

# Model-checking LTL specifications

(IF311 : "Formal methods in software design")

Frédéric Herbreteau (fh@labri.fr)

Bordeaux INP



LABORATOIRE  
BORDELAIS  
DE RECHERCHE  
EN INFORMATIQUE

**LaBRI**

# What is model-checking?

INPUT:

- ▶ A transition system  $M$
- ▶ An LTL formula  $\Phi$

Determine whether:

$$M \models \Phi$$

→ check that **every infinite run** in  $M$  satisfies  $\Phi$ .

# LTL formulas as languages (1/2)

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

- ▶  $\llbracket a \rrbracket = \{A \in \Sigma \mid a \in A\} \Sigma^\omega$
- ▶  $\llbracket \neg\alpha \rrbracket = \Sigma^\omega - \llbracket \alpha \rrbracket$
- ▶  $\llbracket \alpha \vee \beta \rrbracket = \llbracket \alpha \rrbracket \cup \llbracket \beta \rrbracket$
- ▶  $\llbracket X\alpha \rrbracket = \{w \in \Sigma^\omega \mid w_{1\dots} \in \llbracket \alpha \rrbracket\} = \Sigma \llbracket \alpha \rrbracket$
- ▶  $\llbracket \alpha U \beta \rrbracket = \{w \in \Sigma^\omega \mid \exists i. w_{i\dots} \in \llbracket \beta \rrbracket, \forall j < i. w_{j\dots} \in \llbracket \alpha \rrbracket\}$

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

# LTL formulas as languages (2/2)

## Example

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

▶  $\llbracket a \rrbracket =$

▶  $\llbracket \neg a \rrbracket =$

▶  $\llbracket a \vee b \rrbracket =$

▶  $\llbracket Xa \rrbracket =$

▶  $\llbracket aU b \rrbracket =$

# Satisfaction as a automaton problem

- ▶  $\text{trace}(\pi) = L(s_0)L(s_1)\cdots L(s_i)\cdots$  for  $\pi = s_0s_1\cdots s_i\cdots$
- ▶  $\text{trace}(M) = \{\text{trace}(\pi) \mid \pi \text{ infinite run of } M\}$

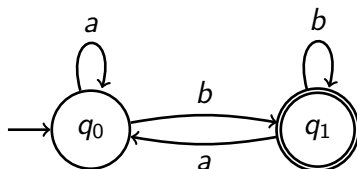
Then:

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

where  $\mathcal{A}_{\neg\Phi}$  is an **automaton** that accepts  $\llbracket \neg\Phi \rrbracket$  and  $\otimes$  is a product operation (see later)

# From LTL formulas to automata

# Automata on finite words

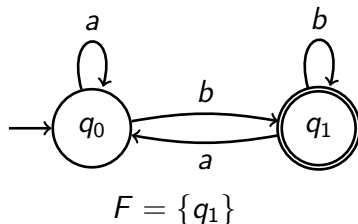


A word  $w$  is accepted if  $q_0 \xrightarrow{w} q$  with  $q \in F$ .

## Example

- ▶  $aabb$  is accepted
- ▶  $aba$  is not accepted

# Automata on infinite words (Büchi)



**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

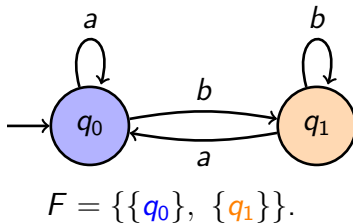
## Example

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.  $\{w \mid w \text{ contains finitely many } 1\text{'s}\}$
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)



**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  $w = A_0A_1 \cdots A_i$  that satisfies  $\Phi$

Let  $B_i$  be the set of **subformulas of  $\Phi$  that are true on the suffix of  $w$  from  $i$** :  $\Psi \in B_i$  iff  $w_{i\dots} \in \llbracket \Psi \rrbracket$

## Example

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

$w$	$\{a, c\}$	$\{a\}$	$\{a, c\}$	$\{b\}$	$\{b\}$	$\cdots$
$B$	$\{a, a \cup b\}$	$\{a, a \cup b\}$	$\{a, a \cup b\}$	$\{b, a \cup b\}$	$\{b, a \cup b\}$	$\cdots$

$w \in \llbracket a \cup b \rrbracket$  since  $(a \cup b) \in B_0$

# Principle of automata construction (a)

## Example

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

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

$w \in \llbracket a \rrbracket$  since  $a \in B_0$

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

$w \notin \llbracket a \rrbracket$  since  $a \notin B_0$

# Principle of automata construction (X)

## Example

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

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

$w : \{c\}$	$\{a, b\}$	$\{b, c\}$	$\{a, c\}$	$\dots$
$B : \{Xa\}$	$\{a\}$	$\{Xa\}$	$\{a\}$	$\dots$

$w \in \llbracket Xa \rrbracket$  since  $(Xa) \in B_0$  and consistent

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

$w \notin \llbracket Xa \rrbracket$  since no consistent  $B$  with  $(Xa) \in B_0$ .

# Principle of automata construction (U)

## Example

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

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

$w : \{a, c\}$	$\{a\}$	$\{b, c\}$	$\{a, c\}$	$\dots$
$B : \{a, aUb\}$	$\{a, aUb\}$	$\{b, aUb\}$	$\{a\}$	$\dots$

$w \in \llbracket aUb \rrbracket$  since  $(aUb) \in B_0$ , consistent (observe  $b \in B_2$ )

$w : \{a, c\}$	$\{c\}$	$\{b, c\}$	$\{a, c\}$	$\dots$
$B : \{a\}$	$\{c\}$	$\{b, aUb\}$	$\{a\}$	$\dots$

$w \notin \llbracket aUb \rrbracket$  since no consistent  $B$  with  $(aUb) \in B_0$

# From $\Phi$ 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  $\phi$ 
  - tell if each subformula is true or false

## Example

Let  $\Phi = aU(\neg a \wedge b)$

Subformulas of  $\Phi$ :  $\{a, b, \neg a, \neg a \wedge b, aU(\neg a \wedge b)\}$

$\{a, \neg a\}$  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:  $\emptyset$ ,  $\{a\}$ ,  $\{a, aU(\neg a \wedge b)\}$ ,  $\{a, b\}$ ,  $\{a, b, aU(\neg a \wedge b)\}$   
and  $\{b, \neg a \wedge b, aU(\neg a \wedge b)\}$ .

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

# From $\Phi$ 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  $X\phi \in B$  then  $\phi \in B'$

► if  $\phi_1 U \phi_2 \in B$  then  $\phi_2 \in B$ , or  $\phi_1 \in B$  and  $\phi_1 U \phi_2 \in B'$

Observe that:

$$\phi_1 U \phi_2 \equiv \phi_2 \vee (\phi_1 \wedge X(\phi_1 U \phi_2))$$



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

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

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

- $F$  is a **generalized Büchi** condition with **one accepting set in  $F_{\phi_1 U \phi_2}$  per subformula  $\phi_1 U \phi_2$  in  $\Phi$ .**

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

## Example

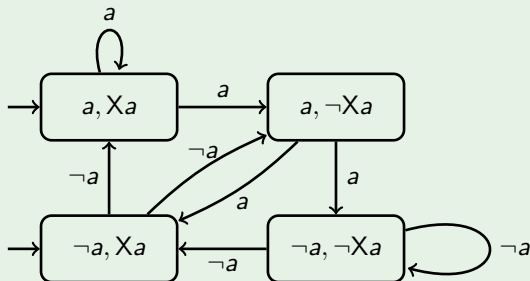
The automaton for  $aUb$  starts from initial state with  $aUb$ :

$$\begin{array}{ccccccc} w : \{a, c\} & & \{a\} & & \{a, c\} & & \dots \\ & & \{a, aUb\} & \rightarrow & \{a, aUb\} & \rightarrow & \{a, aUb\} \dots \end{array}$$

$w \notin \llbracket aUb \rrbracket$  since accepting set  $F_{aUb}$  not visited  $\infty$ -often

# Automaton for $\chi_a$

## Example (automaton for $\chi_a$ )



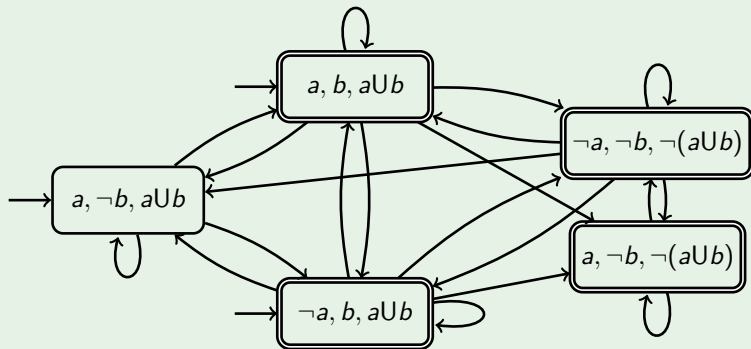
All runs are accepting since  $F = \emptyset$ .

Runs on:

- ▶  $\{a\}\{a\}\emptyset\{a\}\emptyset\{a\}\emptyset\dots$
- ▶  $\emptyset\emptyset\{a\}\{a\}\{a\}\{a\}\{a\}\dots$

# Automaton for $aUb$

## Example (automaton for $aUb$ )



Runs on:

- ▶  $\{a\}\{a\}\{b\}\emptyset\{a\}\emptyset\dots$
- ▶  $\{a\}\{a\}\{a\}\{a\}\{a\}\{a\}\dots$

# Efficient algorithms and implementations

- ▶ Our construction builds an automaton with  $2^{\mathcal{O}(\Phi)}$  **states and an alphabet of size  $2^{AP}$**  such that  $\mathcal{L}(\mathcal{A}_\Phi) = \llbracket \Phi \rrbracket$ .
- ▶ 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:

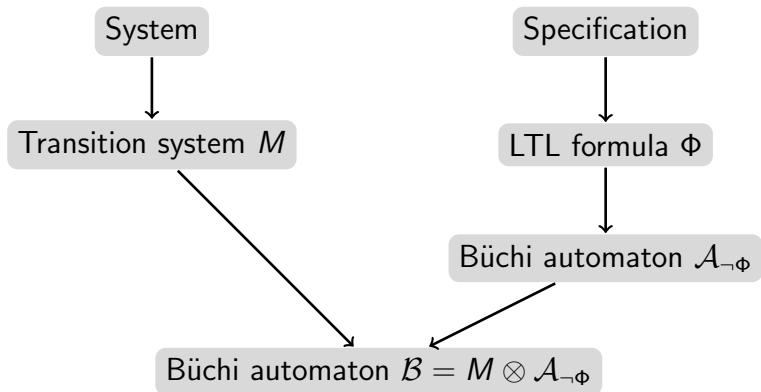
<https://spot.lrde.epita.fr/app/>

## Example

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

# Emptiness checking algorithms

# Model-checking algorithm



$$M \models \Phi \text{ iff } \text{trace}(M) \subseteq \llbracket \Phi \rrbracket \text{ iff } \mathcal{L}(\mathcal{B}) = \emptyset$$

# 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  $\Phi$

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

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

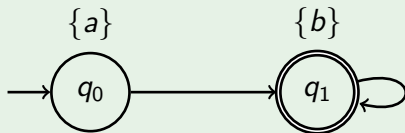
►  $I' = S_0 \times I$

►  $F' = \{(s, B) \mid B \in F\}$

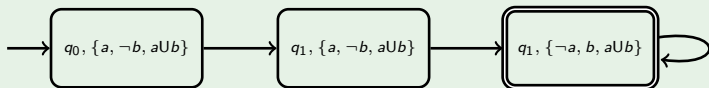
# Product $M \otimes \mathcal{A}_{aUb}$

## Example

Transition system  $M$ :



Product  $\mathcal{B} = M \otimes \mathcal{A}_{aUb}$ :





# 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  $\mathcal{B} = (Q, \Sigma, \Delta, I, F)$ 
2  RETURN: "accepting run" if  $\mathcal{L}(\mathcal{B}) \neq \emptyset$ , "no accepting run" otherwise
3
4  forall  $q_0 \in I$  do
5    dfs_blue( $q_0$ )
6  report "no accepting run"
7
8  procedure dfs_blue( $s$ ):
9     $s$ .color := cyan
10   forall  $s \rightarrow t$  in  $\Delta$  do
11     if ( $t$ .color = cyan and ( $s \in F$  or  $t \in F$ ))
12       report "accepting run" and exit
13     if ( $t$ .color = white)
14       dfs_blue( $t$ )
15     if ( $s \in F$ )
16       dfs_red( $s$ )
17     else
18        $s$ .color := blue
19
20  procedure dfs_red( $s$ ):
21     $s$ .color := red
22   forall  $s \rightarrow t$  in  $\Delta$  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  $\mathcal{B} = (Q, \Sigma, \Delta, I, F)$ 
2  RETURN: "accepting run" if  $\mathcal{L}(\mathcal{B}) \neq \emptyset$ , "no accepting run" otherwise
3
4  count := 0; Roots :=  $\emptyset$ ; Active :=  $\emptyset$ 
5  forall  $q_0 \in I$  do
6    couv( $q_0$ )
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  $q \xrightarrow{A} q'$  in  $\Delta$  do
15     if ( $q'.dfsnum = 0$ )
16       couv( $q'$ )
17     else if ( $q' \in Active$ )
18       repeat
19          $u := pop(Roots)$ 
20         if ( $u \in F$ )
21           report "accepting run" and exit
22         until ( $u.dfsnum \leq q'.dfsnum$ )
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  $\mathcal{O}(|\mathcal{B}|)$  with **generalized** conditions