

$$\mathcal{T} \not\models \Diamond\Box\Phi$$
$$\Updownarrow$$
$$\exists\, s \in \mathit{Reach}(\mathcal{T}),\ s \not\models \Phi \text{ and } s \text{ is on a cycle in the graph of } \mathcal{T}.$$

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

Persistence checking

Algorithmic solutions for cycle detection

1. Compute the reachable SCCs and check if one contains a state satisfying $\neg\Phi$.
   - ↪ Linear time but requires to construct entirely the product TS $\mathcal{T} \otimes \mathcal{A}_{\neg\phi}$ which may be very large (exponential).
2. Another solution: **on-the-fly algorithms**.
   - ▷ Construct $\mathcal{T}$ and $\mathcal{A}_{\neg\phi}$ in parallel and simultaneously construct the reachable fragment of $\mathcal{T} \otimes \mathcal{A}_{\neg\phi}$ via nested depth-first search.
   - ↪ Construction of the product "on demand".
   - ↪ **More efficient in practice** (used in software solutions such as Spin).

$\Longrightarrow$ **See the book for more.**

Still, the complexity of LTL model checking remains high!

Wrap-up of the automata-based approach

$$\begin{aligned}
\mathcal{T} \models \phi \quad & \text{iff} \quad \mathit{Traces}(\mathcal{T}) \subseteq \mathit{Words}(\phi) \\
& \text{iff} \quad \mathit{Traces}(\mathcal{T}) \cap ((2^{AP})^\omega \setminus \mathit{Words}(\phi)) = \emptyset \\
& \text{iff} \quad \mathit{Traces}(\mathcal{T}) \cap \mathit{Words}(\neg\phi) = \emptyset \\
& \text{iff} \quad \mathit{Traces}(\mathcal{T}) \cap \mathcal{L}_\omega(\mathcal{A}_{\neg\phi}) = \emptyset \\
& \text{iff} \quad \mathcal{T} \otimes \mathcal{A}_{\neg\phi} \models \Diamond\Box\neg F
\end{aligned}$$

Complexity of this approach

The time and space complexity is $\mathcal{O}(|\mathcal{T}|) \cdot 2^{\mathcal{O}(|\phi|)}$.

Complexity of LTL model checking

Complexity of the model checking problem for LTL

The LTL model checking problem is PSPACE-complete.

$\Longrightarrow$ **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.
MIT Press, 2008.