

# Verification formelle à l'aide d'automates (model checking)

Renault Etienne & Alexandre Duret-Lutz

15 septembre 2015

[www.lrde.epita.fr/~renault/teaching/imc/infospe.pdf](http://www.lrde.epita.fr/~renault/teaching/imc/infospe.pdf)

# Model checking, kézako ?



Le (top) model checking permet de s'assurer qu'un système se **comporte** correctement !

# Qu'est ce qu'un système ?



# Différents types de programmes (2)

## ► Les programmes *standards*

- ① terminent
- ② produisent des résultats
- ③ peuvent être exprimés en terme de pré-conditions et de post-conditions
- ④ données complexes mais structure de contrôle simple

*Exemples:* Traduction, chiffrement, tris, compilateurs, traitement d'images, ...

**Input:** Mot fini

**Output:** Mot fini

**Manipulation d'Automates Finis.**

Vérifier le bon comportement peut se faire en regardant la sortie.

# Différents types de programmes (2)

## ► Les systèmes *réactifs*

- ① ne terminent pas nécessairement
- ② interagissent avec leur environnement
- ③ maintiennent une interaction
- ④ données simples mais structure de contrôle complexe

*Exemples:* Protocoles, système d'exploitation, jeux vidéos, ...

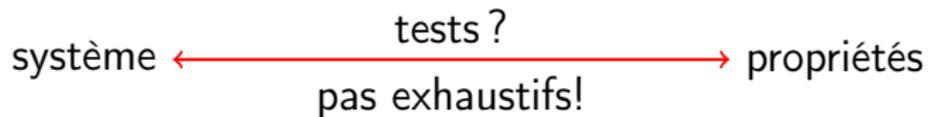
**Input:** Mot infini

**Output:** Mot infini

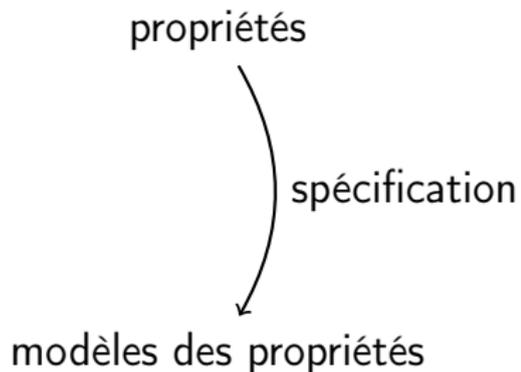
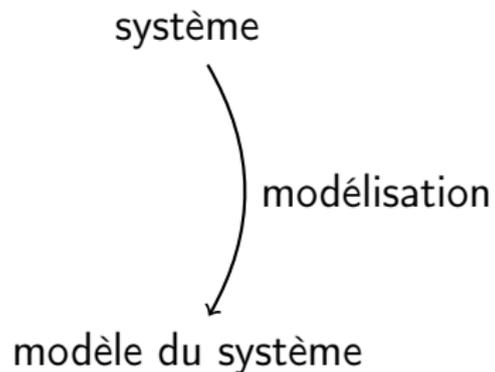
**Besoin d'Automates spéciaux (Büchi)**

Vérifier le bon comportement peut se faire via de la vérification formelle.

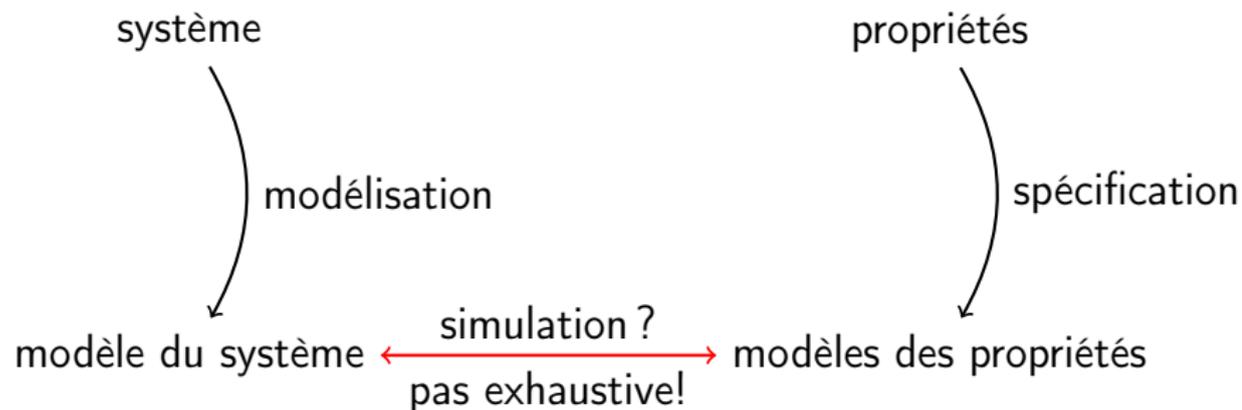
# Vérification formelle



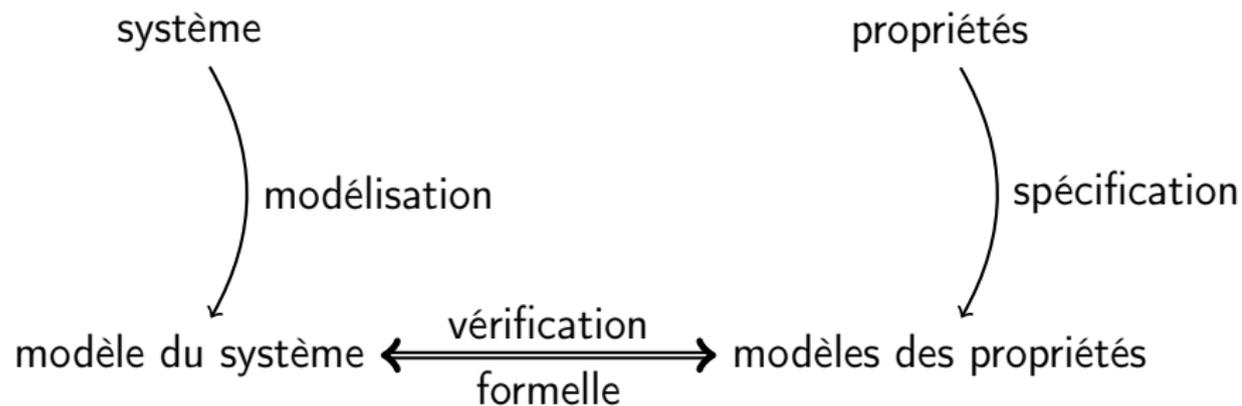
# Vérification formelle



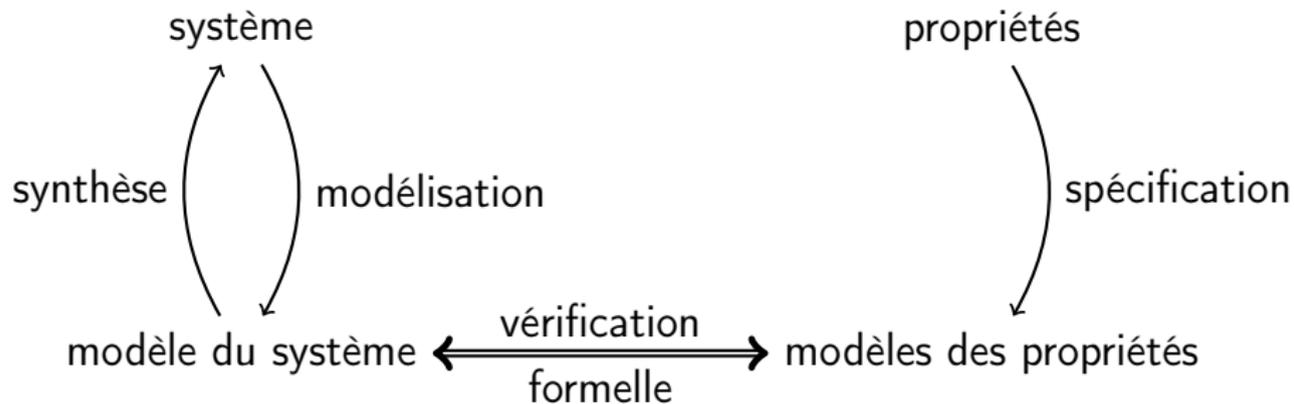
# Vérification formelle



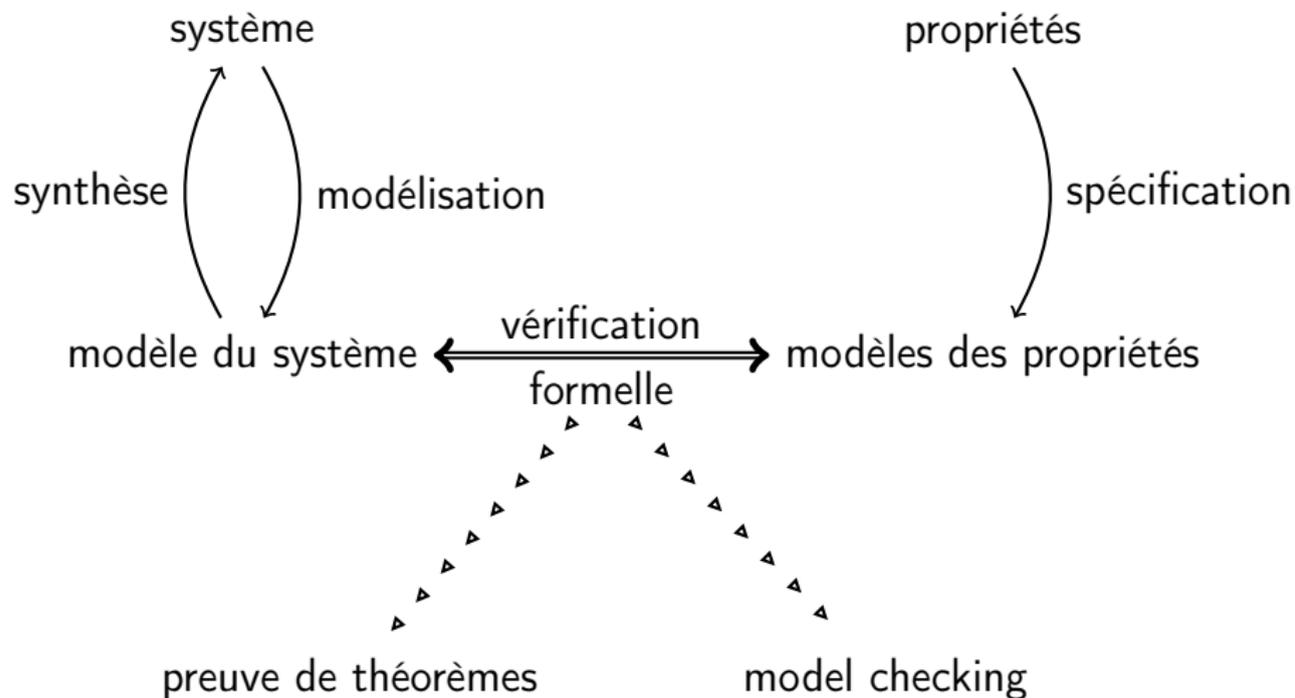
# Vérification formelle



# Vérification formelle



# Vérification formelle



Approche **automatique** de la vérification formelle, utilisant des automates.

Vérification exhaustive de tous les comportements du modèle.

L'**arnaque** : le modèle doit être suffisamment abstrait pour que son exploration soit réalisable.

# Exemple: algorithme d'exclusion mutuelle

Variables globales:  $req_P$  et  $req_Q$ .

## Processus P (boucle infinie)

1.  $req_P \leftarrow 1$
2.  $wait(req_Q = 0)$
3. section critique
4.  $req_P \leftarrow 0$

## Processus Q (boucle infinie)

1.  $req_Q \leftarrow 1$
2.  $wait(req_P = 0)$
3. section critique
4.  $req_Q \leftarrow 0$

État initial:  $P = 1, Q = 1, req_P = 0, req_Q = 0$ .

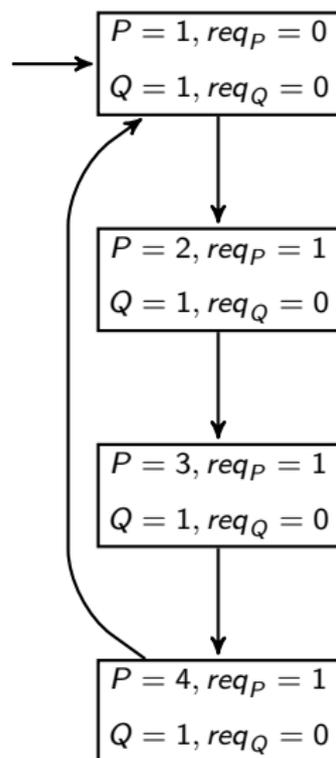
Propriétés à vérifier:

- ① À tout moment il y a au plus un processus en section critique.
- ② Tout processus demandant l'entrée en section critique finit par y entrer.

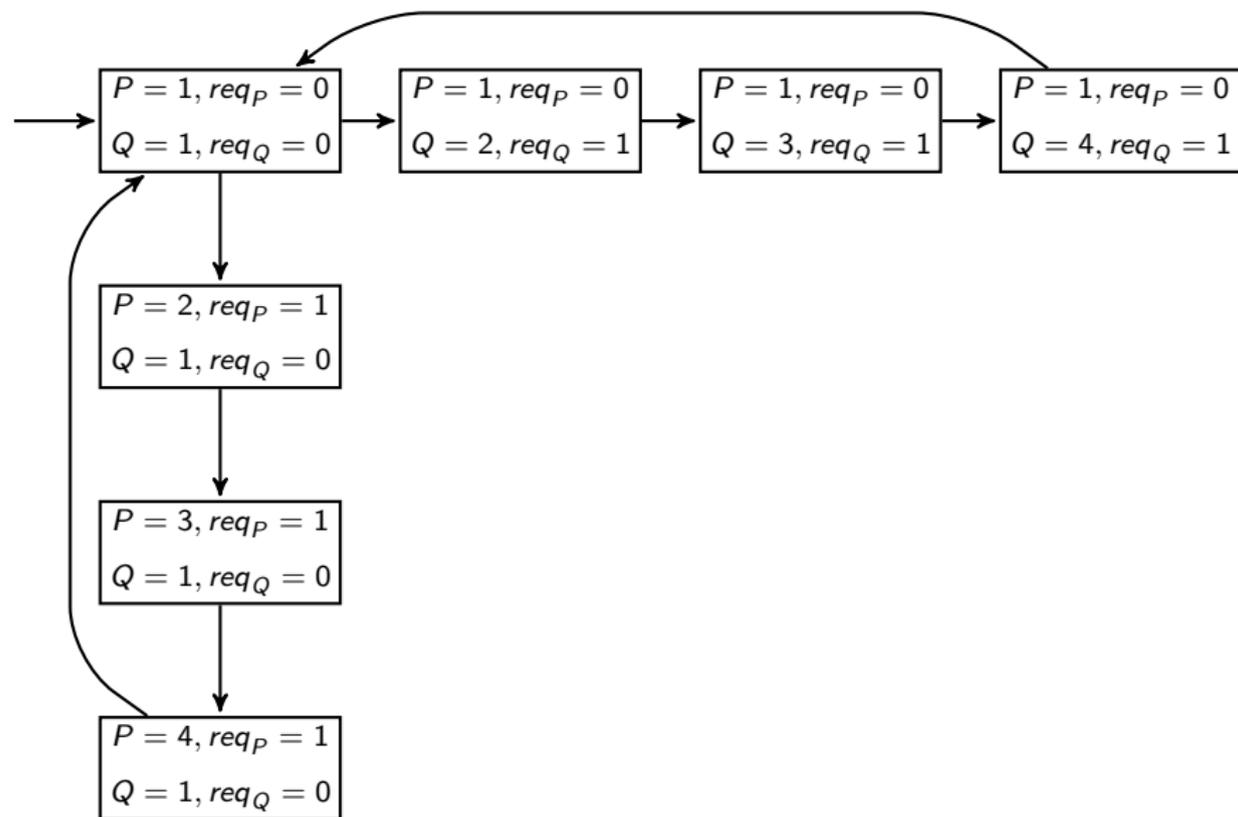
# Exemple: espace d'état ou graphe d'accessibilité

$$\longrightarrow \begin{array}{|l} P = 1, req_P = 0 \\ Q = 1, req_Q = 0 \end{array}$$

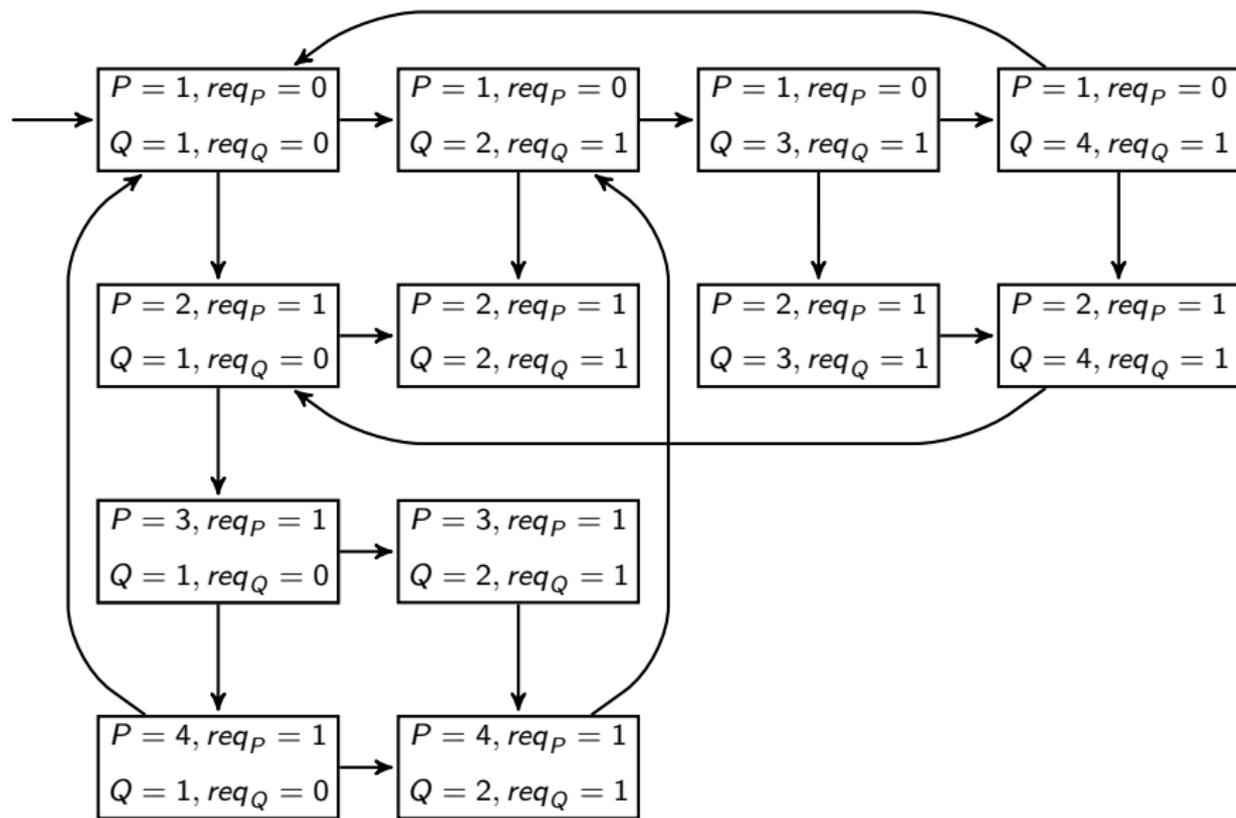
# Exemple: espace d'état ou graphe d'accessibilité



# Exemple: espace d'état ou graphe d'accessibilité



# Exemple: espace d'état ou graphe d'accessibilité



# Propriété 1

À tout moment il y a au plus un processus en section critique.

Traduction : dans aucun état on a  $P = 3$  et  $Q = 3$ .

C'est vrai.

Pour vérifier cette propriété il suffit de parcourir tout l'espace d'état une fois. On n'a besoin de connaître que l'ensemble des états, pas leurs liens.

Tout processus demandant l'entrée en section critique finit par y entrer.

Traduction: chaque chemin débutant dans un état accessible tel que  $P = 2$  passe par un état où  $P = 3$ ; idem pour  $Q = 2$  et  $Q = 3$ .

C'est faux.

L'état 

$P = 2, req_P = 1$
$Q = 2, req_Q = 1$

 n'a aucun successeur (c'est un **deadlock**).

Pour vérifier cette propriété on a besoin de connaître le graphe d'accessibilité (les états seuls ne suffisent pas).

- ▶ On représente un système avec un **automate** fini.
- ▶ On représente une propriété avec une formule de logique (temporelle).
- ▶ Pour comparer ces deux objets on convertit la formule sous forme d'automate.
- ▶ Un travail sur les deux automates nous dit s'ils sont « compatibles ».

# Logique des propositions: l'instant présent

La logique propositionnelle peut caractériser **un** instant.

$r$  : feu rouge allumé

$o$  : feu orange allumé

$v$  : feu vert allumé

$$r \wedge o \wedge v = \text{🚦}, \quad r \wedge \neg o \wedge \neg v = \text{🚦}, \quad \neg r \wedge \neg o \wedge v = \text{🚦}, \quad \neg r \wedge \neg o \wedge \neg v = \text{🚦}.$$

Comment dire que 🚦 précède 🚦 ?

Comment dire que le système ne reste pas toujours sur 🚦 ?

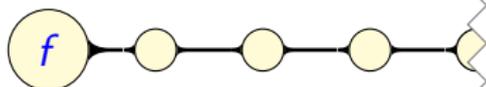
⇒ besoin de faire apparaître le temps

# Operateurs LTL

Pour  $f$  et  $g$  deux formules propositionnelles:

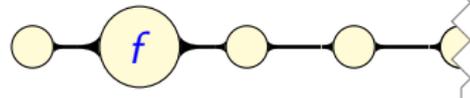
Present

$f$



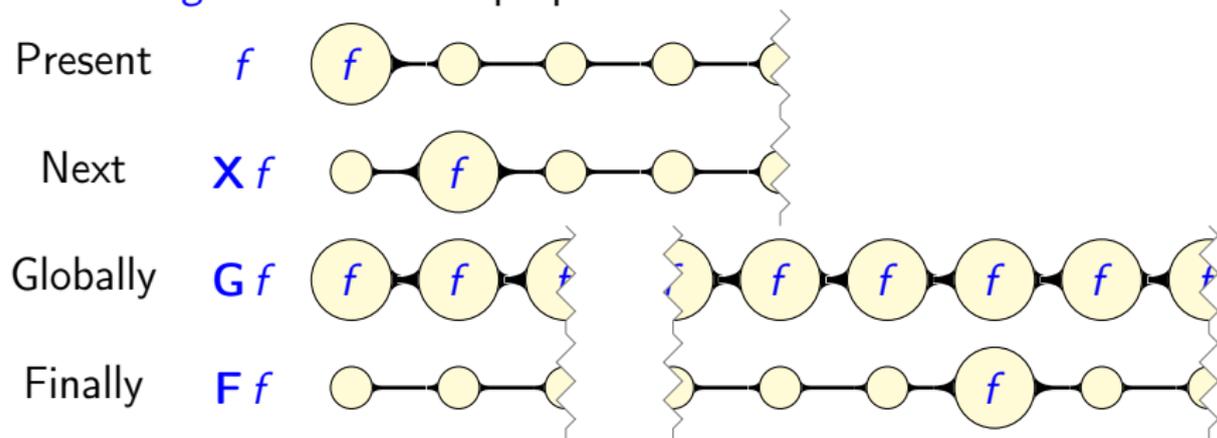
Next

$Xf$



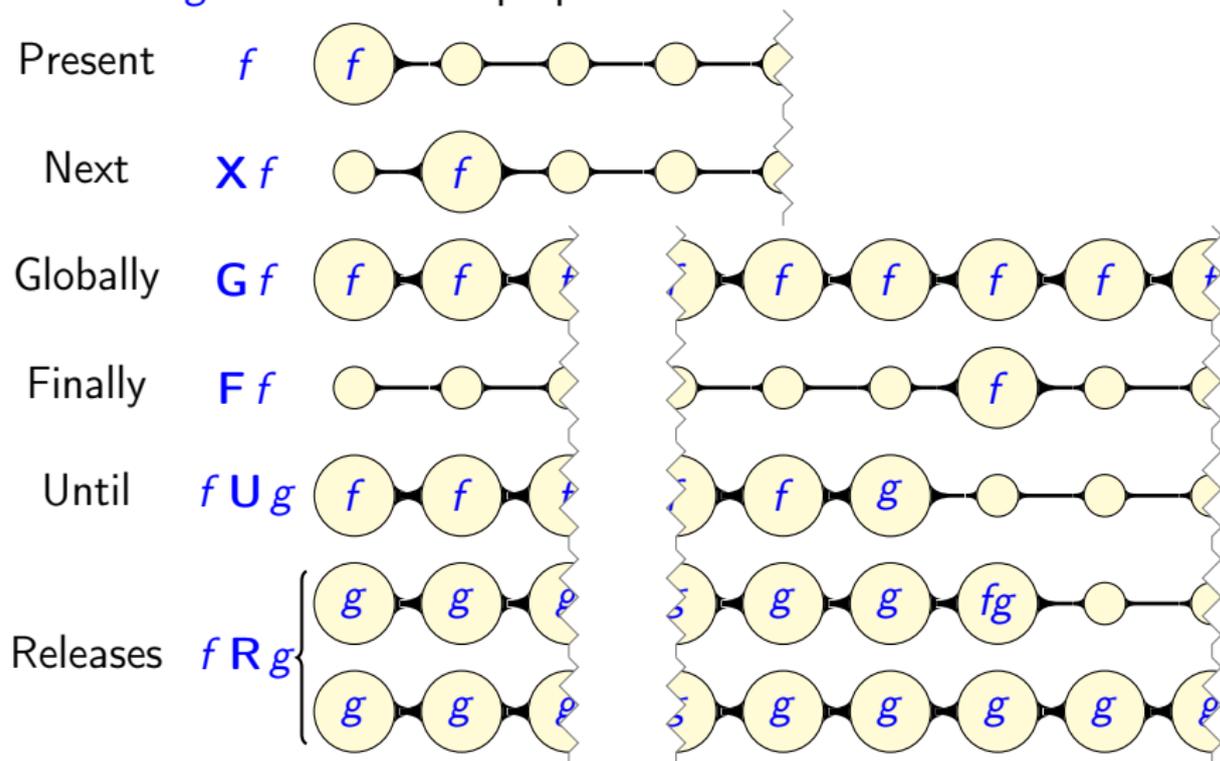
# Opérateurs LTL

Pour  $f$  et  $g$  deux formules propositionnelles:



# Operateurs LTL

Pour  $f$  et  $g$  deux formules propositionnelles:



# LTL: Exemples

Next	$X f$	$f$ est vraie à l'instant suivant
Globally	$G f$	$f$ est vraie a tout instant
Finally	$F f$	$f$ sera vraie à un instant (présent ou futur)
Until	$f U g$	$f$ est toujours vraie jusqu'à ce que $g$ le soit

$\neg G(r \wedge \neg o \wedge \neg v)$ : le système ne reste pas tout le temps .

$G((\neg r \wedge o \wedge \neg v) \rightarrow X(r \wedge \neg o \wedge \neg v))$ :  est tjs imm. suivi de .

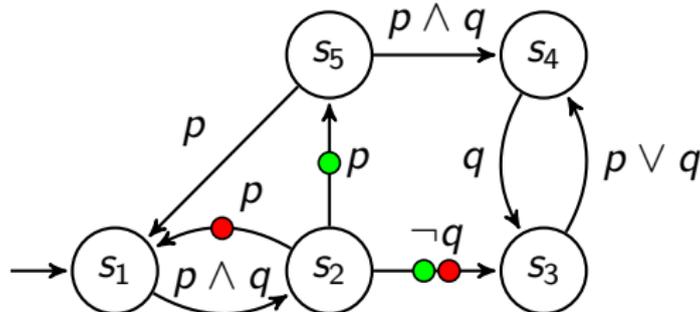
$GF(\neg r \wedge \neg o \wedge v)$ : le système passe infiniment souvent par .

# TGBA: Automates de Büchi généralisés

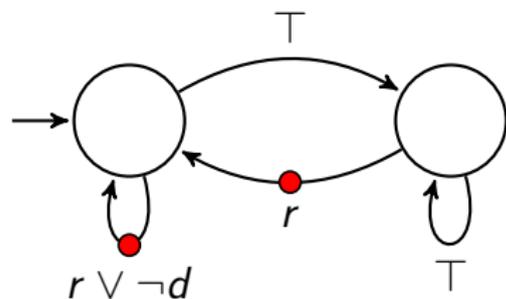
Un automate de Büchi généralisé étiqueté sur les transitions possède:

- ▶ un ensemble d'états, avec un état *initial*,
- ▶ un ensemble de transitions entre ces états, étiquetées par des formules de logique booléenne,
- ▶ un ensemble d'ensembles de transitions indiquant les conditions d'acceptation,

Un chemin infini de cet automate est accepté s'il visite infiniment souvent chaque condition d'acceptation.

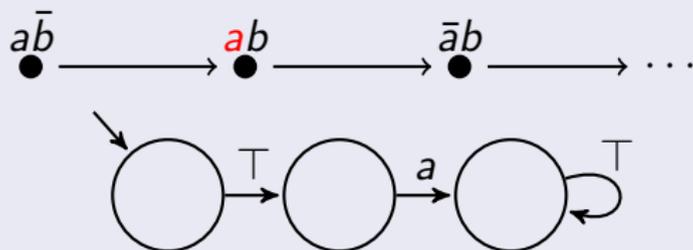


# Exemple de TGBA reconnaissant $G(d \rightarrow F r)$

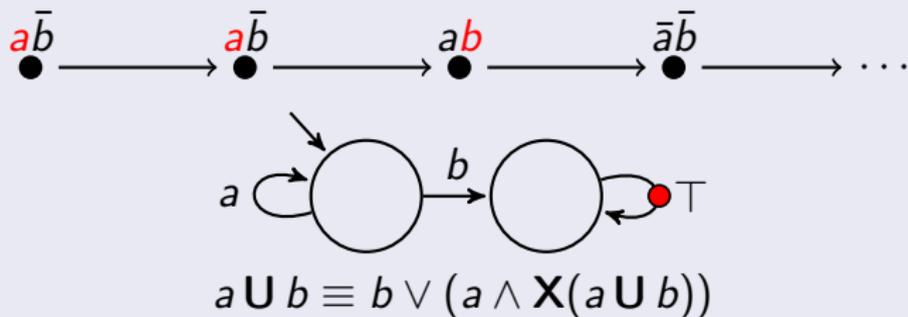


# LTL et automates de Büchi

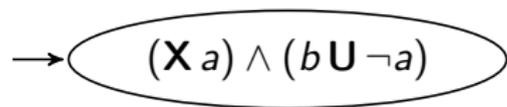
$\mathbf{X} a$



$a \mathbf{U} b$

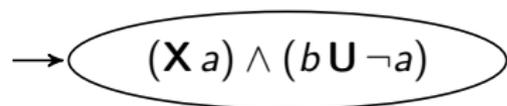


# Tableau Construction for LTL



- 1 Label the initial state by the formula to translate

# Tableau Construction for LTL



- 1 Label the initial state by the formula to translate
- 2 Rewrite each state label  $\varphi$  as

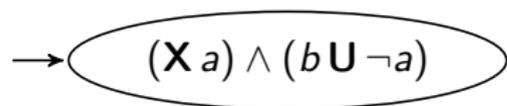
$$\bigvee_i \beta_i \wedge \mathbf{X} \psi_i$$

Boolean formula                  LTL formula

Since  $f \mathbf{U} g = g \vee (f \wedge \mathbf{X}(f \mathbf{U} g))$  we have:

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

# Tableau Construction for LTL



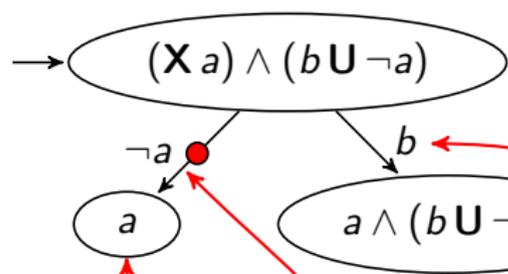
- 1 Label the initial state by the formula to translate
- 2 Rewrite each state label  $\varphi$  as

$$\bigvee_i \beta_i \wedge \mathbf{X} \psi_i$$

Since  $f \mathbf{U} g = g \vee (f \wedge \mathbf{X}(f \mathbf{U} g))$  we have:

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

# Tableau Construction for LTL



- 1 Label the initial state by the formula to translate
- 2 Rewrite each state label  $\varphi$  as

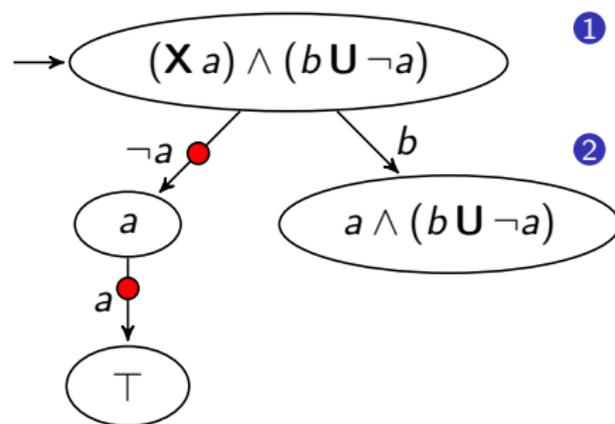
$$\bigvee_i \beta_i \wedge \mathbf{X} \psi_i$$

Then connect  $\varphi$  to each  $\psi_i$  using  $\beta_i$  as label

Since  $f \mathbf{U} g = g \vee (f \wedge \mathbf{X}(f \mathbf{U} g))$  we have:

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

# Tableau Construction for LTL



- ① Label the initial state by the formula to translate
- ② Rewrite each state label  $\varphi$  as

$$\bigvee_i \beta_i \wedge \mathbf{X} \psi_i$$

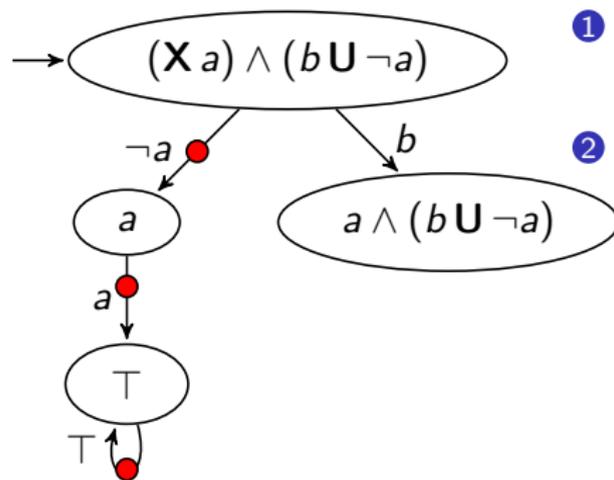
Then connect  $\varphi$  to each  $\psi_i$  using  $\beta_i$  as label

Since  $f \mathbf{U} g = g \vee (f \wedge \mathbf{X}(f \mathbf{U} g))$  we have:

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

$$a = a \wedge \mathbf{X} \top$$

# Tableau Construction for LTL



- 1 Label the initial state by the formula to translate
- 2 Rewrite each state label  $\varphi$  as

$$\bigvee_i \beta_i \wedge \mathbf{X} \psi_i$$

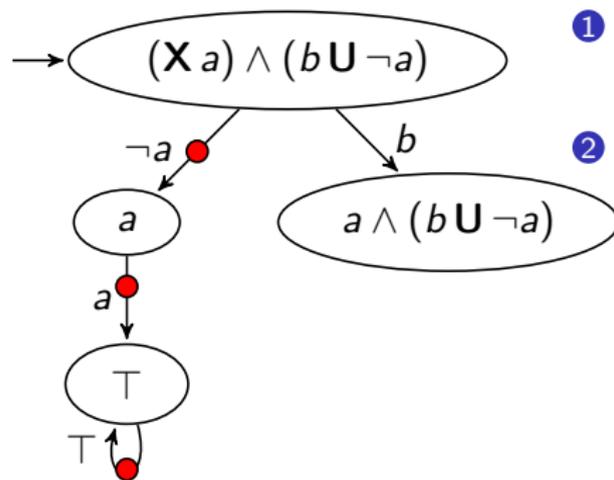
Then connect  $\varphi$  to each  $\psi_i$  using  $\beta_i$  as label

Since  $f \mathbf{U} g = g \vee (f \wedge \mathbf{X}(f \mathbf{U} g))$  we have:

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

$$a = a \wedge \mathbf{X} \top; \quad \top = \top \wedge \mathbf{X} \top$$

# Tableau Construction for LTL



- 1 Label the initial state by the formula to translate
- 2 Rewrite each state label  $\varphi$  as

$$\bigvee_i \beta_i \wedge \mathbf{X} \psi_i$$

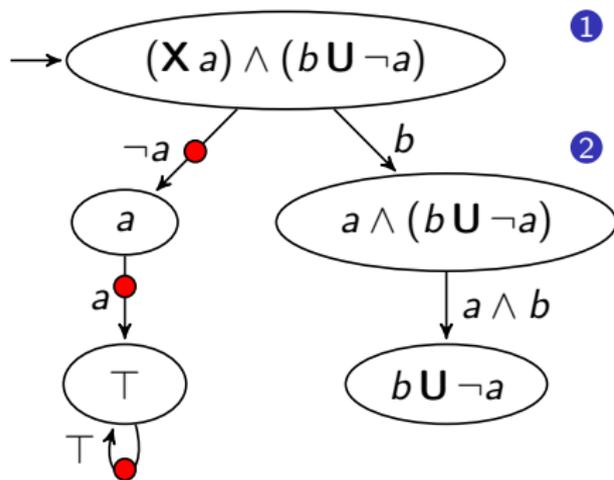
Then connect  $\varphi$  to each  $\psi_i$  using  $\beta_i$  as label

Since  $f \mathbf{U} g = g \vee (f \wedge \mathbf{X}(f \mathbf{U} g))$  we have:

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

$$a = a \wedge \mathbf{X} \top; \quad \top = \top \wedge \mathbf{X} \top; \quad a \wedge (b \mathbf{U} \neg a) = a \wedge (\neg a \vee (b \wedge \mathbf{X}(b \mathbf{U} \neg a)))$$

# Tableau Construction for LTL



- 1 Label the initial state by the formula to translate
- 2 Rewrite each state label  $\varphi$  as

$$\bigvee_i \beta_i \wedge \mathbf{X} \psi_i$$

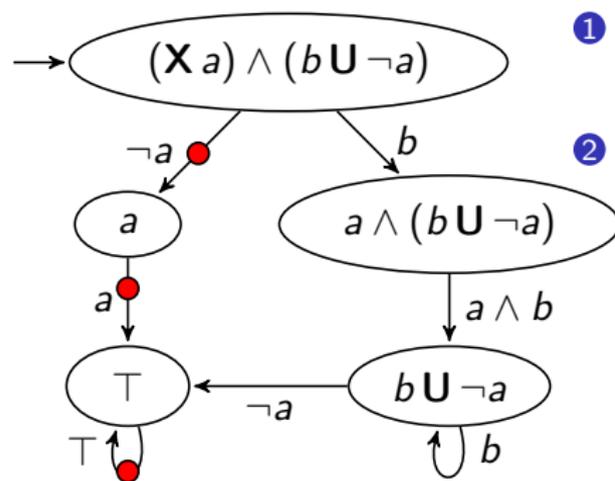
Then connect  $\varphi$  to each  $\psi_i$  using  $\beta_i$  as label

Since  $f \mathbf{U} g = g \vee (f \wedge \mathbf{X}(f \mathbf{U} g))$  we have:

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

$$a = a \wedge \mathbf{X} \top; \quad \top = \top \wedge \mathbf{X} \top; \quad a \wedge (b \mathbf{U} \neg a) = a \wedge b \wedge \mathbf{X}(b \mathbf{U} \neg a)$$

# Tableau Construction for LTL



- 1 Label the initial state by the formula to translate
- 2 Rewrite each state label  $\varphi$  as

$$\bigvee_i \beta_i \wedge \mathbf{X} \psi_i$$

Then connect  $\varphi$  to each  $\psi_i$  using  $\beta_i$  as label

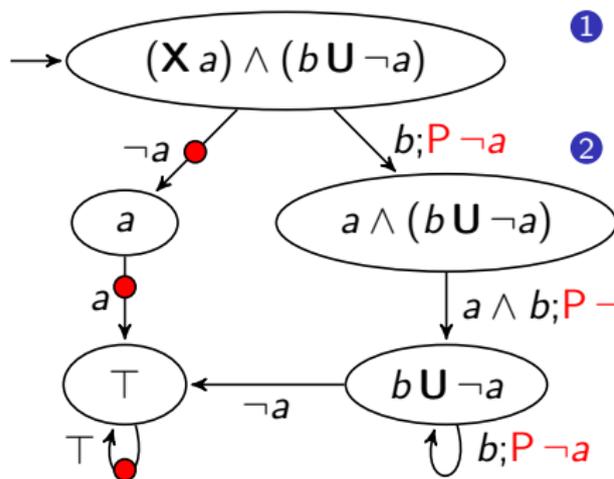
Since  $f \mathbf{U} g = g \vee (f \wedge \mathbf{X}(f \mathbf{U} g))$  we have:

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

$$a = a \wedge \mathbf{X} \top; \quad \top = \top \wedge \mathbf{X} \top; \quad a \wedge (b \mathbf{U} \neg a) = a \wedge b \wedge \mathbf{X}(b \mathbf{U} \neg a)$$

$$b \mathbf{U} \neg a = (a \wedge \mathbf{X} \top) \vee (b \wedge \mathbf{X}(b \mathbf{U} \neg a))$$

# Tableau Construction for LTL



① Label the initial state by the formula to translate

② Rewrite each state label  $\varphi$  as

$$\bigvee \left( \beta_i \wedge \mathbf{X} \psi_i \wedge \bigwedge_j \mathbf{P} \gamma_{ij} \right)$$

Then connect  $\varphi$  to each  $\psi_i$  using  $\beta_i$  as label and  $\{\mathbf{P} \gamma_{ij}\}_j$  as promises

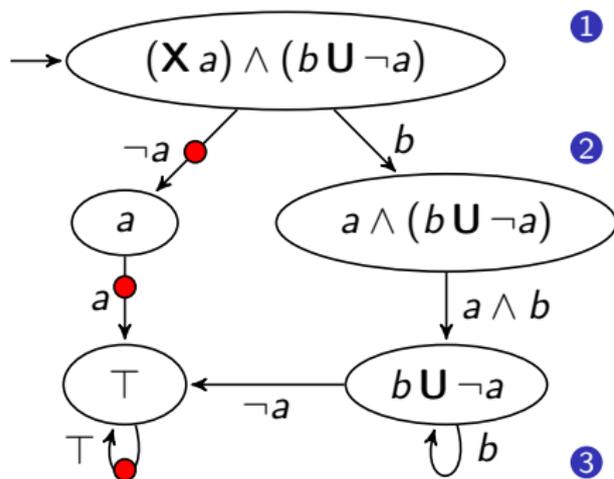
Since  $f \mathbf{U} g = g \vee (f \wedge \mathbf{X}(f \mathbf{U} g) \wedge \mathbf{P} g)$  we have:

$$(\mathbf{X} a) \wedge (b \mathbf{U} \neg a) = (\neg a \wedge \mathbf{X} a) \vee (b \wedge \mathbf{X}(a \wedge (b \mathbf{U} \neg a))) \wedge \mathbf{P} \neg a$$

$$a = a \wedge \mathbf{X} \top; \quad \top = \top \wedge \mathbf{X} \top; \quad a \wedge (b \mathbf{U} \neg a) = a \wedge b \wedge \mathbf{X}(b \mathbf{U} \neg a) \wedge \mathbf{P} \neg a$$

$$b \mathbf{U} \neg a = (a \wedge \mathbf{X} \top) \vee (b \wedge \mathbf{X}(b \mathbf{U} \neg a) \wedge \mathbf{P} \neg a)$$

# Tableau Construction for LTL



① Label the initial state by the formula to translate

② Rewrite each state label  $\varphi$  as

$$\bigvee_i \left( \beta_i \wedge \mathbf{X} \psi_i \wedge \bigwedge_j \mathbf{P} \gamma_{ij} \right)$$

Then connect  $\varphi$  to each  $\psi_i$  using  $\beta_i$  as label and  $\{\mathbf{P} \gamma_{ij}\}_j$  as promises

③ Create Büchi acceptance sets complementing each promise

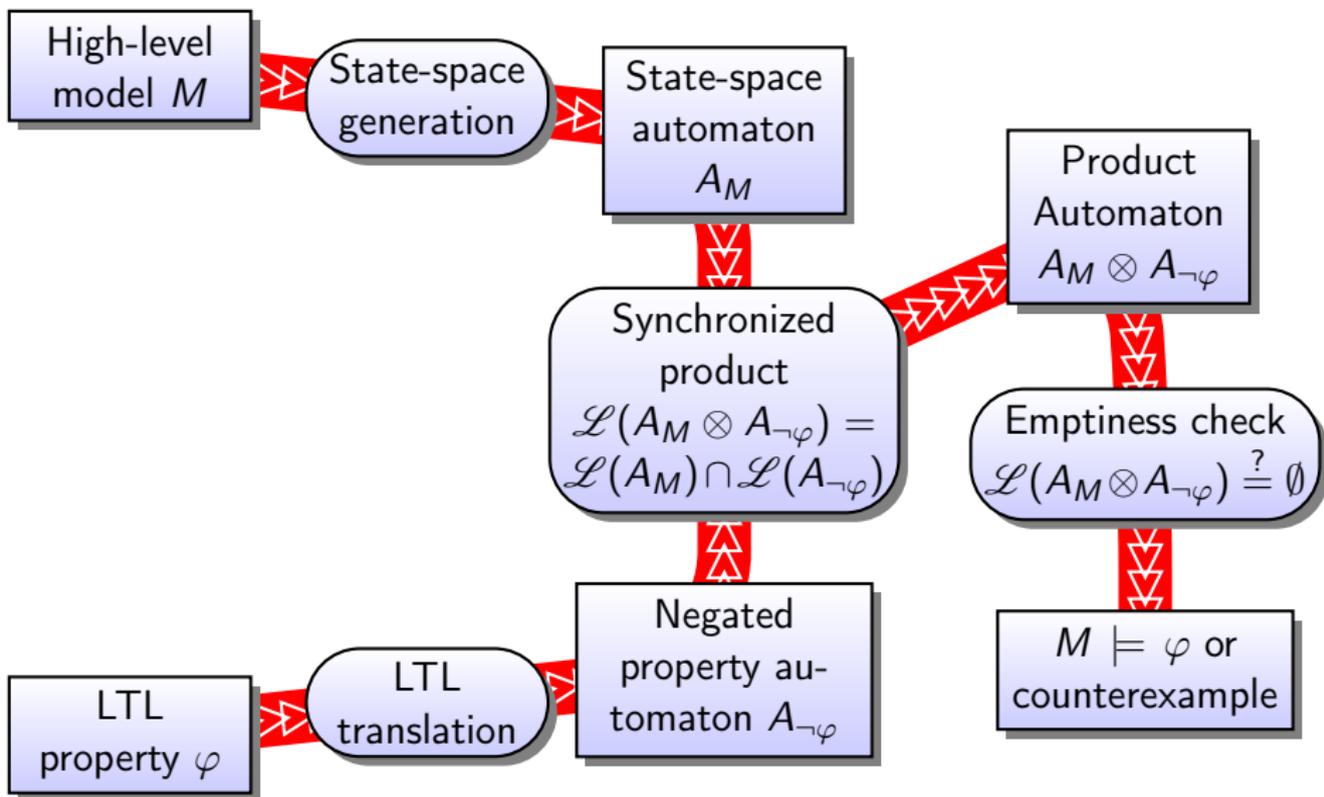
Since  $f \mathbf{U} g = g \vee (f \wedge \mathbf{X}(f \mathbf{U} g)) \wedge \mathbf{P} g$  we have:

$$(\mathbf{X} a) \wedge (b \mathbf{U} \neg a) = (\neg a \wedge \mathbf{X} a) \vee (b \wedge \mathbf{X}(a \wedge (b \mathbf{U} \neg a))) \wedge \mathbf{P} \neg a$$

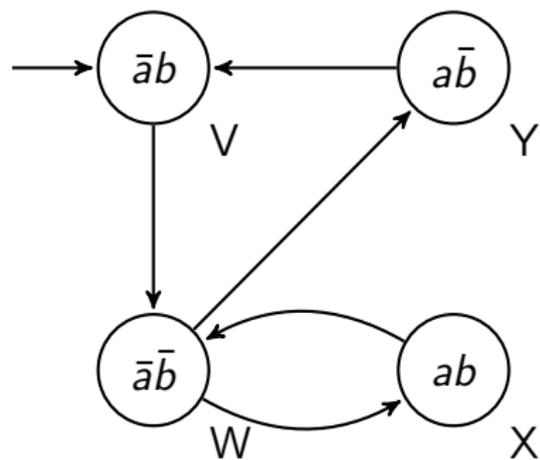
$$a = a \wedge \mathbf{X} T; T = T \wedge \mathbf{X} T; a \wedge (b \mathbf{U} \neg a) = a \wedge b \wedge \mathbf{X}(b \mathbf{U} \neg a) \wedge \mathbf{P} \neg a$$

$$b \mathbf{U} \neg a = (a \wedge \mathbf{X} T) \vee (b \wedge \mathbf{X}(b \mathbf{U} \neg a)) \wedge \mathbf{P} \neg a$$

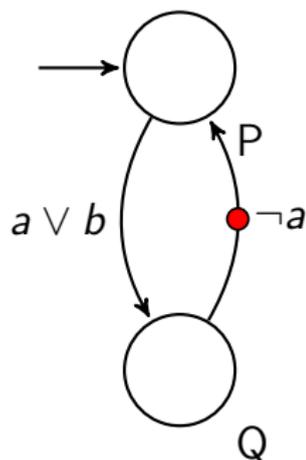
# Automata-Theoretic LTL Model Checking



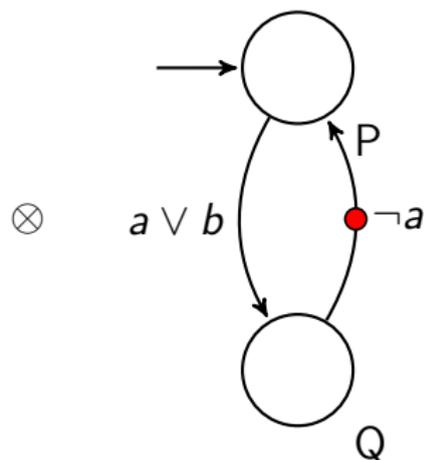
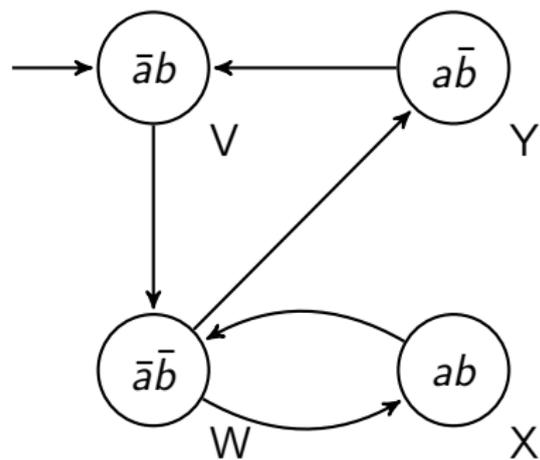
# Produit entre structure de Kripke et TGBA



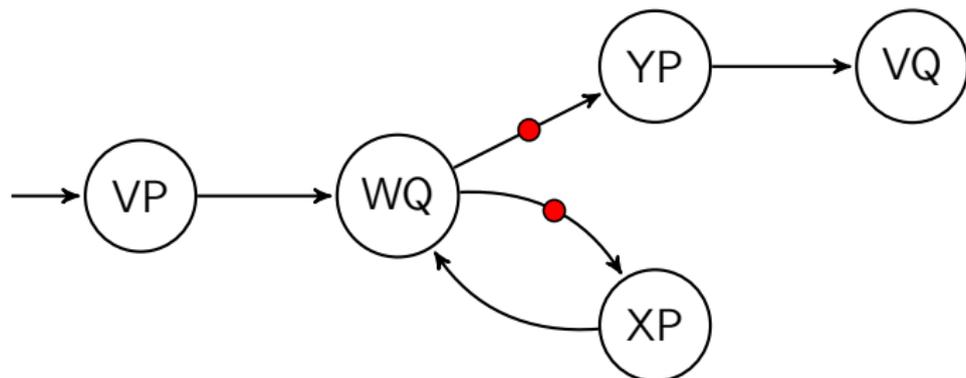
$\otimes$



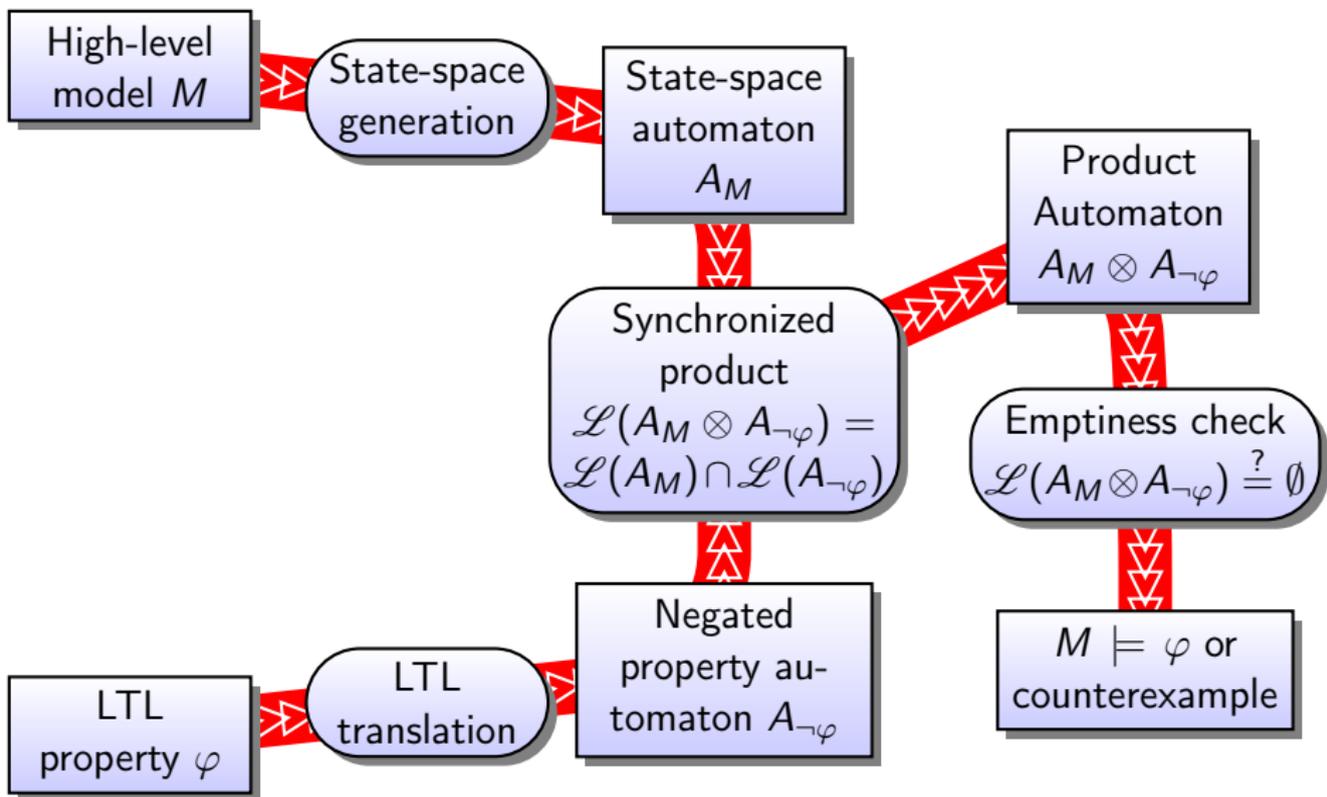
# Produit entre structure de Kripke et TGBA



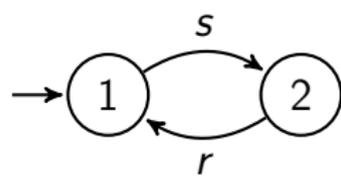
$\otimes$



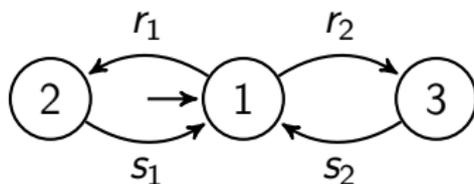
# Automata-Theoretic LTL Model Checking



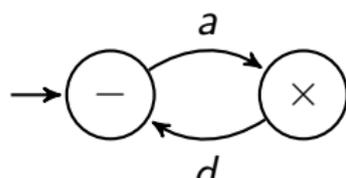
# Ex.: clients/serveur par automates synchronisés



Client C



Serveur S



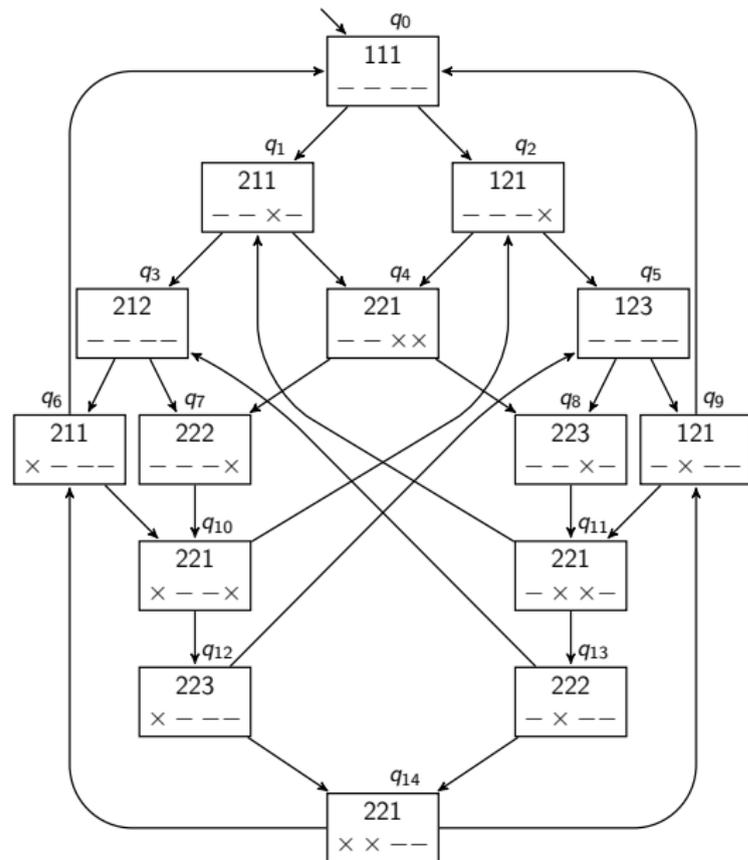
Canal B

Règles de synchronisation pour le système  $\langle C, C, S, B, B, B, B \rangle$ :

- (1)  $\langle s, \cdot, \cdot, \cdot, \cdot, \cdot, a, \cdot \rangle$
- (2)  $\langle \cdot, s, \cdot, \cdot, \cdot, \cdot, \cdot, a \rangle$
- (3)  $\langle r, \cdot, \cdot, \cdot, d, \cdot, \cdot, \cdot \rangle$
- (4)  $\langle \cdot, r, \cdot, \cdot, \cdot, d, \cdot, \cdot \rangle$
- (5)  $\langle \cdot, \cdot, r_1, \cdot, \cdot, \cdot, d, \cdot \rangle$
- (6)  $\langle \cdot, \cdot, \cdot, s_1, a, \cdot, \cdot, \cdot \rangle$
- (7)  $\langle \cdot, \cdot, \cdot, r_2, \cdot, \cdot, \cdot, d \rangle$
- (8)  $\langle \cdot, \cdot, \cdot, s_2, \cdot, a, \cdot, \cdot \rangle$

Si un client envoie une requête, recevra-t-il forcément une réponse ?

# Espace d'états de l'exemple



# Propositions atomiques pour l'exemple

On souhaite exprimer des propriétés concernant les envois et réceptions de messages.  $AP = \{r_1, r_2, d_1, d_2\}$  avec:

- ▶  $r_1$ : une réponse est en chemin entre le serveur et le premier client
- ▶  $r_2$ : une réponse est en chemin entre le serveur et le second client
- ▶  $d_1$ : une requête ( $d$  pour demande) est en chemin entre le premier client et le serveur
- ▶  $d_2$ : une requête est en chemin entre le second client et le serveur

Comment traduire « Si un client envoie une requête, il recevra forcément une réponse » avec ces propositions atomiques ?

# Propositions atomiques pour l'exemple

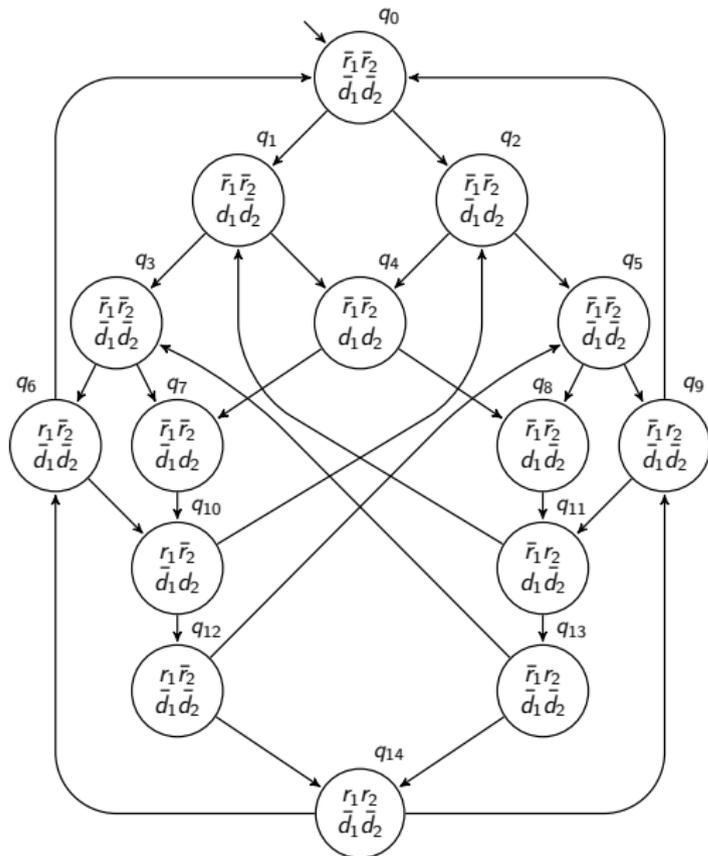
On souhaite exprimer des propriétés concernant les envois et réceptions de messages.  $AP = \{r_1, r_2, d_1, d_2\}$  avec:

- ▶  $r_1$ : une réponse est en chemin entre le serveur et le premier client
- ▶  $r_2$ : une réponse est en chemin entre le serveur et le second client
- ▶  $d_1$ : une requête ( $d$  pour demande) est en chemin entre le premier client et le serveur
- ▶  $d_2$ : une requête est en chemin entre le second client et le serveur

Comment traduire « Si un client envoie une requête, il recevra forcément une réponse » avec ces propositions atomiques ?

Pour tout  $i \in \{1, 2\}$ , si un état vérifie  $d_i$  alors dans tous ses futurs possibles il possède un successeur qui vérifie  $r_i$ .

# Structure de Kripke pour l'exemple

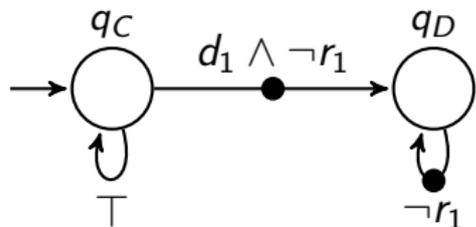


# Exploration de la structure de Kripke

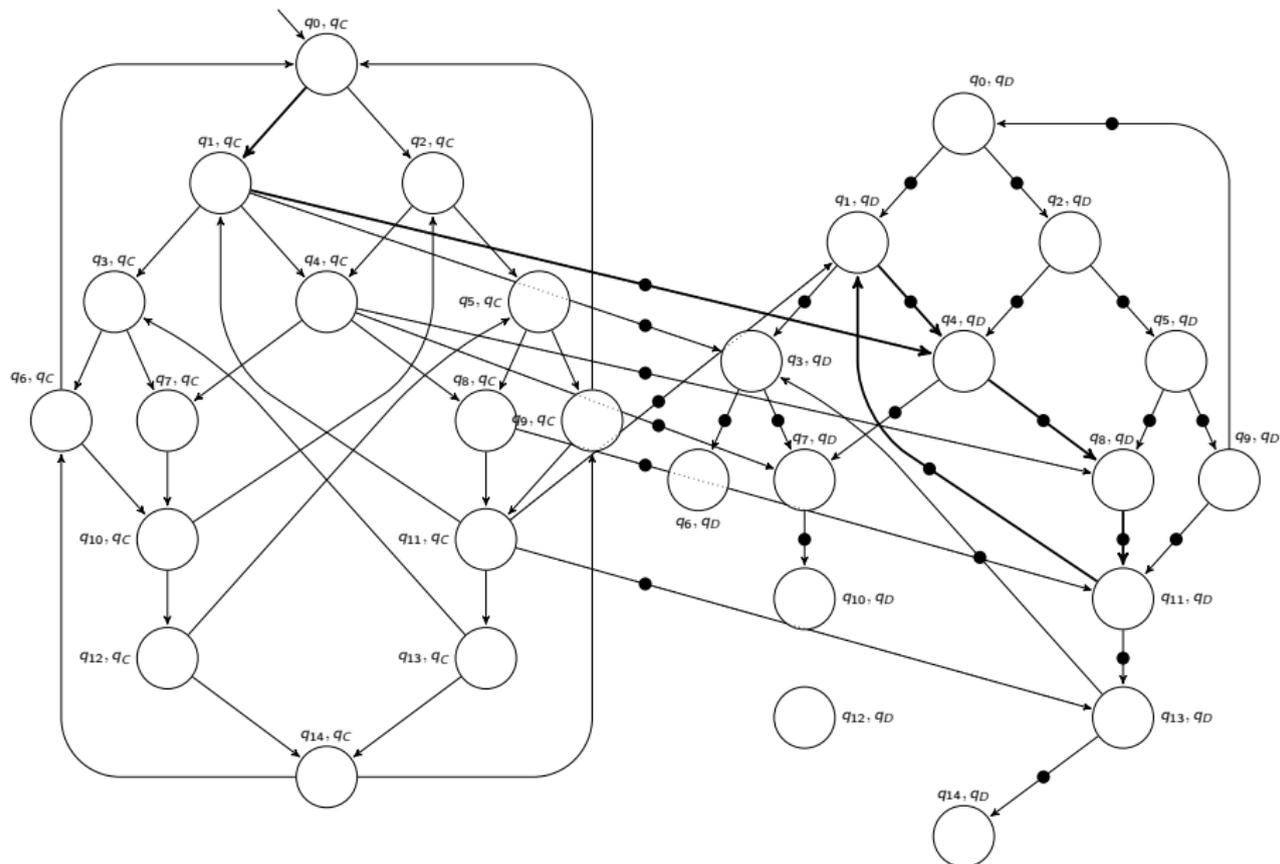
Pour tout  $i \in \{1, 2\}$ , si un état vérifie  $d_i$  alors dans tous ses futurs possibles il possède un successeur qui vérifie  $r_i$ .

On cherche un contre-exemple, c'est-à-dire un chemin infini qui passe par  $d_i$  sans jamais passer par  $r_i$ . Par symétrie on peut se limiter à  $i = 1$ .

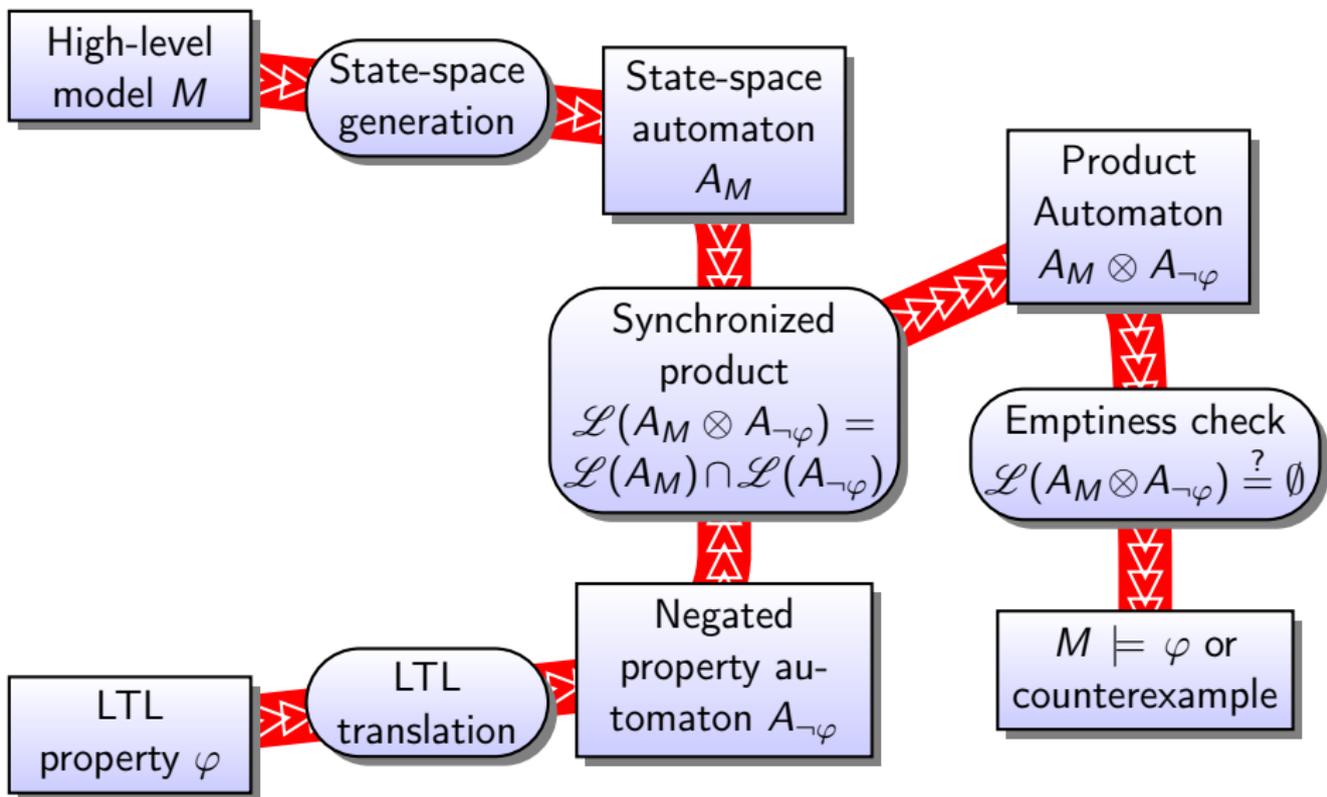
Le chemin qu'on voudrait reconnaître peut être reconnu par l'automate suivant, négation de la formule  $\mathbf{G}(d_1 \rightarrow \mathbf{F} r_1)$ :



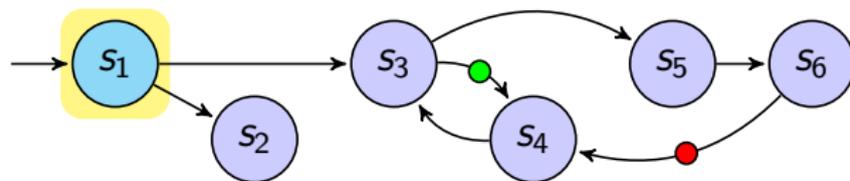
# Produit structure de Kripke/Automate



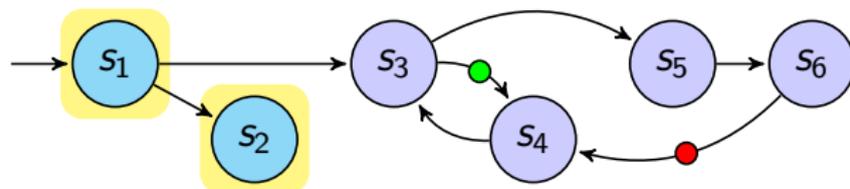
# Automata-Theoretic LTL Model Checking



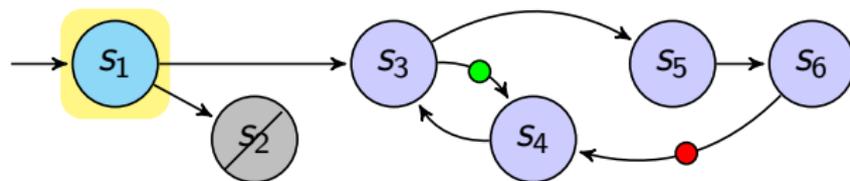
# Emptiness check



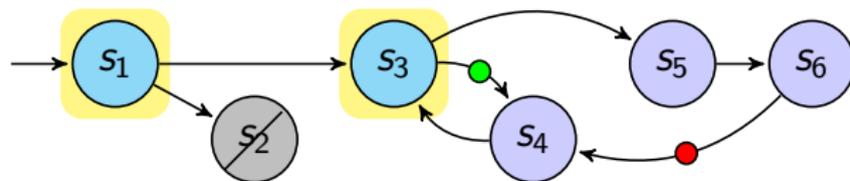
# Emptiness check



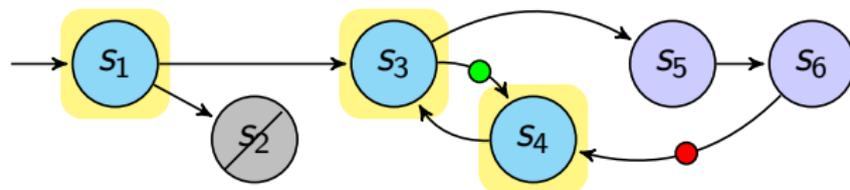
# Emptiness check



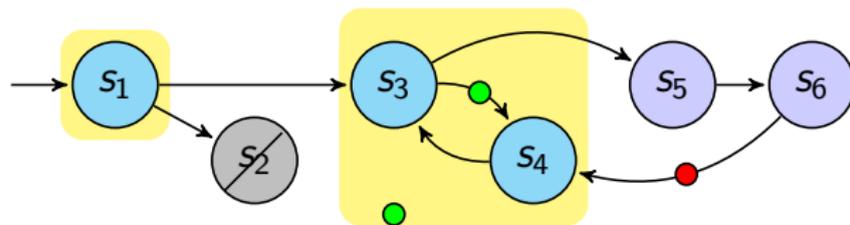
# Emptiness check



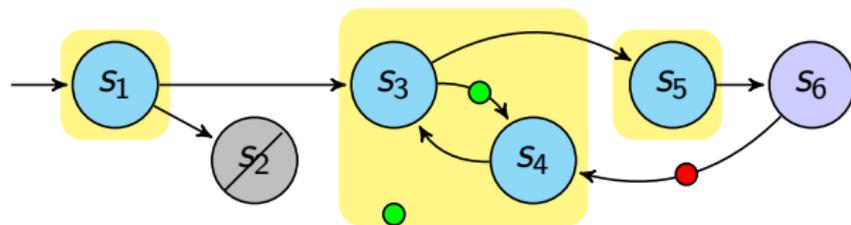
# Emptiness check



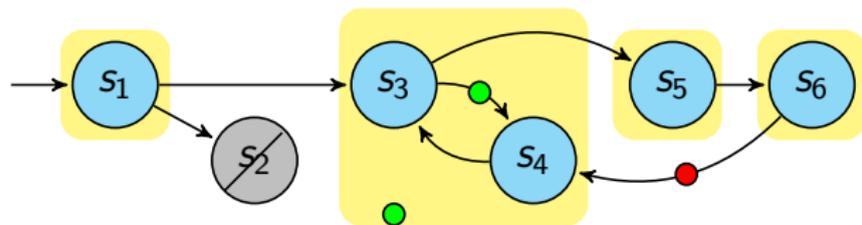
# Emptiness check



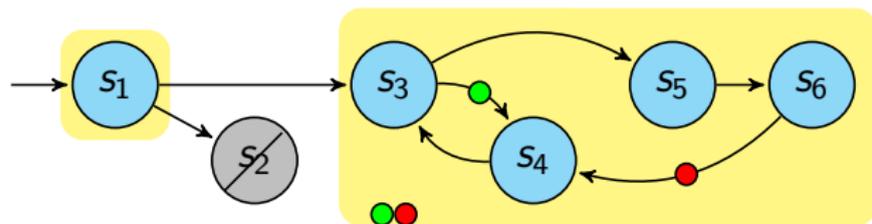
# Emptiness check



# Emptiness check

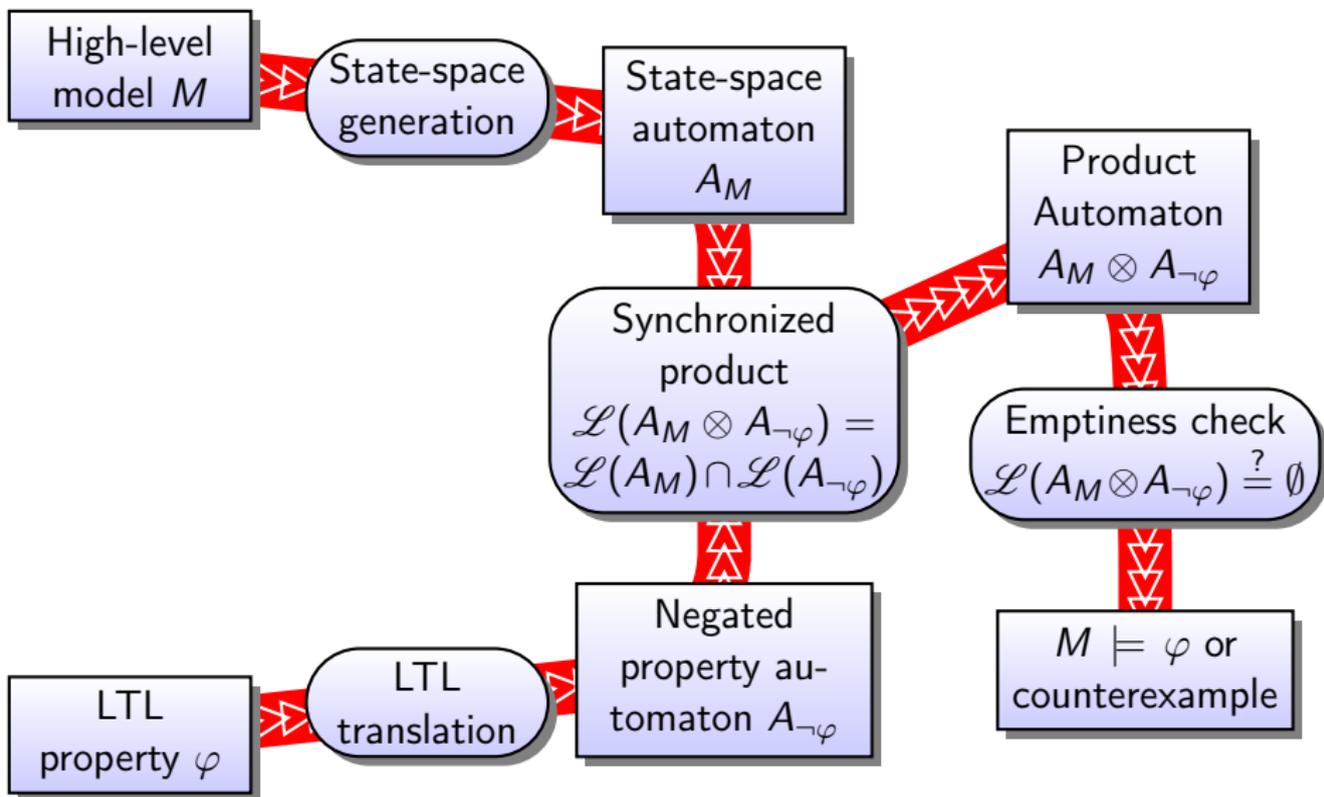


# Emptiness check

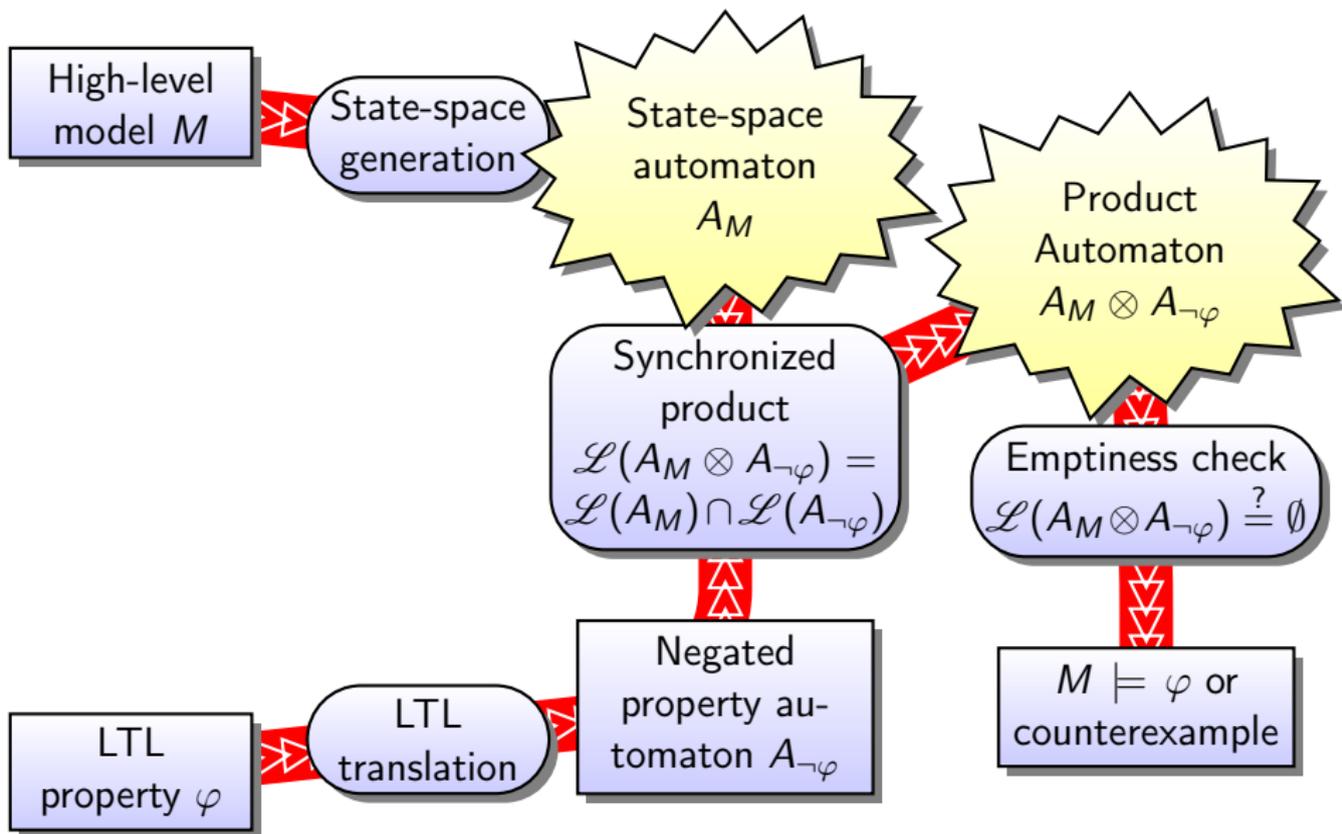


Counterexample found !

# Automata-Theoretic LTL Model Checking



# Automata-Theoretic LTL Model Checking



# Automata-Theoretic LTL Model Checking

High-level  
model  $M$

On-the-fly generation  
of state-space automaton  
 $A_M$

Product  
Automaton  
 $A_M \otimes A_{\neg\varphi}$

Synchronized  
product  
 $\mathcal{L}(A_M \otimes A_{\neg\varphi}) =$   
 $\mathcal{L}(A_M) \cap \mathcal{L}(A_{\neg\varphi})$

Emptiness check  
 $\mathcal{L}(A_M \otimes A_{\neg\varphi}) \stackrel{?}{=} \emptyset$

LTL  
property  $\varphi$

LTL  
translation

Negated  
property au-  
tomaton  $A_{\neg\varphi}$

$M \models \varphi$  or  
counterexample

# Automata-Theoretic LTL Model Checking

High-level  
model  $M$

On-the-fly generation  
of state-space automaton  
 $A_M$

On-the-fly  
synchronized product  
 $\mathcal{L}(A_M \otimes A_{\neg\varphi}) =$   
 $\mathcal{L}(A_M) \cap \mathcal{L}(A_{\neg\varphi})$

Emptiness check  
 $\mathcal{L}(A_M \otimes A_{\neg\varphi}) \stackrel{?}{=} \emptyset$

LTL  
property  $\varphi$

LTL  
translation

Negated  
property au-  
tomaton  $A_{\neg\varphi}$

$M \models \varphi$  or  
counterexample

# Automata-Theoretic LTL Model Checking

