An Introduction to Model Checking

Alexandre Duret-Lutz

23 August 2010
IIT Jodhpur

http://www.lrde.epita.fr/~adl/ens/mc/iitj.pdf

Formal Verification

$$\begin{tikzcd}\text{system} \arrow[r, leftrightarrow, "\text{tests?}", "\text{not exhaustive!}"'] & \text{properties}\end{tikzcd}$$

Formal Verification

$$\begin{tikzcd}
\text{system} \arrow[d, bend left] & & \text{properties} \arrow[d, bend left] \\
\text{model of the system} & & \text{model of the properties}
\end{tikzcd}$$

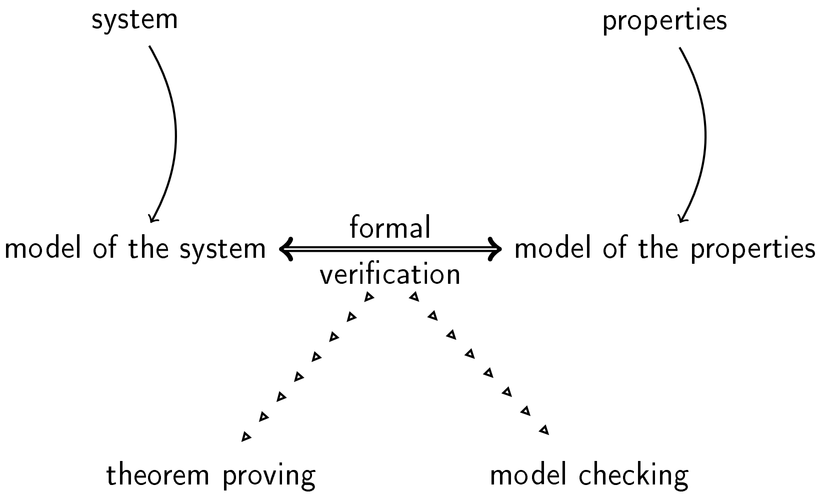
Formal Verification

$$\begin{tikzcd}
\text{system} \arrow[d, bend left] & & \text{properties} \arrow[d, bend left] \\
\text{model of the system} \arrow[rr, leftrightarrow, "\text{simulation?}", "\text{not exhaustive!}"'] & & \text{model of the properties}
\end{tikzcd}$$

Formal Verification

$$\begin{tikzcd}
\text{system} \arrow[d, bend left] & & \text{properties} \arrow[d, bend left] \\
\text{model of the system} \arrow[rr, Leftrightarrow, "\text{formal}", "\text{verification}"'] & & \text{model of the properties}
\end{tikzcd}$$

Formal Verification



Theorem Proving

1. Describe the system in a way that allows reasoning
2. Prove property by logical reasoning

This can be entirely manual, or using the help of a theorem prover (e.g. Coq) that is not fully automatic.

Problem: it is hard to produce a counterexample when a theorem is false.

Research work in the area: new proof systems, study of the expressive power of various logics...

Model checking

An automatic approach to formal verification.

An exhaustive verification of all behaviors of a model.

The catch: the model has to be abstract enough (i.e. not too detailed) to allow its complete exploration.

Example: an Algorithm for Mutual Exclusion

Global variables: req_P and req_Q.

Process P (infinite loop)

```
1. $req_P \leftarrow 1$
2. $wait(req_Q = 0)$
3. Critical Section
4. $req_P \leftarrow 0$
```

Process Q (infinite loop)

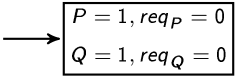
```
1. $req_Q \leftarrow 1$
2. $wait(req_P = 0)$
3. Critical Section
4. $req_Q \leftarrow 0$
```

Initial state: $P = 1$, $Q = 1$, $req_P = 0$, $req_Q = 0$.

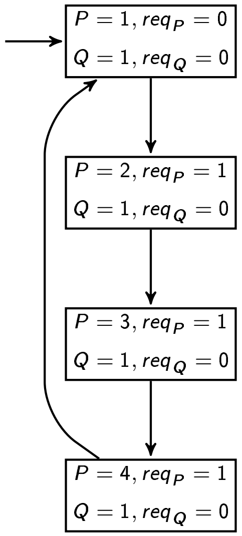
Properties to check:

1. At any time, there is at most one process in Critical Section.
2. Any process requesting entrance to the CS will eventually enter it.
3. The order of entrances to the CS should follow the order of requests.

Example: State Space or Accessibility Graph



Example: State Space or Accessibility Graph


$$\begin{tikzcd}
\arrow[r] & \boxed{\begin{array}{l} P=1, req_P=0 \\ Q=1, req_Q=0 \end{array}} \arrow[d] \\
& \boxed{\begin{array}{l} P=2, req_P=1 \\ Q=1, req_Q=0 \end{array}} \arrow[d] \\
& \boxed{\begin{array}{l} P=3, req_P=1 \\ Q=1, req_Q=0 \end{array}} \arrow[d] \\
& \boxed{\begin{array}{l} P=4, req_P=1 \\ Q=1, req_Q=0 \end{array}} \arrow[uuu, bend left]
\end{tikzcd}$$

Example: State Space or Accessibility Graph

$$\begin{tikzcd}
{} \arrow[r] & \boxed{\begin{matrix} P=1, req_P=0 \\ Q=1, req_Q=0 \end{matrix}} \arrow[r] \arrow[d] & \boxed{\begin{matrix} P=1, req_P=0 \\ Q=2, req_Q=1 \end{matrix}} \arrow[r] & \boxed{\begin{matrix} P=1, req_P=0 \\ Q=3, req_Q=1 \end{matrix}} \arrow[r] & \boxed{\begin{matrix} P=1, req_P=0 \\ Q=4, req_Q=1 \end{matrix}} \arrow[lll, bend right] \\
& \boxed{\begin{matrix} P=2, req_P=1 \\ Q=1, req_Q=0 \end{matrix}} \arrow[d] & & & \\
& \boxed{\begin{matrix} P=3, req_P=1 \\ Q=1, req_Q=0 \end{matrix}} \arrow[d] & & & \\
& \boxed{\begin{matrix} P=4, req_P=1 \\ Q=1, req_Q=0 \end{matrix}} \arrow[uuu, bend left] & & &
\end{tikzcd}$$

Example: State Space or Accessibility Graph

$$\begin{tikzcd}
{} \arrow[r] & \boxed{\begin{matrix} P=1, req_P=0 \\ Q=1, req_Q=0 \end{matrix}} \arrow[r] \arrow[d] & \boxed{\begin{matrix} P=1, req_P=0 \\ Q=2, req_Q=1 \end{matrix}} \arrow[r] \arrow[d] & \boxed{\begin{matrix} P=1, req_P=0 \\ Q=3, req_Q=1 \end{matrix}} \arrow[r] \arrow[d] & \boxed{\begin{matrix} P=1, req_P=0 \\ Q=4, req_Q=1 \end{matrix}} \arrow[lll, bend right] \arrow[d] \\
& \boxed{\begin{matrix} P=2, req_P=1 \\ Q=1, req_Q=0 \end{matrix}} \arrow[r] \arrow[d] & \boxed{\begin{matrix} P=2, req_P=1 \\ Q=2, req_Q=1 \end{matrix}} & \boxed{\begin{matrix} P=2, req_P=1 \\ Q=3, req_Q=1 \end{matrix}} \arrow[r] & \boxed{\begin{matrix} P=2, req_P=1 \\ Q=4, req_Q=1 \end{matrix}} \arrow[lll, bend left] \\
& \boxed{\begin{matrix} P=3, req_P=1 \\ Q=1, req_Q=0 \end{matrix}} \arrow[r] \arrow[d] & \boxed{\begin{matrix} P=3, req_P=1 \\ Q=2, req_Q=1 \end{matrix}} \arrow[d] & & \\
& \boxed{\begin{matrix} P=4, req_P=1 \\ Q=1, req_Q=0 \end{matrix}} \arrow[r] \arrow[uuu, bend left] & \boxed{\begin{matrix} P=4, req_P=1 \\ Q=2, req_Q=1 \end{matrix}} \arrow[uuu, bend right] & &
\end{tikzcd}$$

Property 1

At any time, there is at most one process in CS.

Translation: there is no state with $P = 3$ and $Q = 3$.

It is true.

To check this property we need to explore the entire state space once.
We only need to know the set of states, not how they are connected.

Property 2

Any process requesting entrance to CS will eventually enter it.

Translation: any execution that visits a state with $P = 2$ should later visit a state with $P = 3$; likewise for $Q = 2$ and $Q = 3$.

It is false.

The state $\boxed{\begin{array}{l} P = 2, req_P = 1 \\ Q = 2, req_Q = 1 \end{array}}$ has no successor (it is a deadlock).

To check this property, we have to know the entire graph (states alone are not enough).

Property 3

The order of entrances into the CS follow the order of requests.

Translation: any execution path that sees a state with $P = 2 \wedge Q = 1$ should not visit any state with $Q = 3$ before visiting a state with $P = 3$ (+ symmetric property for Q).

It is true if we ignore the deadlock.

Same kind of verification as property 2.

A Formalization

- Represent the system using a finite **automaton**.
- Represent the property using a temporal logic formula.
- To compare these two objects, convert the temporal logic formula into an automaton.
- Some work on the two automata will tell us if they are "compatible".

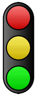
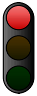
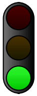
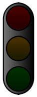
Propositional Logic: the Present

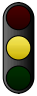
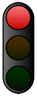
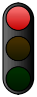
Propositional logic formulas can be use characterize one instant.

r: red light on

y: yellow light on

g: green light on

$r \wedge y \wedge g =$ , $r \wedge \neg y \wedge \neg g =$ , $\neg r \wedge \neg y \wedge g =$ , $\neg r \wedge \neg y \wedge \neg g =$ .

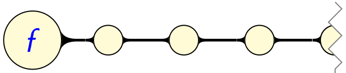
How can we say that  precedes  ?
How can we say that the system is not always  ?

$\Rightarrow$ we need to make time apparent in the formula

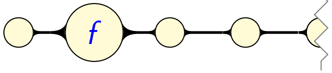
Linear-time Temporal Logic (LTL) Operators

Let f and g be two propositional logic formulas:

Present f

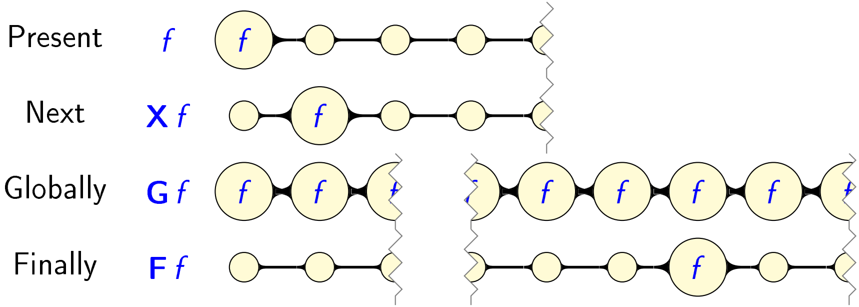


Next $\mathbf{X}\, f$



Linear-time Temporal Logic (LTL) Operators

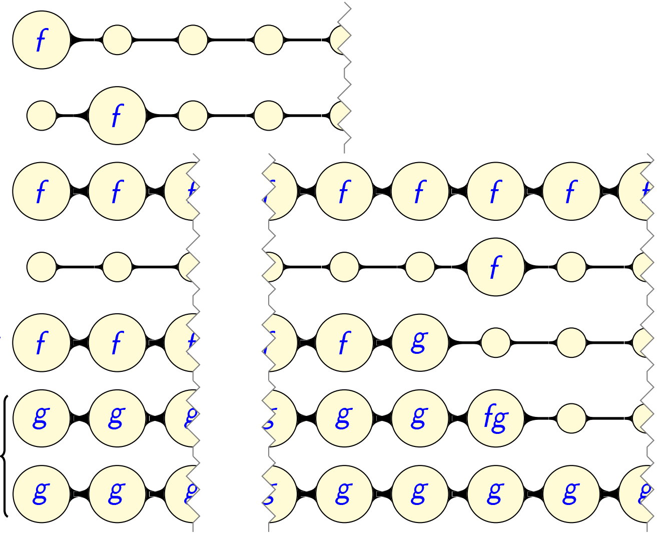
Let f and g be two propositional logic formulas:



Linear-time Temporal Logic (LTL) Operators

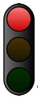
Let f and g be two propositional logic formulas:

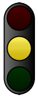
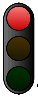
Present	f
Next	$\mathbf{X}\,f$
Globally	$\mathbf{G}\,f$
Finally	$\mathbf{F}\,f$
Until	$f\,\mathbf{U}\,g$
Releases	$f\,\mathbf{R}\,g$

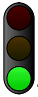


LTL: Examples

Next	$\mathbf{X}\,f$	f is true at next instant
Globally	$\mathbf{G}\,f$	f it true at all instants
Finally	$\mathbf{F}\,f$	f will be true eventually (now or in the future)
Until	$f\,\mathbf{U}\,g$	f stays true until g becomes true

$\neg\,\mathbf{G}(r \land \neg y \land \neg g)$: the system is not always .

$\mathbf{G}((\neg r \land y \land \neg g) \rightarrow \mathbf{X}(r \land \neg y \land \neg g))$:  always imm. flw'd by .

$\mathbf{G}\,\mathbf{F}(\neg r \land \neg y \land g)$: the systems is infinitely often .

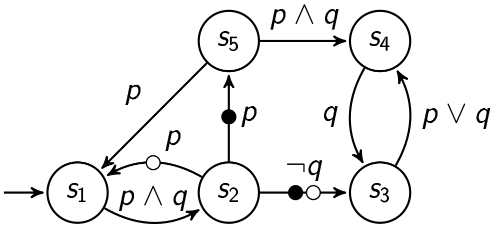
These formulas can be translated into automata.

TGBA: Generalized Büchi Automata

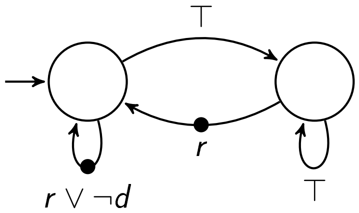
A Transition-based Generalized Büchi Automata has:

- a set of states, with a designated *initial* state,
- a set of transitions between these states, labeled by propositional logic formulas,
- a set of sets of transitions, called acceptance sets.

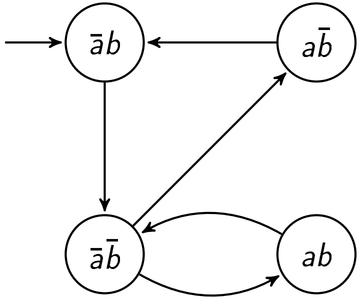
An infinite path in this automaton is accepted if it visits infinitely often a transition for each acceptance sets.



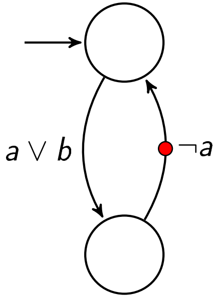
Example of TGBA for $\mathbf{G}(d \rightarrow \mathbf{F}\, r)$



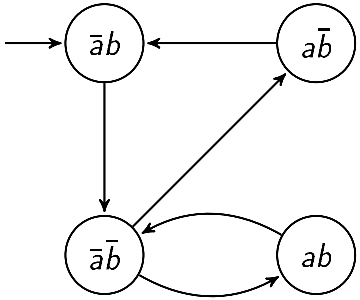
Product between a System and a TGBA



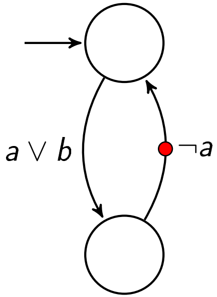
$\otimes$

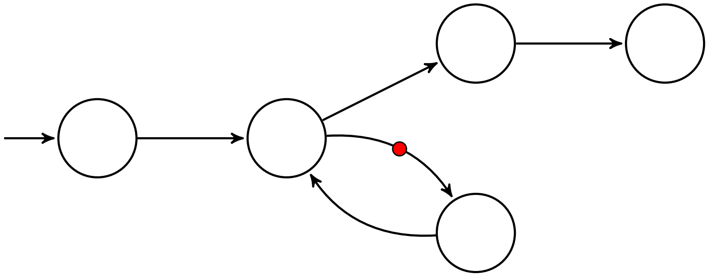


Product between a System and a TGBA



$\otimes$





Automata Theoretic Approach to Model Checking

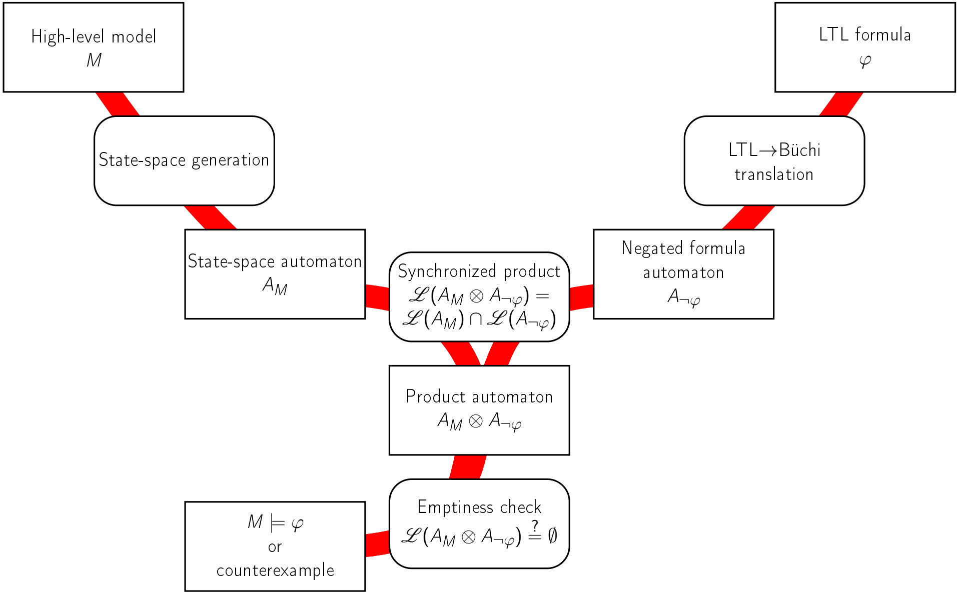

High-level model M
State-space generation
State-space automaton A_M
LTL formula φ
LTL→Büchi translation
Negated formula automaton $A_{\neg\varphi}$
Synchronized product $\mathscr{L}(A_M \otimes A_{\neg\varphi}) = \mathscr{L}(A_M) \cap \mathscr{L}(A_{\neg\varphi})$
Product automaton $A_M \otimes A_{\neg\varphi}$
Emptiness check $\mathscr{L}(A_M \otimes A_{\neg\varphi}) \overset{?}{=} \emptyset$
$M \models \varphi$ or counterexample

Tableau Rules for Propositional Logic

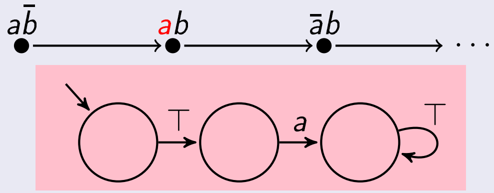
formula set	1st child	2nd child
$\Gamma \cup \{\neg\top\}$	$\Gamma \cup \{\bot\}$	
$\Gamma \cup \{\neg\bot\}$	$\Gamma \cup \{\top\}$	
$\Gamma \cup \{\neg\neg f\}$	$\Gamma \cup \{f\}$	
$\Gamma \cup \{f \wedge g\}$	$\Gamma \cup \{f, g\}$	
$\Gamma \cup \{f \vee g\}$	$\Gamma \cup \{f\}$	$\Gamma \cup \{g\}$
$\Gamma \cup \{\neg(f \wedge g)\}$	$\Gamma \cup \{\neg f\}$	$\Gamma \cup \{\neg g\}$
$\Gamma \cup \{\neg(f \vee g)\}$	$\Gamma \cup \{\neg f, \neg g\}$	

Tableau for $\neg\varphi$ with $\varphi = \neg(\neg A \vee B) \vee (\neg(A \wedge C) \vee (B \wedge C))$

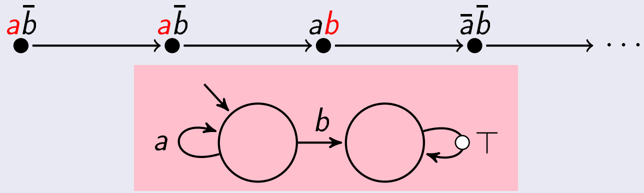
$$\begin{tikzcd}[column sep=tiny]
 & \{\neg(\neg(\neg A \vee B) \vee (\neg(A \wedge C) \vee (B \wedge C)))\} \arrow[d] & & \\
 & \{\neg\neg(\neg A \vee B), \neg(\neg(A \wedge C) \vee (B \wedge C))\} \arrow[d] & & \\
 & \{\neg A \vee B, \neg(\neg(A \wedge C) \vee (B \wedge C))\} \arrow[d] & & \\
 & \{\neg A \vee B, \neg\neg(A \wedge C), \neg(B \wedge C)\} \arrow[d] & & \\
 & \{\neg A \vee B, A \wedge C, \neg(B \wedge C)\} \arrow[d] & & \\
 & \{\neg A \vee B, A, C, \neg(B \wedge C)\} \arrow[dl] \arrow[dr] & & \\
\{\textcolor{red}{\neg A}, \textcolor{red}{A}, C, \neg(B \wedge C)\} & & \{B, A, C, \neg(B \wedge C)\} \arrow[dl] \arrow[dr] & \\
 & \{\textcolor{red}{B}, A, C, \textcolor{red}{\neg B}\} & & \{B, A, \textcolor{red}{C}, \textcolor{red}{\neg C}\}
\end{tikzcd}$$

LTL and Automata

$\mathbf{X}\,a$



$a\,\mathbf{U}\,b$



$$a\,\mathbf{U}\,b \equiv b \vee (a \wedge \mathbf{X}(a\,\mathbf{U}\,b))$$

Tableau Rules for Propositional Logic

formula set	1st child	2nd child
$\Gamma \cup \{\neg\top\}$	$\Gamma \cup \{\bot\}$	
$\Gamma \cup \{\neg\bot\}$	$\Gamma \cup \{\top\}$	
$\Gamma \cup \{\neg\neg f\}$	$\Gamma \cup \{f\}$	
$\Gamma \cup \{f \land g\}$	$\Gamma \cup \{f, g\}$	
$\Gamma \cup \{f \lor g\}$	$\Gamma \cup \{f\}$	$\Gamma \cup \{g\}$
$\Gamma \cup \{\neg(f \land g)\}$	$\Gamma \cup \{\neg f\}$	$\Gamma \cup \{\neg g\}$
$\Gamma \cup \{\neg(f \lor g)\}$	$\Gamma \cup \{\neg f, \neg g\}$	

Tableau Rules for Propositional Logic

formula set	1st child	2nd child
$\Gamma \cup \{\neg\top\}$	$\Gamma \cup \{\bot\}$	
$\Gamma \cup \{\neg\bot\}$	$\Gamma \cup \{\top\}$	
$\Gamma \cup \{\neg\neg f\}$	$\Gamma \cup \{f\}$	
$\Gamma \cup \{f \wedge g\}$	$\Gamma \cup \{f, g\}$	
$\Gamma \cup \{f \vee g\}$	$\Gamma \cup \{f\}$	$\Gamma \cup \{g\}$
$\Gamma \cup \{\neg(f \wedge g)\}$	$\Gamma \cup \{\neg f\}$	$\Gamma \cup \{\neg g\}$
$\Gamma \cup \{\neg(f \vee g)\}$	$\Gamma \cup \{\neg f, \neg g\}$	
$\Gamma \cup \{\neg \mathbf{X} f\}$	$\Gamma \cup \{\mathbf{X} \neg f\}$	
$\Gamma \cup \{f \mathbf{U} g\}$	$\Gamma \cup \{g\}$	$\Gamma \cup \{f, \mathbf{X}(f \mathbf{U} g), \mathsf{P}\, g\}$
$\Gamma \cup \{\neg(f \mathbf{U} g)\}$	$\Gamma \cup \{\neg f, \neg g\}$	$\Gamma \cup \{\neg g, \mathbf{X} \neg(f \mathbf{U} g)\}$

$\mathsf{P}\, g$ is a promise that g will be fulfilled

Tableau for $(\mathbf{X}\, a) \wedge (b\, \mathbf{U}\, \neg a)$

$\{(\mathbf{X}\, a) \wedge (b\, \mathbf{U}\, \neg a)\}$

Règles de tableau

formula set	1st child	2nd child
$\Gamma \cup \{f \wedge g\}$	$\Gamma \cup \{f, g\}$	
$\Gamma \cup \{f \vee g\}$	$\Gamma \cup \{f\}$	$\Gamma \cup \{g\}$
$\Gamma \cup \{f\, \mathbf{U}\, g\}$	$\Gamma \cup \{g\}$	$\Gamma \cup \{f, \mathbf{X}(f\, \mathbf{U}\, g), \mathrm{P}\, g\}$
$\vdots$	$\vdots$	$\vdots$

Tableau for $(\mathbf{X}\, a) \wedge (b\, \mathbf{U}\, \neg a)$

$$\begin{tikzcd}
 & \{(\mathbf{X}\, a) \wedge (b\, \mathbf{U}\, \neg a)\} \arrow[d] & \\
 & \{\mathbf{X}\, a, b\, \mathbf{U}\, \neg a\} \arrow[dl] \arrow[dr] & \\
\{\mathbf{X}\, a,\ \neg a\} & & \{\mathbf{X}\, a,\ b,\ \mathbf{X}(b\, \mathbf{U}\, \neg a),\ \mathrm{P}\, \neg a\}
\end{tikzcd}$$

Règles de tableau

formula set	1st child	2nd child
$\Gamma \cup \{f \wedge g\}$	$\Gamma \cup \{f, g\}$	
$\Gamma \cup \{f \vee g\}$	$\Gamma \cup \{f\}$	$\Gamma \cup \{g\}$
$\Gamma \cup \{f\, \mathbf{U}\, g\}$	$\Gamma \cup \{g\}$	$\Gamma \cup \{f, \mathbf{X}(f\, \mathbf{U}\, g), \mathrm{P}\, g\}$
⋮	⋮	⋮

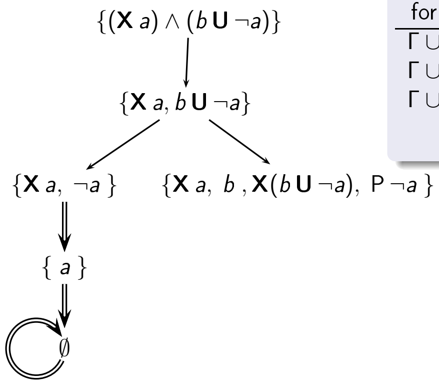
Tableau for $(\mathbf{X}\, a) \land (b\, \mathbf{U}\, \neg a)$

$$\begin{tikzcd}[column sep=small]
& \{(\mathbf{X}\, a) \land (b\, \mathbf{U}\, \neg a)\} \arrow[d] & \\
& \{\mathbf{X}\, a, b\, \mathbf{U}\, \neg a\} \arrow[dl] \arrow[dr] & \\
\{\mathbf{X}\, a, \neg a\} \arrow[d, Rightarrow] & & \{\mathbf{X}\, a, b, \mathbf{X}(b\, \mathbf{U}\, \neg a), \mathrm{P}\, \neg a\} \\
\{a\} & &
\end{tikzcd}$$

Règles de tableau

formula set	1st child	2nd child
$\Gamma \cup \{f \land g\}$	$\Gamma \cup \{f, g\}$	
$\Gamma \cup \{f \lor g\}$	$\Gamma \cup \{f\}$	$\Gamma \cup \{g\}$
$\Gamma \cup \{f\, \mathbf{U}\, g\}$	$\Gamma \cup \{g\}$	$\Gamma \cup \{f, \mathbf{X}(f\, \mathbf{U}\, g), \mathrm{P}\, g\}$
$\vdots$	$\vdots$	$\vdots$

Tableau for $(\mathbf{X}\, a) \wedge (b\, \mathbf{U}\, \neg a)$

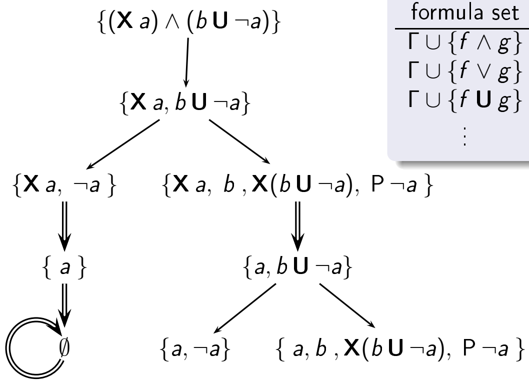


$$\begin{tikzcd}
& \{(\mathbf{X}\, a) \wedge (b\, \mathbf{U}\, \neg a)\} \arrow[d] & \\
& \{\mathbf{X}\, a, b\, \mathbf{U}\, \neg a\} \arrow[dl] \arrow[dr] & \\
\{\mathbf{X}\, a, \neg a\} \arrow[d, Rightarrow] & & \{\mathbf{X}\, a, b, \mathbf{X}(b\, \mathbf{U}\, \neg a), \mathrm{P}\, \neg a\} \\
\{a\} \arrow[d, Rightarrow] & & \\
\emptyset \arrow[loop left, Rightarrow] & &
\end{tikzcd}$$

Règles de tableau

formula set	1st child	2nd child
$\Gamma \cup \{f \wedge g\}$	$\Gamma \cup \{f, g\}$	
$\Gamma \cup \{f \vee g\}$	$\Gamma \cup \{f\}$	$\Gamma \cup \{g\}$
$\Gamma \cup \{f\, \mathbf{U}\, g\}$	$\Gamma \cup \{g\}$	$\Gamma \cup \{f, \mathbf{X}(f\, \mathbf{U}\, g), \mathrm{P}\, g\}$
$\vdots$	$\vdots$	$\vdots$

Tableau for $(\mathbf{X}\, a) \wedge (b\, \mathbf{U}\, \neg a)$



$$\begin{tikzcd}
 & \{(\mathbf{X}\, a) \wedge (b\, \mathbf{U}\, \neg a)\} \arrow[d] & & \\
 & \{\mathbf{X}\, a, b\, \mathbf{U}\, \neg a\} \arrow[dl] \arrow[dr] & & \\
\{\mathbf{X}\, a, \neg a\} \arrow[d, Rightarrow] & & \{\mathbf{X}\, a, b, \mathbf{X}(b\, \mathbf{U}\, \neg a), \mathrm{P}\, \neg a\} \arrow[d, Rightarrow] & \\
\{a\} \arrow[d, Rightarrow] & & \{a, b\, \mathbf{U}\, \neg a\} \arrow[dl] \arrow[dr] & \\
\emptyset \arrow[loop left, Rightarrow] & \{a, \neg a\} & & \{a, b, \mathbf{X}(b\, \mathbf{U}\, \neg a), \mathrm{P}\, \neg a\}
\end{tikzcd}$$

Règles de tableau

formula set	1st child	2nd child
$\Gamma \cup \{f \wedge g\}$	$\Gamma \cup \{f, g\}$	
$\Gamma \cup \{f \vee g\}$	$\Gamma \cup \{f\}$	$\Gamma \cup \{g\}$
$\Gamma \cup \{f\, \mathbf{U}\, g\}$	$\Gamma \cup \{g\}$	$\Gamma \cup \{f, \mathbf{X}(f\, \mathbf{U}\, g), \mathrm{P}\, g\}$
$\vdots$	$\vdots$	$\vdots$

Tableau for $(\mathbf{X}\, a) \land (b\, \mathbf{U}\, \neg a)$

Règles de tableau

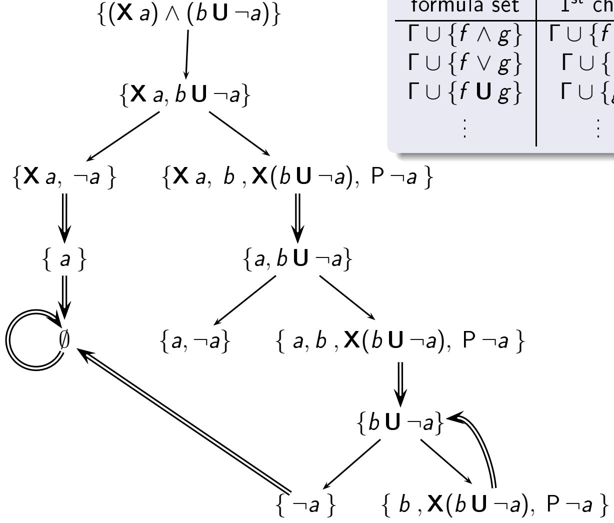
formula set	1st child	2nd child
$\Gamma \cup \{f \land g\}$	$\Gamma \cup \{f, g\}$	
$\Gamma \cup \{f \lor g\}$	$\Gamma \cup \{f\}$	$\Gamma \cup \{g\}$
$\Gamma \cup \{f\, \mathbf{U}\, g\}$	$\Gamma \cup \{g\}$	$\Gamma \cup \{f, \mathbf{X}(f\, \mathbf{U}\, g), \mathrm{P}\, g\}$
$\vdots$	$\vdots$	$\vdots$

$$\begin{tikzcd}
 & \{(\mathbf{X}\, a) \land (b\, \mathbf{U}\, \neg a)\} \arrow[d] & & & \\
 & \{\mathbf{X}\, a, b\, \mathbf{U}\, \neg a\} \arrow[dl] \arrow[dr] & & & \\
\{\mathbf{X}\, a, \neg a\} \arrow[d, Rightarrow] & & \{\mathbf{X}\, a, b, \mathbf{X}(b\, \mathbf{U}\, \neg a), \mathrm{P}\, \neg a\} \arrow[d, Rightarrow] & & \\
\{a\} \arrow[d, Rightarrow] & & \{a, b\, \mathbf{U}\, \neg a\} \arrow[dl] \arrow[dr] & & \\
\emptyset \arrow[loop left, Rightarrow] & \{a, \neg a\} & & \{a, b, \mathbf{X}(b\, \mathbf{U}\, \neg a), \mathrm{P}\, \neg a\} \arrow[d, Rightarrow] & \\
 & & & \{b\, \mathbf{U}\, \neg a\} \arrow[dl] \arrow[dr] & \\
 & & \{\neg a\} & & \{b, \mathbf{X}(b\, \mathbf{U}\, \neg a), \mathrm{P}\, \neg a\}
\end{tikzcd}$$

Tableau for $(\mathbf{X}\, a) \land (b\, \mathbf{U} \neg a)$

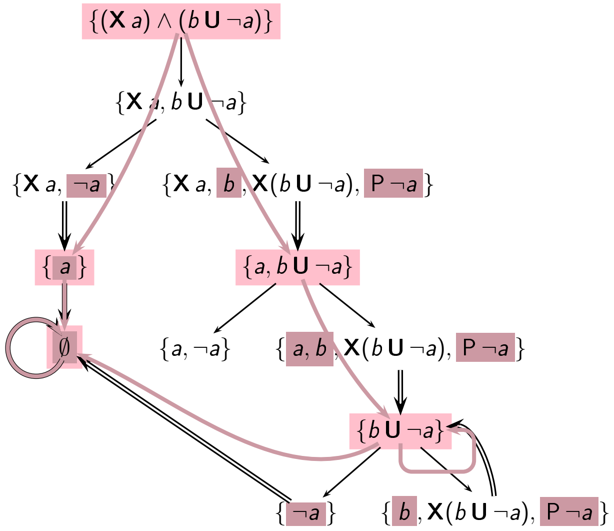
Règles de tableau

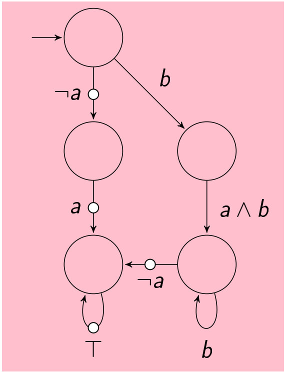
formula set	1st child	2nd child
$\Gamma \cup \{f \land g\}$	$\Gamma \cup \{f, g\}$	
$\Gamma \cup \{f \lor g\}$	$\Gamma \cup \{f\}$	$\Gamma \cup \{g\}$
$\Gamma \cup \{f\, \mathbf{U}\, g\}$	$\Gamma \cup \{g\}$	$\Gamma \cup \{f, \mathbf{X}(f\, \mathbf{U}\, g), \mathrm{P}\, g\}$
$\vdots$	$\vdots$	$\vdots$



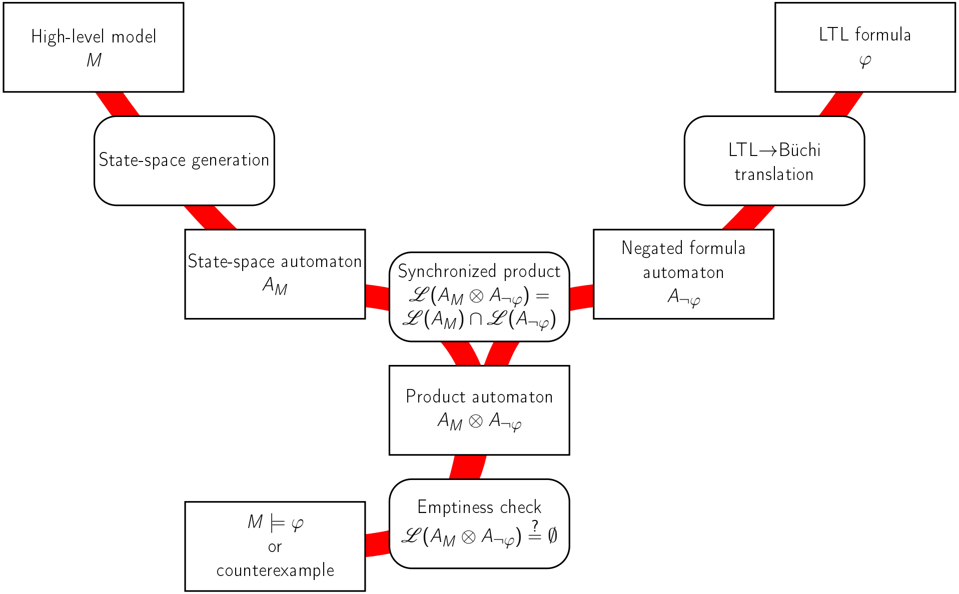
$$\begin{tikzcd}
 & \{(\mathbf{X}\, a) \land (b\, \mathbf{U} \neg a)\} \arrow[d] & & & \\
 & \{\mathbf{X}\, a, b\, \mathbf{U} \neg a\} \arrow[dl] \arrow[dr] & & & \\
\{\mathbf{X}\, a, \neg a\} \arrow[d, Rightarrow] & & \{\mathbf{X}\, a, b, \mathbf{X}(b\, \mathbf{U} \neg a), \mathrm{P} \neg a\} \arrow[d, Rightarrow] & & \\
\{a\} \arrow[d, Rightarrow] & & \{a, b\, \mathbf{U} \neg a\} \arrow[dl] \arrow[dr] & & \\
\emptyset \arrow[loop left, Rightarrow] & \{a, \neg a\} & & \{a, b, \mathbf{X}(b\, \mathbf{U} \neg a), \mathrm{P} \neg a\} \arrow[d, Rightarrow] & \\
 & & & \{b\, \mathbf{U} \neg a\} \arrow[dl] \arrow[dr] & \\
 & & \{\neg a\} \arrow[uull, Rightarrow] & & \{b, \mathbf{X}(b\, \mathbf{U} \neg a), \mathrm{P} \neg a\} \arrow[ul, Rightarrow, bend right]
\end{tikzcd}$$

$(\mathbf{X}\, a) \wedge (b\, \mathbf{U} \neg a)$ into TGBA





Automata Theoretic Approach to Model Checking



High-level model M
State-space generation
State-space automaton A_M
LTL formula φ
LTL→Büchi translation
Negated formula automaton $A_{\neg\varphi}$
Synchronized product $\mathscr{L}(A_M \otimes A_{\neg\varphi}) = \mathscr{L}(A_M) \cap \mathscr{L}(A_{\neg\varphi})$
Product automaton $A_M \otimes A_{\neg\varphi}$
Emptiness check $\mathscr{L}(A_M \otimes A_{\neg\varphi}) \stackrel{?}{=} \emptyset$
$M \models \varphi$ or counterexample

Ex.: clients/server with synchronized automata

$$\begin{tikzcd} {} \arrow[r] & 1 \arrow[r, bend left, "s"] & 2 \arrow[l, bend left, "r"] \end{tikzcd}$$

Client C

$$\begin{tikzcd} 2 \arrow[r, bend right, "s_1"'] & 1 \arrow[l, bend right, "r_1"'] \arrow[r, bend left, "r_2"] & 3 \arrow[l, bend left, "s_2"] \end{tikzcd}$$

Server S

$$\begin{tikzcd} {} \arrow[r] & - \arrow[r, bend left, "a"] & \times \arrow[l, bend left, "d"] \end{tikzcd}$$

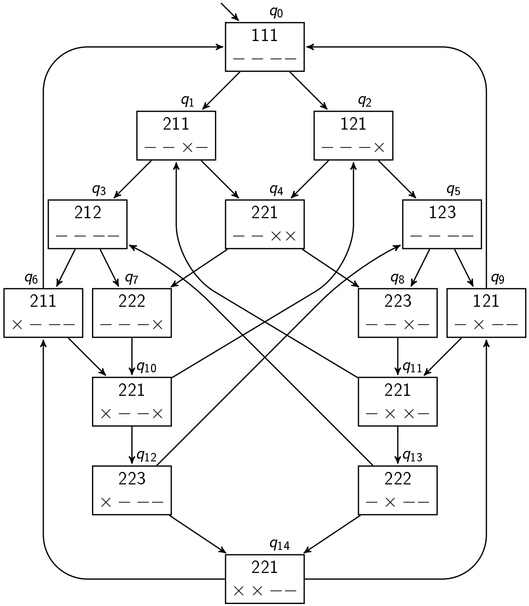
Channel B

Synchronization rules for the system $\langle C, C, S, B, B, B, B\rangle$:

$$\begin{array}{ll}
(1) & \langle\ s\ ,\ .\ ,\ .\ ,\ .\ ,\ .\ ,\ a\ ,\ .\ \rangle \\
(2) & \langle\ .\ ,\ s\ ,\ .\ ,\ .\ ,\ .\ ,\ .\ ,\ a\ \rangle \\
(3) & \langle\ r\ ,\ .\ ,\ .\ ,\ d\ ,\ .\ ,\ .\ ,\ .\ \rangle \\
(4) & \langle\ .\ ,\ r\ ,\ .\ ,\ .\ ,\ d\ ,\ .\ ,\ .\ \rangle \\
(5) & \langle\ .\ ,\ .\ ,\ r_1\ ,\ .\ ,\ .\ ,\ d\ ,\ .\ \rangle \\
(6) & \langle\ .\ ,\ .\ ,\ s_1\ ,\ a\ ,\ .\ ,\ .\ ,\ .\ \rangle \\
(7) & \langle\ .\ ,\ .\ ,\ r_2\ ,\ .\ ,\ .\ ,\ .\ ,\ d\ \rangle \\
(8) & \langle\ .\ ,\ .\ ,\ s_2\ ,\ .\ ,\ a\ ,\ .\ ,\ .\ \rangle
\end{array}$$

If a client sends a request, will he always get an answer?

State space for this example



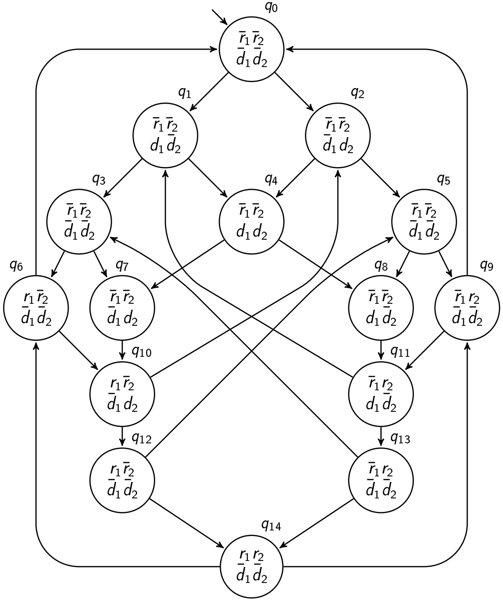
Atomic properties

We will write properties regarding sending and receiving messages:
Let $AP = \{a_1, a_2, r_1, r_2\}$ with:

- a_1: an answer is on its way between S and C_1
- a_2: an answer is on its way between S and C_2
- r_1: a request is on its way between C_1 and S
- r_2: a request is on its way between C_2 and S

The property "if a client sends a request, he will get an answer" can be rewritten as "$\forall i \in \{1, 2\}$ an execution that visits a state where r_i is true will visit a state where a_i is true."

Kripke Structure for this example


q_0 $\bar{r}_1\bar{r}_2$ $\bar{d}_1\bar{d}_2$
q_1 $\bar{r}_1\bar{r}_2$ $d_1\bar{d}_2$
q_2 $\bar{r}_1\bar{r}_2$ $\bar{d}_1 d_2$
q_3 $\bar{r}_1\bar{r}_2$ $\bar{d}_1\bar{d}_2$
q_4 $\bar{r}_1\bar{r}_2$ $d_1 d_2$
q_5 $\bar{r}_1\bar{r}_2$ $\bar{d}_1\bar{d}_2$
q_6 $r_1\bar{r}_2$ $\bar{d}_1\bar{d}_2$
q_7 $\bar{r}_1\bar{r}_2$ $\bar{d}_1 d_2$
q_8 $\bar{r}_1\bar{r}_2$ $d_1\bar{d}_2$
q_9 $\bar{r}_1 r_2$ $\bar{d}_1\bar{d}_2$
q_{10} $r_1\bar{r}_2$ $\bar{d}_1 d_2$
q_{11} $\bar{r}_1 r_2$ $d_1\bar{d}_2$
q_{12} $r_1\bar{r}_2$ $\bar{d}_1\bar{d}_2$
q_{13} $\bar{r}_1 r_2$ $\bar{d}_1\bar{d}_2$
q_{14} $r_1 r_2$ $\bar{d}_1\bar{d}_2$

Exploring the Kripke Structure

"an execution that visits a state where r_i is true will visit a state where a_i is true." In LTL: $\mathbf{G}(r_i \rightarrow \mathbf{F}\, a_i)$.
(by symmetry on the model, let's deal only with $i = 1$).

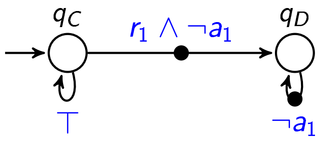
We are looking for a counterexample: an execution that visits a state where r_1 is true and which will never verify a_1 from then on. In LTL:
$\neg\, \mathbf{G}(r_1 \rightarrow \mathbf{F}\, a_1) = \mathbf{F}(r_1 \wedge \mathbf{G}\, \neg a_1)$

Exploring the Kripke Structure

“an execution that visits a state where r_i is true will visit a state where a_i is true.” In LTL: $\mathbf{G}(r_i \rightarrow \mathbf{F}\, a_i)$.
(by symmetry on the model, let's deal only with $i = 1$).

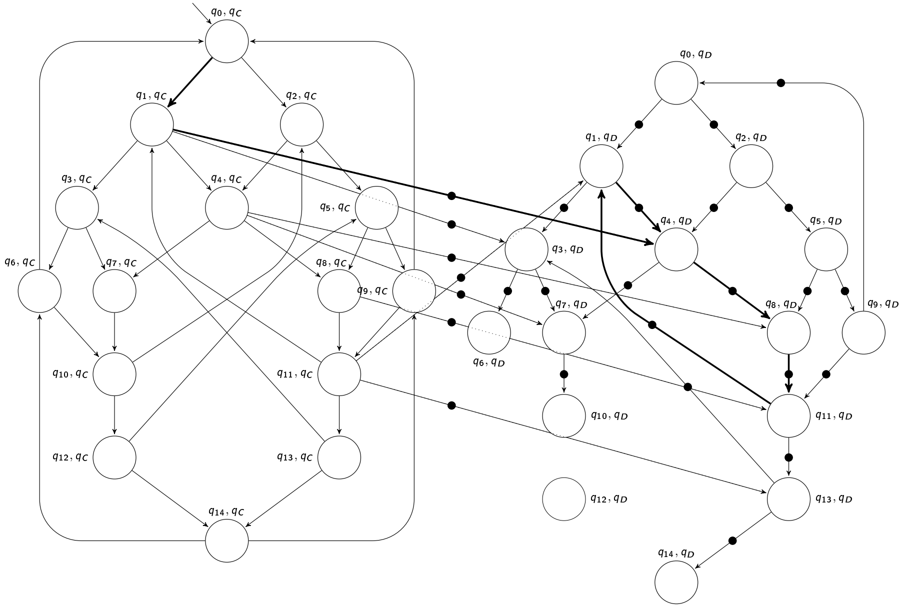
We are looking for a counterexample: an execution that visits a state where r_1 is true and which will never verify a_1 from then on. In LTL:
$\neg\,\mathbf{G}(r_1 \rightarrow \mathbf{F}\, a_1) = \mathbf{F}(r_1 \wedge \mathbf{G}\,\neg a_1)$

Such a counterexample can be represented by a (transition-based) Büchi automaton:

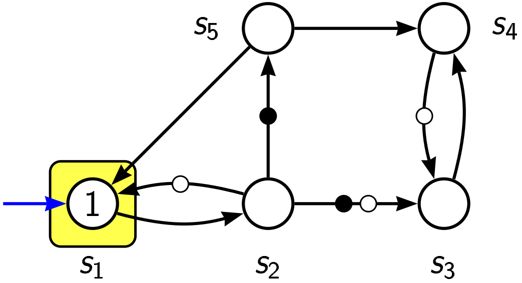


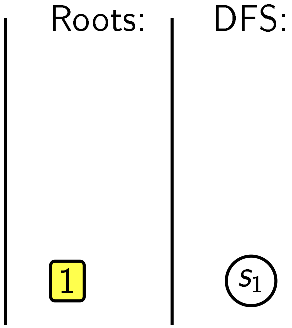
Where accepting runs must visit transitions with ● infinitely often.

Synchronized Product

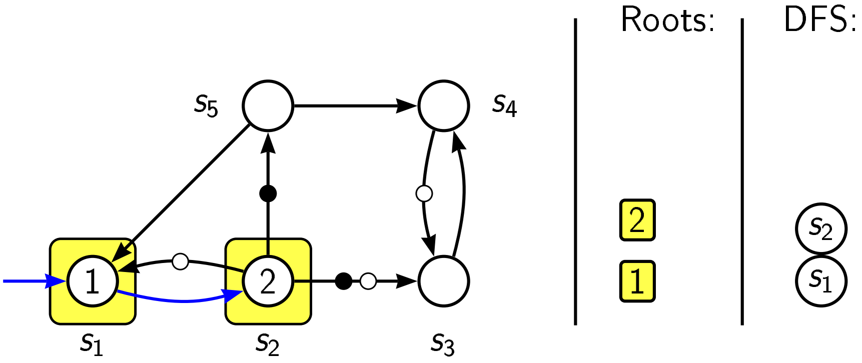


Emptiness check

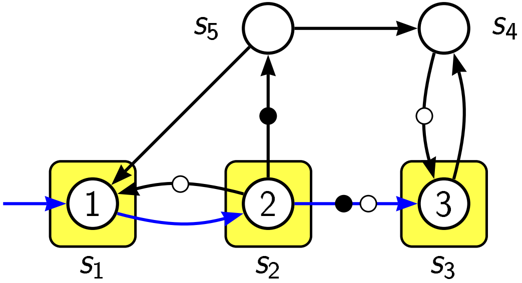


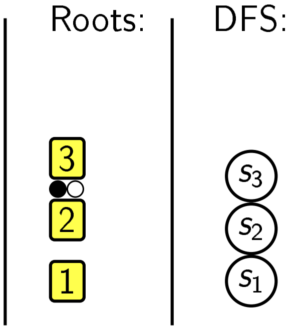


Emptiness check

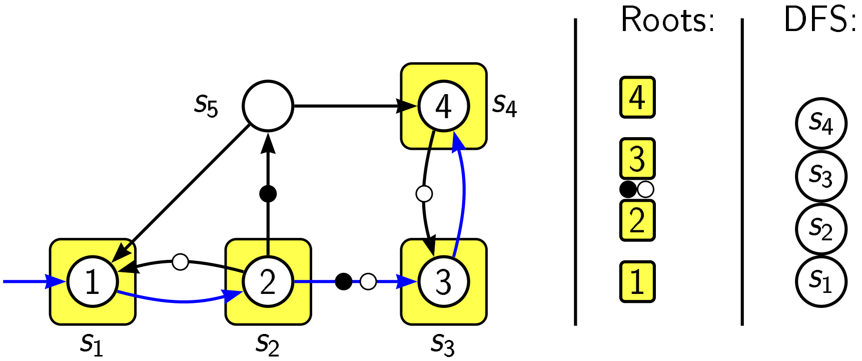


Emptiness check

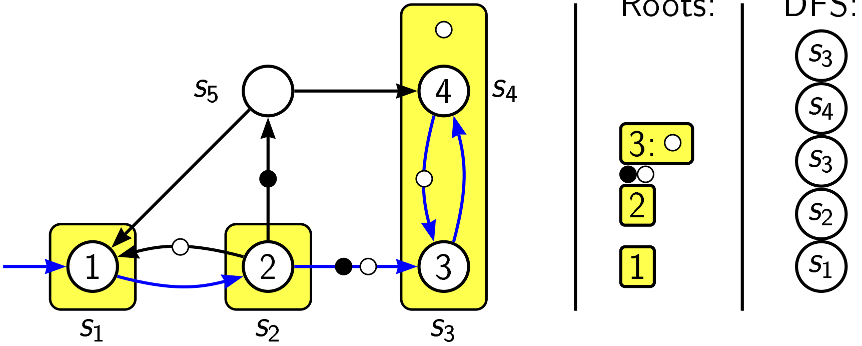




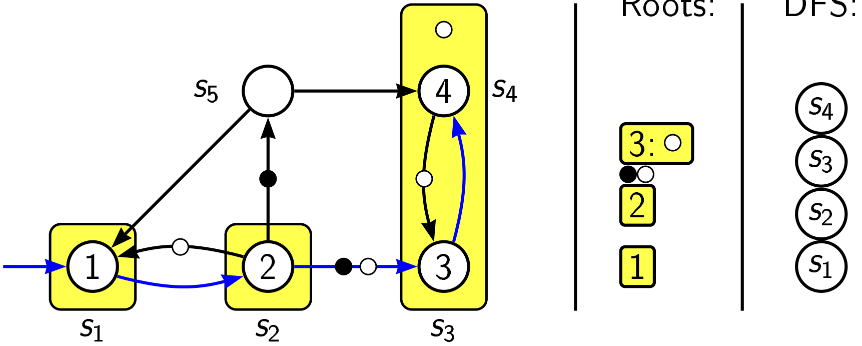
Emptiness check



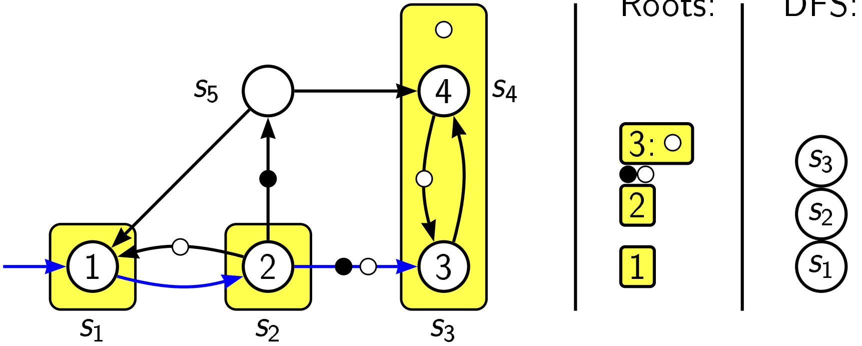
Emptiness check



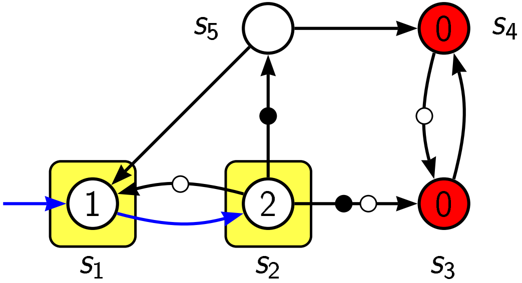
Emptiness check

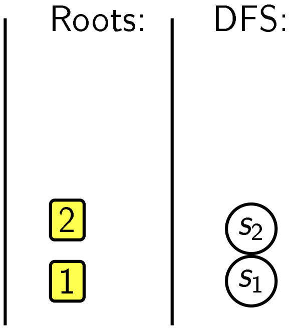


Emptiness check

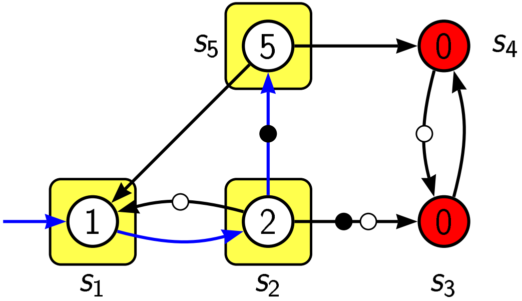


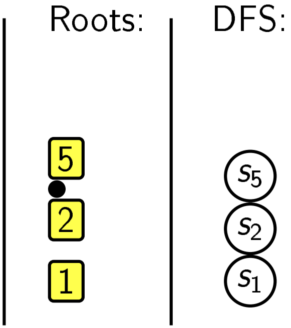
Emptiness check



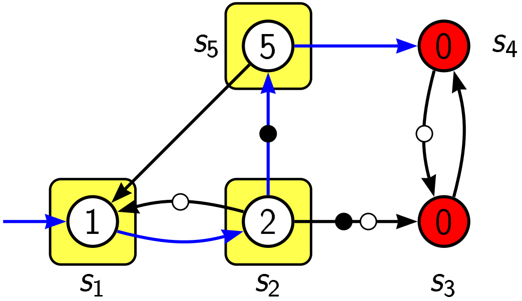


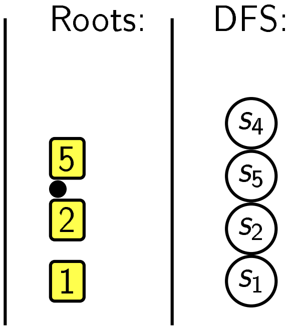
Emptiness check



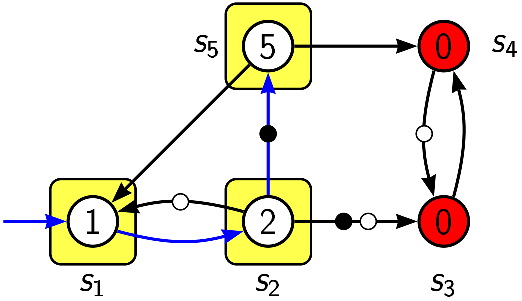


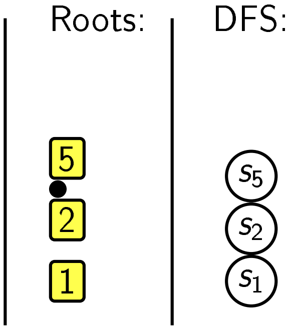
Emptiness check



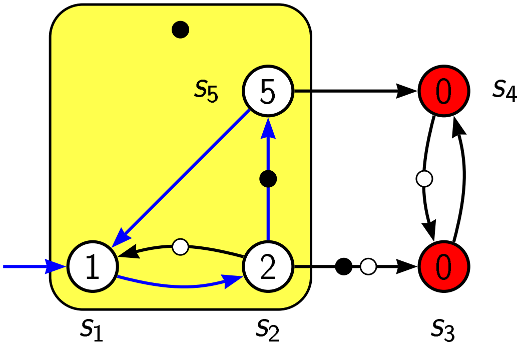


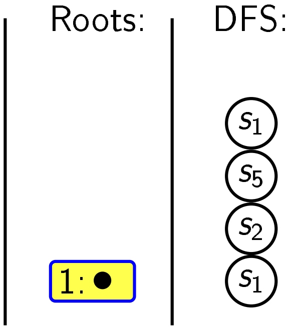
Emptiness check



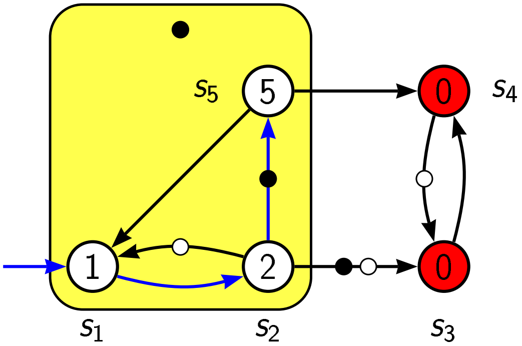


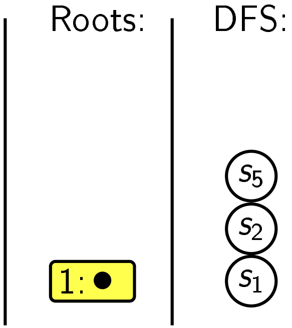
Emptiness check



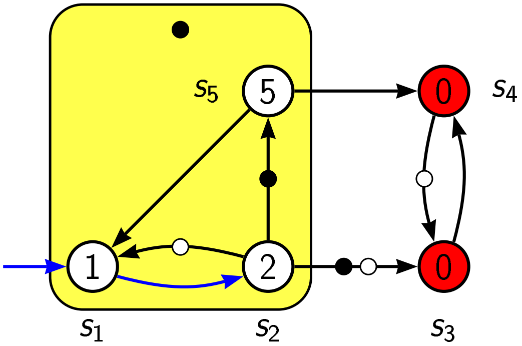


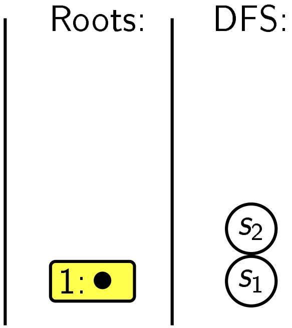
Emptiness check



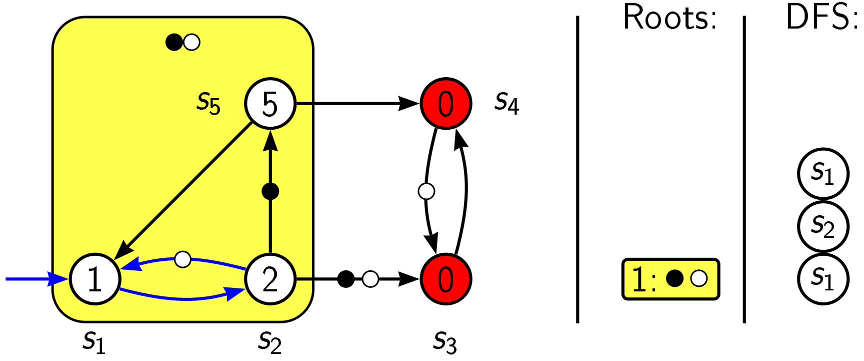


Emptiness check



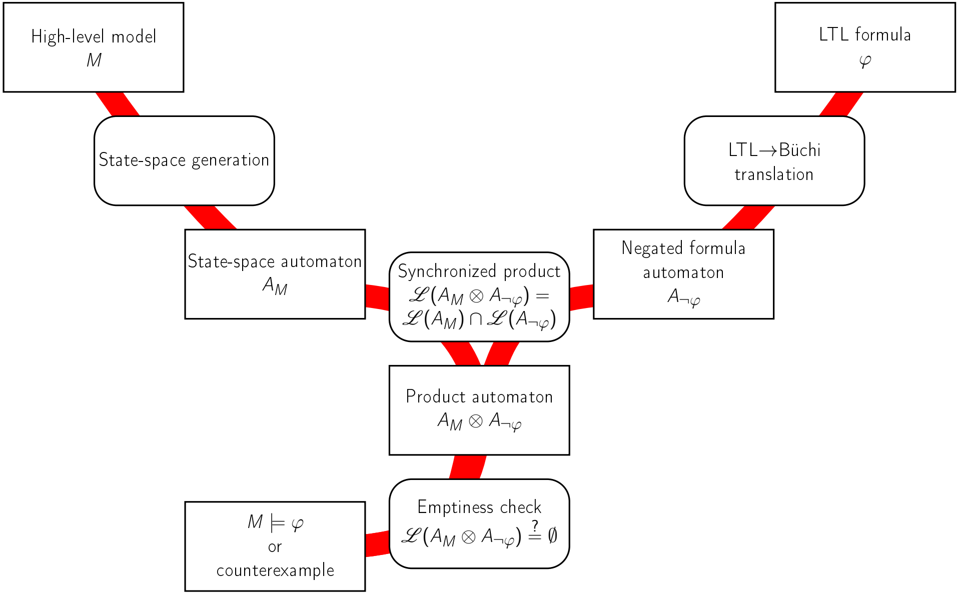


Emptiness check



Found!

Automata Theoretic Approach to Model Checking


High-level model M
State-space generation
State-space automaton A_M
LTL formula φ
LTL→Büchi translation
Negated formula automaton $A_{\neg\varphi}$
Synchronized product $\mathscr{L}(A_M \otimes A_{\neg\varphi}) = \mathscr{L}(A_M) \cap \mathscr{L}(A_{\neg\varphi})$
Product automaton $A_M \otimes A_{\neg\varphi}$
Emptiness check $\mathscr{L}(A_M \otimes A_{\neg\varphi}) \overset{?}{=} \emptyset$
$M \models \varphi$ or counterexample

Conclusion

- Büchi automata can be used to represent sets (finite or infinite) of infinite behaviors. Some operations are easy to perform on these sets: union, intersection, and emptiness check. Some are harder (e.g. complementation, universality check)
- By reducing the verification problem to some operations between automata, we actually obtained an efficient verification procedure.
- Bottleneck: translating a formula of size n can lead to a TGBA of size $2^{\mathrm{O}(n)}$. The size of the product of two automata is bounded by the product of the sizes, so it is important to have small automata on both sides. Emptiness check is linear in the size of the product.

For CSE students: the automata seen in ToC are simpler because they recognize finite words. Yet they allow similar operations and applications.