Model-checking LTL specifications

(IF311 : "Formal methods in software design")

Frédéric Herbreteau (fh@labri.fr)

Bordeaux INP

What is model-checking?

`INPUT:`

- A transition system M
- An LTL formula Φ

`Determine whether:`

$$M \models \Phi$$

$\rightarrow$ check that **every infinite run** in M satisfies Φ.

LTL formulas as languages (1/2)

Set of atomic propositions AP, trace alphabet $\Sigma = 2^{AP}$.

- $[\![a]\!] = \{A \in \Sigma \mid a \in A\}\Sigma^\omega$
- $[\![\neg\alpha]\!] = \Sigma^\omega - [\![\alpha]\!]$
- $[\![\alpha \vee \beta]\!] = [\![\alpha]\!] \cup [\![\beta]\!]$
- $[\![\mathsf{X}\alpha]\!] = \{w \in \Sigma^\omega \mid w_{1\ldots} \in [\![\alpha]\!]\} = \Sigma\, [\![\alpha]\!]$
- $[\![\alpha \mathsf{U} \beta]\!] = \{w \in \Sigma^\omega \mid \exists i.\ w_{i\ldots} \in [\![\beta]\!],\ \forall j < i.\ w_{j\ldots} \in [\![\alpha]\!]\}$

where $w_{i\ldots} = w_i w_{i+1} \ldots$ for $w = w_0 w_1 w_2 \cdots w_i w_{i+1} \cdots$.

LTL formulas as languages (2/2)

Example

$AP = \{a, b\}$, $\Sigma = \{\emptyset, \{a\}, \{b\}, \{a, b\}\}$

- $[\![a]\!] =$
- $[\![\neg a]\!] =$
- $[\![a \vee b]\!] =$
- $[\![\mathsf{X} a]\!] =$
- $[\![a \mathsf{U} b]\!] =$

Satisfaction as a automaton problem

- $\text{trace}(\pi) = L(s_0)L(s_1)\cdots L(s_i)\cdots$ for $\pi = s_0 s_1 \cdots s_i \cdots$
- $\text{trace}(M) = \{\text{trace}(\pi) \mid \pi \text{ infinite run of M}\}$

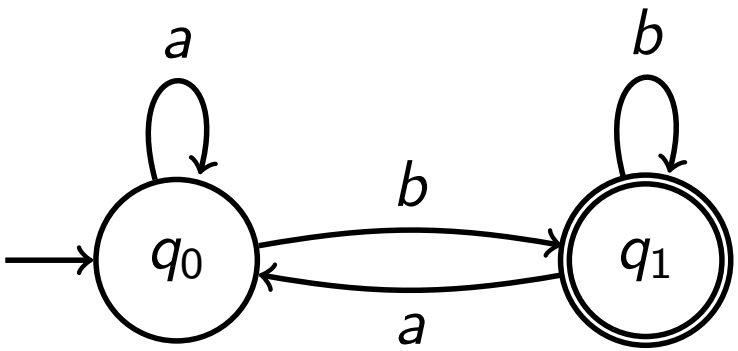
Then:

$$
\begin{aligned}
M \models \Phi \text{ iff } & \text{trace}(M) \subseteq [\![\Phi]\!] \\
\text{iff } & \text{trace}(M) \cap \overline{[\![\Phi]\!]} = \emptyset \\
\text{iff } & \text{trace}(M) \cap [\![\neg\Phi]\!] = \emptyset \\
\text{iff } & \text{trace}(M) \cap \mathcal{L}(\mathcal{A}_{\neg\Phi}) = \emptyset
\end{aligned}
$$

where $\mathcal{A}_{\neg\Phi}$ is an **automaton** that accepts $[\![\neg\Phi]\!]$.

From LTL formulas to automata

Automata on finite words

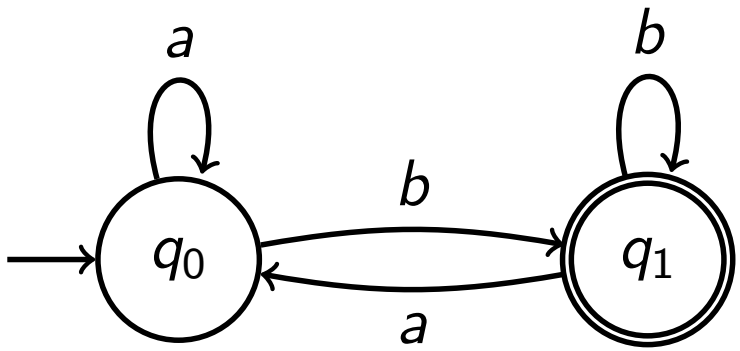


A word w is accepted if $q_0 \xrightarrow{w} q$ with $q \in F$.

Example

- $aabb$ is accepted
- aba is not accepted

Automata on infinite words (Büchi)



$$F = \{q_1\}$$

Büchi acceptance: w is accepted if there is a run that visits an **accepting state infinitely many times** (i.e. every suffix of the run has an accepting state)

Example

- $a(bba)^\omega$ is accepted
- $abbbaa^\omega$ is not accepted
- language accepted by the automaton above?

Examples of Büchi automata

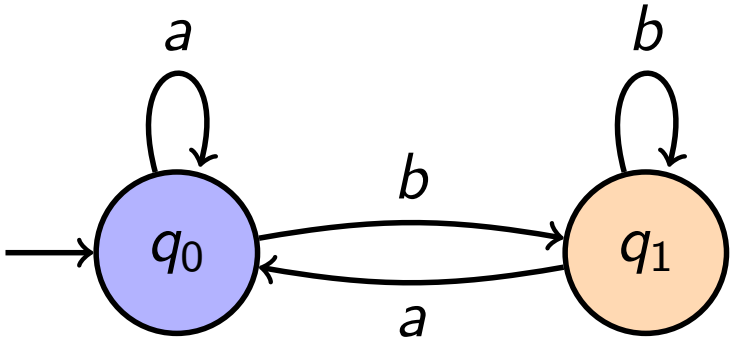
Example

Büchi automata and LTL formulas for the following languages on $\{0, 1\}$:

1. $\{w \mid 0 \text{ occurs exactly once in } w\}$
2. $\{w \mid \text{each } 0 \text{ is eventually followed by a } 1\}$
3. $\{w \mid w \text{ contains finitely many 1's}\}$
4. $\{(01)^n 0^\omega \mid n \in \mathbb{N}\}$
5. $\{w \mid 0 \text{ occurs in every even position of } w\}$

Give an infinite regular language that is not accepted by a **deterministic** Büchi automaton?

Automata on infinite words (Gen. Büchi)



$$F = \{\{q_0\},\ \{q_1\}\}.$$

Generalized Büchi acceptance: w is accepted if there is a run that visits **each accepting set infinitely many times**

Example

- $a(bba)^\omega$ is accepted
- $abbbaa^\omega$ is not accepted
- language accepted by the automaton above?

Principle of automata construction

Consider a word $w = A_0 A_1 \cdots A_i$ that satisfies Φ

Let B_i be the set of **subformulas of Φ that are true on the suffix of w from i**: $\Psi \in B_i$ iff $w_{i\ldots} \in [\![\Psi]\!]$

Example

Let $\Phi = a \,\mathsf{U}\, b$ over $AP = \{a, b, c\}$, subformulas: $\{a, b, a \,\mathsf{U}\, b\}$

w :	$\{a, c\}$	$\{a\}$	$\{a, c\}$	$\{b\}$	$\{b\}$	$\cdots$
B :	$\{a, a\mathsf{U}b\}$	$\{a, a\mathsf{U}b\}$	$\{a, a\mathsf{U}b\}$	$\{b, a\mathsf{U}b\}$	$\{b, a\mathsf{U}b\}$	$\cdots$

$w \in [\![a \,\mathsf{U}\, b]\!]$ since $(a \,\mathsf{U}\, b) \in B_0$

Principle of automata construction (a)

Example

Let $\Phi = a$ and $AP = \{a, b, c\}$

w :	$\{a, c\}$	$\{a, b\}$	$\{b, c\}$	$\{a, c\}$	$\cdots$
B :	$\{a\}$	$\{a\}$	$\emptyset$	$\{a\}$	$\cdots$

$w \in [\![a]\!]$ since $a \in B_0$

w :	$\{c\}$	$\{a, b\}$	$\{b, c\}$	$\{a, c\}$	$\cdots$
B :	$\emptyset$	$\{a\}$	$\emptyset$	$\{a\}$	$\cdots$

$w \notin [\![a]\!]$ since $a \notin B_0$

Principle of automata construction (X)

Example

Let $\Phi = \mathrm{X}a$ and $AP = \{a, b, c\}$

Temporal consistency: if $(\mathrm{X}a) \in B_i$ then $a \in B_{i+1}$.

w :	$\{c\}$	$\{a, b\}$	$\{b, c\}$	$\{a, c\}$	$\cdots$
B :	$\{\mathrm{X}a\}$	$\{a\}$	$\{\mathrm{X}a\}$	$\{a\}$	$\cdots$

$w \in [\![\mathrm{X}a]\!]$ since $(\mathrm{X}a) \in B_0$ and consistent

w :	$\{c\}$	$\{b\}$	$\{b, c\}$	$\{a, c\}$	$\cdots$
B :	$\emptyset$	$\{b\}$	$\{\mathrm{X}a\}$	$\{a\}$	$\cdots$

$w \notin [\![\mathrm{X}a]\!]$ since no consistent B with $(\mathrm{X}a) \in B_0$.

Principle of automata construction (U)

Example

Let $\Phi = a \mathsf{U} b$ and $AP = \{a, b, c\}$

Temporal consistency: if $(a \mathsf{U} b) \in B_i$ then either $b \in B_i$, or $a \in B_i$ and $(a \mathsf{U} b) \in B_{i+1}$.

w :	$\{a, c\}$	$\{a\}$	$\{b, c\}$	$\{a, c\}$	$\cdots$
B :	$\{a, a \mathsf{U} b\}$	$\{a, a \mathsf{U} b\}$	$\{b, a \mathsf{U} b\}$	$\{a\}$	$\cdots$

$w \in [\![a \mathsf{U} b]\!]$ since $(a \mathsf{U} b) \in B_0$, consistent (observe $b \in B_2$)

w :	$\{a, c\}$	$\{c\}$	$\{b, c\}$	$\{a, c\}$	$\cdots$
B :	$\{a\}$	$\{c\}$	$\{b, a \mathsf{U} b\}$	$\{a\}$	$\cdots$

$w \notin [\![a \mathsf{U} b]\!]$ since no consistent B with $(a \mathsf{U} b) \in B_0$

From Φ to $\mathcal{A}_\Phi$: states

$\mathcal{A}_\Phi = (Q, 2^{AP}, \Delta, I, F)$

- the **states** Q of $\mathcal{A}_\phi$ are the maximal and consistent sets B of subformulas of ϕ
  $\rightarrow$ tell if each subformula is true or false

Example

Let $\Phi = a\mathsf{U}(\neg a \wedge b)$

Subformulas of Φ: $\{a, b, \neg a, \neg a \wedge b, a\mathsf{U}(\neg a \wedge b)\}$

$\{a, \neg a\}$ and $\{a, b, \neg a \wedge b\}$ are not consistent

$\{\neg a, b\}$ and $\{\neg a, b, \neg a \wedge b\}$ are not maximal

States: $\emptyset$, $\{a\}$, $\{a, a\mathsf{U}(\neg a \wedge b)\}$, $\{a, b\}$, $\{a, b, a\mathsf{U}(\neg a \wedge b)\}$ and $\{b, \neg a \wedge b, a\mathsf{U}(\neg a \wedge b)\}$.

- **Initial states** are the sets B such that $\Phi \in B$.

From Φ to $\mathcal{A}_\Phi$: transitions

$\mathcal{A}_\Phi = (Q, 2^{AP}, \Delta, I, F)$

Transitions $B \xrightarrow{A} B'$ (partially) guarantee that all formulas in B are true on the suffix of w starting from A.

- **logically consistent:** $A = B \cap AP$
- **temporally consistent:**
  - if $\mathsf{X}\phi \in B$ then $\phi \in B'$
  - if $\phi_1 \mathsf{U} \phi_2 \in B$ then $\phi_2 \in B$, or $\phi_1 \in B$ and $\phi_1 \mathsf{U} \phi_2 \in B'$

    Observe that:

$$\phi_1 \mathsf{U} \phi_2 \quad \equiv \quad \phi_2 \vee (\phi_1 \wedge \mathsf{X}(\phi_1 \mathsf{U} \phi_2))$$

From Φ to $\mathcal{A}_\Phi$: transitions

$\mathcal{A}_\Phi = (Q, 2^{AP}, \Delta, I, F)$

Accepting states guarantee that for every subformula $\phi_1 \mathsf{U} \phi_2$, ϕ_2 eventually becomes true.

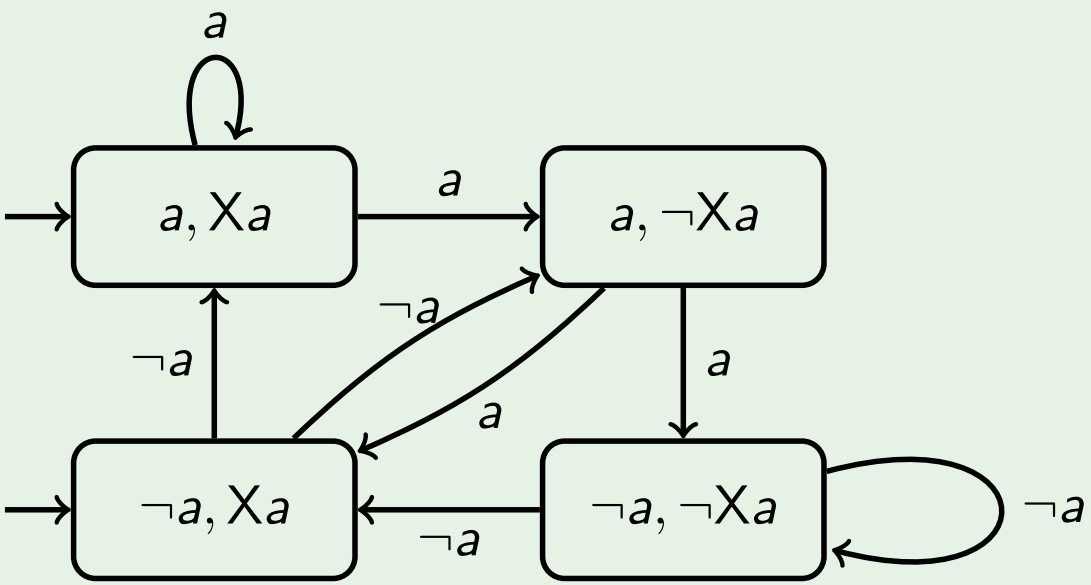
- F is a **generalized Büchi** condition with one accepting set in F per subformula U in Φ.

  Accepting condition:
  $F_{\phi_1 \mathsf{U} \phi_2} = \{B \in Q \mid \phi_1 \mathsf{U} \phi_2 \notin B \text{ or } \phi_2 \in B\}$

  ensures that for every run starting from some B that contains $\phi_1 \mathsf{U} \phi_2$, then ϕ_2 eventually becomes true.

Automaton for $\mathsf{X}a$

Example (automaton for $\mathsf{X}a$)



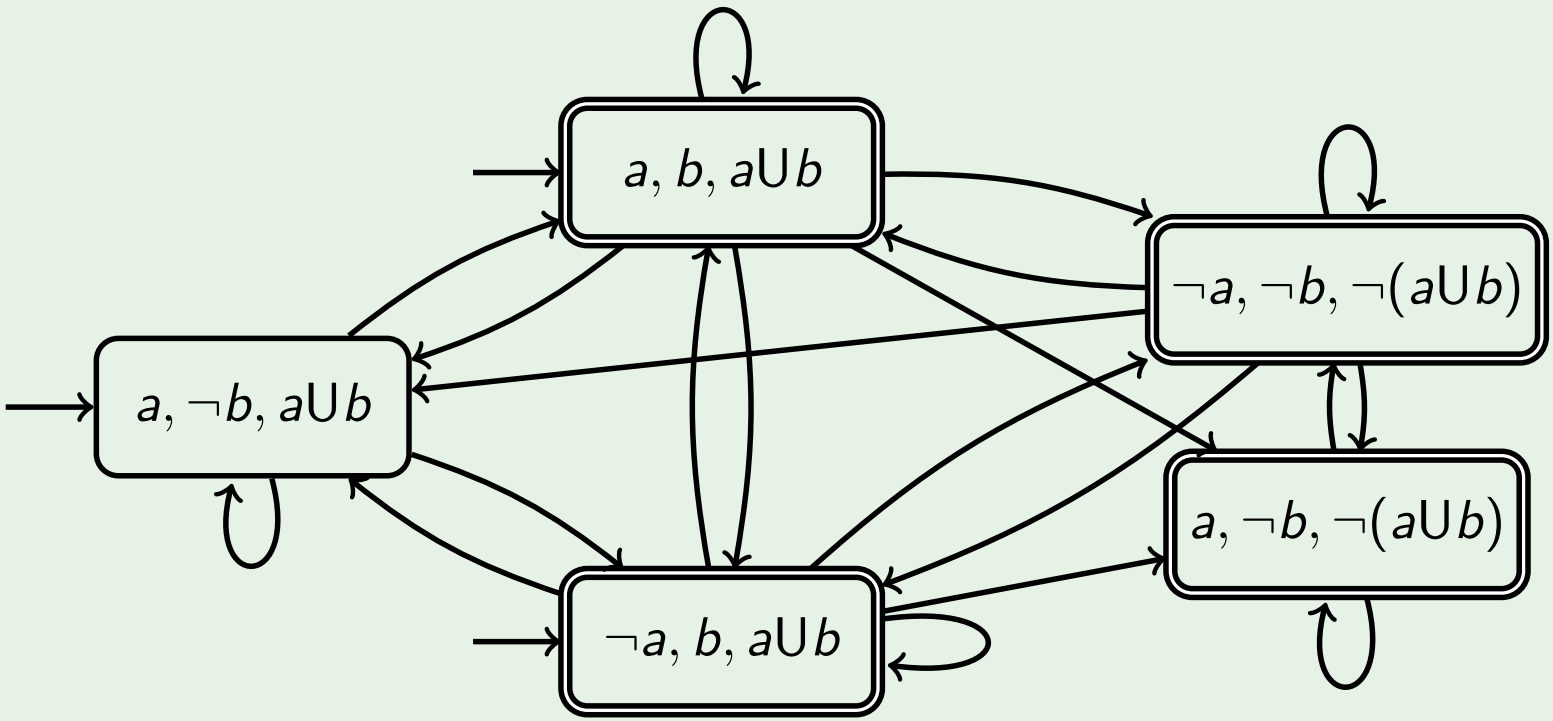
All runs are accepting since $F = \emptyset$.

Runs on:

- $\{a\}\{a\}\emptyset\{a\}\emptyset\{a\}\emptyset\cdots$
- $\emptyset\emptyset\{a\}\{a\}\{a\}\{a\}\{a\}\cdots$

Automaton for $a \mathsf{U} b$

Example (automaton for $a \mathsf{U} b$)



Runs on:

- $\{a\}\{a\}\{b\}\emptyset\{a\}\emptyset\cdots$
- $\{a\}\{a\}\{a\}\{a\}\{a\}\{a\}\cdots$

Efficient algorithms and implementations

- Our construction builds an automaton with $2^{\mathcal{O}(\Phi)}$ **states and an alphabet of size** 2^{AP} such that $\mathcal{L}(\mathcal{A}_\Phi) = [\![\Phi]\!]$.
- The **satisfiability** problem for LTL is PSPACE-complete.
- The resulting automaton has **generalized accepting conditions** but it can be converted in a $\mathcal{O}(|F|)$ bigger automaton with **standard Büchi accepting conditions**.
- Efficient algorithms and implementations manage to build "small" automata most of the time:

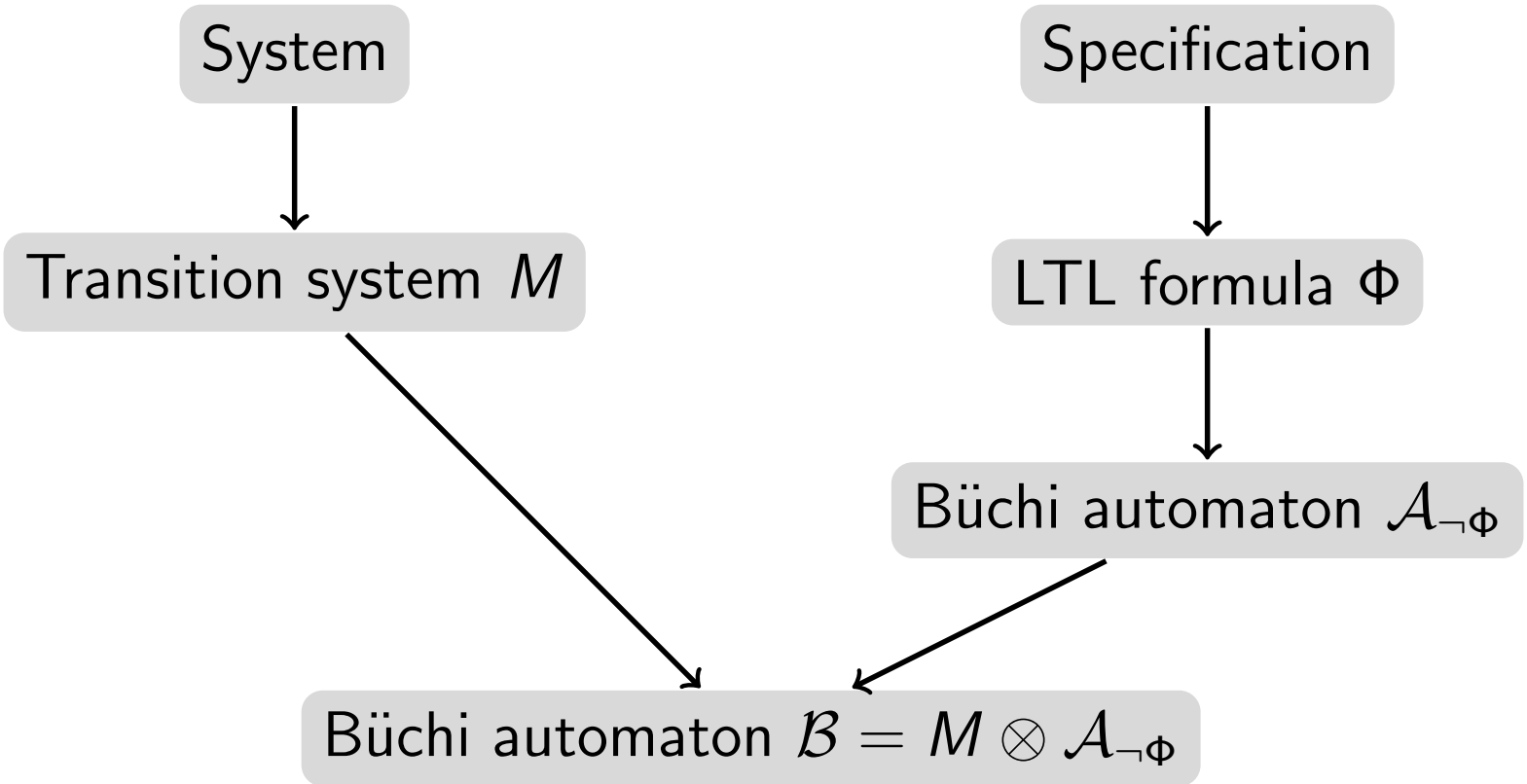
`https://spot.lrde.epita.fr/app/`

Example

Compare automata computed by SPOT with the ones above for Xa and aUb.

Emptiness checking algorithms

Model-checking algorithm



$$M \models \Phi \quad \text{iff} \quad \text{trace}(M) \subseteq [\![\Phi]\!] \quad \text{iff} \quad \mathcal{L}(\mathcal{B}) = \emptyset$$

The product $M \otimes \mathcal{A}_{\neg\Phi}$

$M = (S, S_0, \rightarrow, L)$ and $\mathcal{A}_{\neg\Phi} = (Q, 2^{AP}, \Delta, I, F)$, **non-blocking**

Goal: $\mathcal{B} = M \otimes \mathcal{A}_{\neg\Phi}$ is a Büchi automaton that accepts all traces of M that falsify Φ

$\mathcal{B} = (S \times Q, \emptyset, \Delta', I', F')$ with transitions in Δ' defined as:

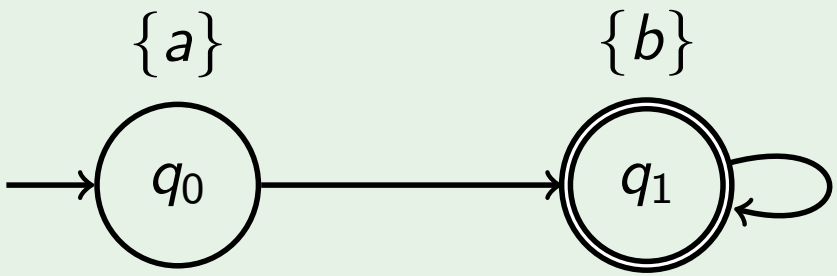
$$\frac{s \rightarrow s' \qquad B \xrightarrow{L(s')} B'}{(s, B) \rightarrow (s', B')}$$

- $I' = \{(s_0, B) \mid s_0 \in S_0 \text{ and } B_0 \xrightarrow{L(s_0)} B \text{ for some } B_0 \in I\}$
- $F' = \{(s, B) \mid B \in F\}$

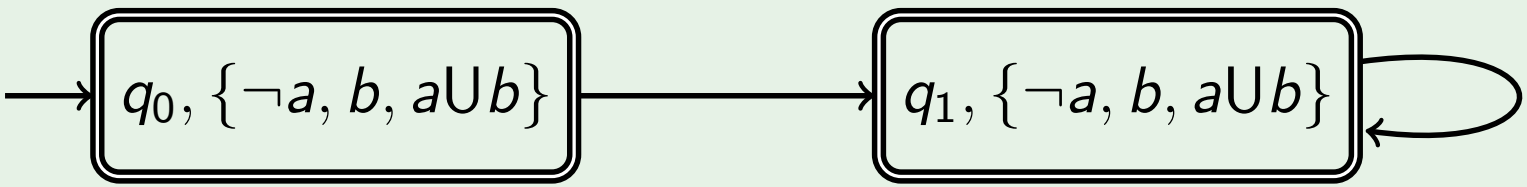
Product $M \otimes \mathcal{A}_{a\mathsf{U}b}$

Example

Transition system M:



Product $\mathcal{B} = M \otimes \mathcal{A}_{a\mathsf{U}b}$:



NB: initial state comes from $\{a, \neg b, a\mathsf{U}b\} \xrightarrow{\{a\}} \{\neg a, b, a\mathsf{U}b\}$ in $\mathcal{A}_{a\mathsf{U}b}$

Naive emptiness checking algorithm

Reduction: $M \models \Phi$ iff $\mathcal{L}(M \otimes \mathcal{A}_{\neg\Phi}) = \emptyset$

Goal: determine when $\mathcal{L}(\mathcal{B}) = \emptyset$ for a Büchi automaton $\mathcal{B}$.

Property

A Büchi automaton $\mathcal{B}$ has an accepting run iff it has an accepting state that is reachable both from an initial state and from itself

Naive algorithm:

- Enumerate every accepting $f \in F$ from $\mathcal{B}$
- Check if $DFS(q_0, f)$ and $DFS(f, f)$ both return true for some initial state q_0 of $\mathcal{B}$
- Complexity: $\mathcal{O}(|\mathcal{B}|^2)$ (recall $|\mathcal{B}| = 2^{\mathcal{O}(|\Phi|)}$)

Nested depth-first search algorithm

```
1   INPUT: Buechi automaton B = (Q, Σ, Δ, I, F)
2   RETURN: "accepting run" if L(B) ≠ ∅, "no accepting run" otherwise
3
4   forall q0 ∈ I do
5     dfs_blue(q0)
6   report "no accepting run"
7
8   procedure dfs_blue(s):
9     s.color := cyan
10    forall s → t in Δ do
11      if (t.color = cyan and (s ∈ F or t ∈ F))
12        report "accepting run" and exit
13      if (t.color = white)
14        dfs_blue(t)
15    if (s ∈ F)
16      dfs_red(s)
17    else
18      s.color := blue
19
20  procedure dfs_red(s):
21    s.color := red
22    forall s → t in Δ do
23      if (t.color = cyan)
24        report "accepting run" and exit
25      else if (t.color = blue)
26        dfs_red(t)
```

Runs in time and memory $\mathcal{O}(|\mathcal{B}|)$

Couvreur's SCC-decomposition algorithm

```
1   INPUT: Buechi automaton B = (Q, Σ, Δ, I, F)
2   RETURN: "accepting run" if L(B) ≠ ∅, "no accepting run" otherwise
3
4   count := 0; Roots := ∅; Active := ∅
5   forall q0 ∈ I do
6     couv(q0)
7   report "no accepting run"
8
9   procedure couv(q):
10    count := count + 1
11    q.dfsnum := count
12    push(Roots, q)
13    push(Active, q)
14    forall q -A-> q' in Δ do
15      if (q'.dfsnum = 0)
16        couv(q')
17      else if (q' ∈ Active)
18        repeat
19          u := pop(Roots)
20          if (u ∈ F)
21            report "accepting run" and exit
22        until (u.dfsnum ≤ q'.dfsnum)
23        push(Roots, u)
24    if (top(Roots) = q)
25      pop(Roots)
26      repeat
27        u := pop(Active)
28      until (u == q)
```

Runs in time and memory $\mathcal{O}(|\mathcal{B}|)$