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

Alexandre Duret-Lutz

12 septembre 2017

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

Vérification formelle

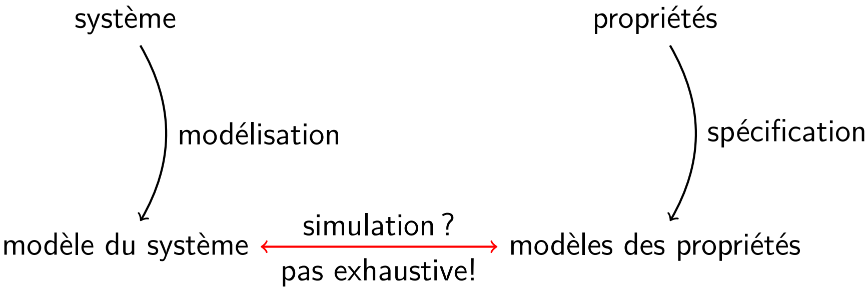
$$\text{système} \xleftrightarrow[\text{pas exhaustifs!}]{\text{tests ?}} \text{propriétés}$$

Vérification formelle

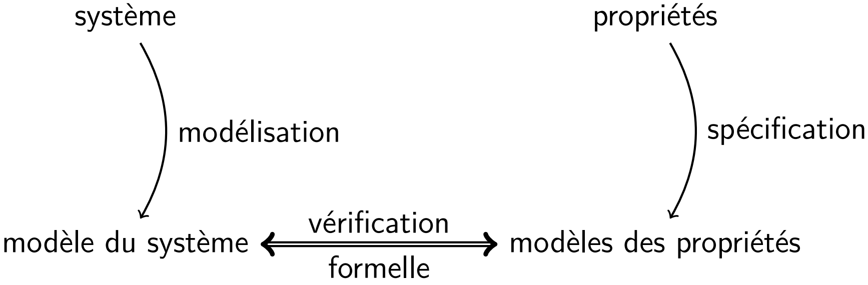
$$\text{système} \xrightarrow{\text{modélisation}} \text{modèle du système}$$

$$\text{propriétés} \xrightarrow{\text{spécification}} \text{modèles des propriétés}$$

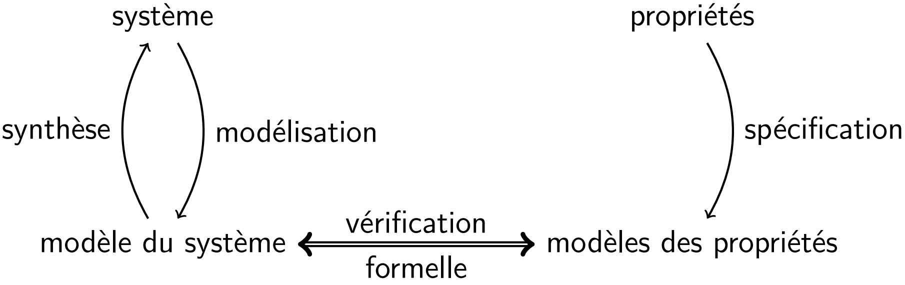
Vérification formelle



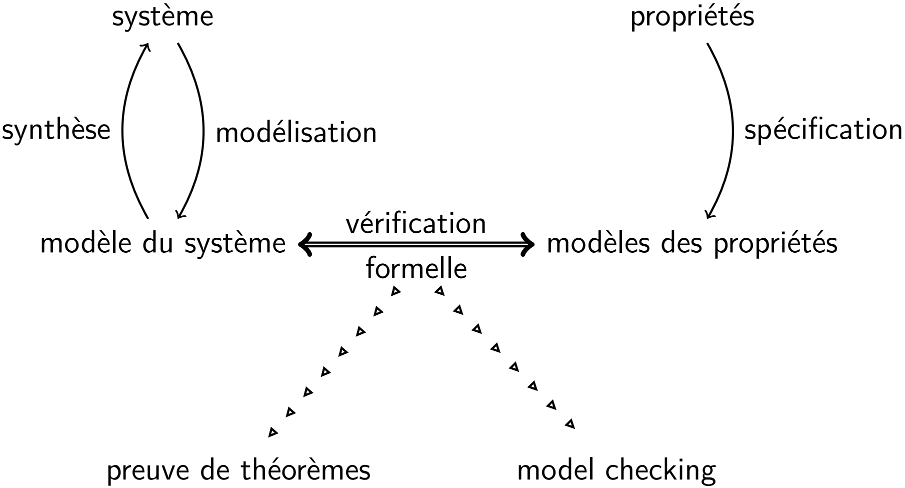
Vérification formelle



Vérification formelle



Vérification formelle



Model checking

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

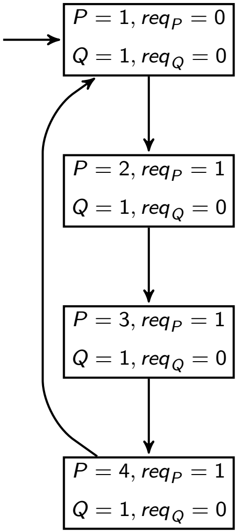
Propriétés à vérifier:

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.

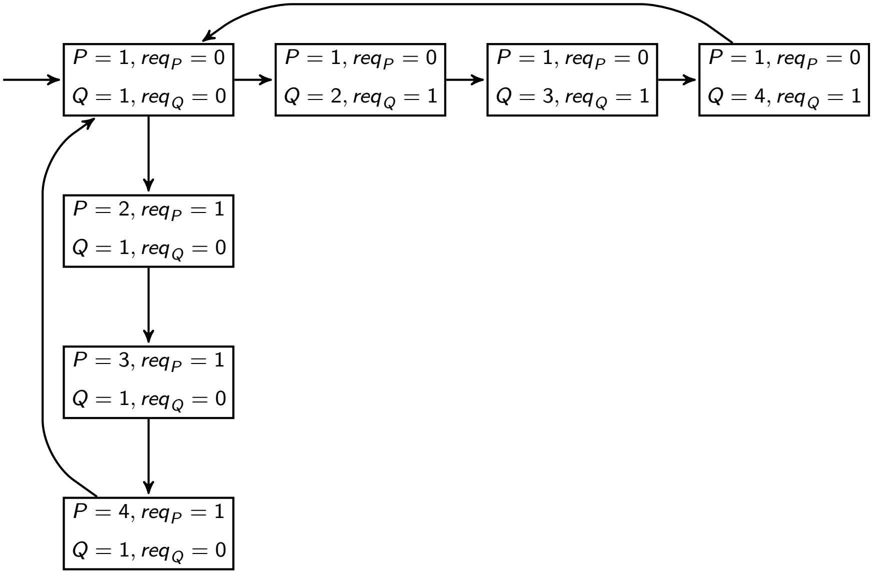
Exemple: espace d'état ou graphe d'accessibilité

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

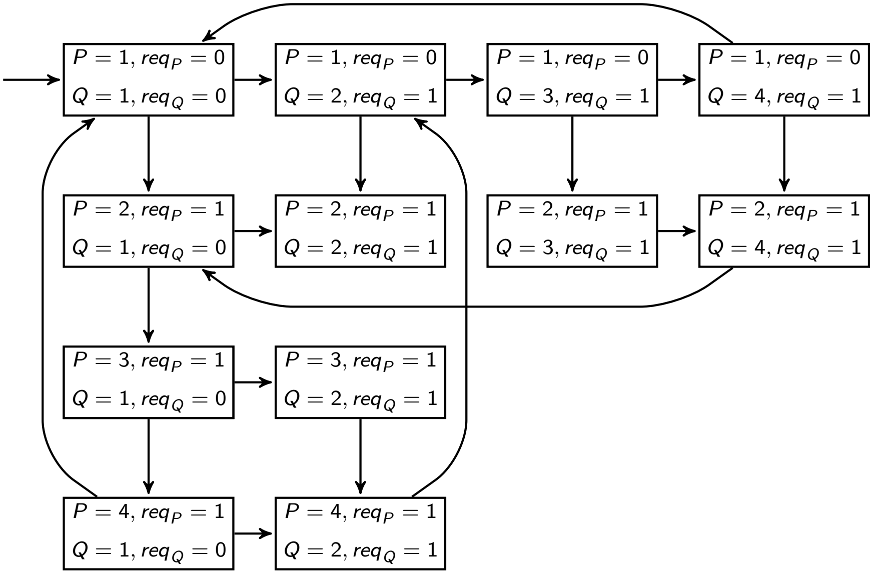
Exemple: espace d'état ou graphe d'accessibilité



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.

Propriété 2

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 $\boxed{\begin{array}{c} P = 2, req_P = 1 \\ Q = 2, req_Q = 1 \end{array}}$ 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).

Une formalisation

- 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

$r \wedge o \wedge v =$ , $r \wedge \neg o \wedge \neg v =$ , $\neg r \wedge \neg o \wedge v =$ , $\neg r \wedge \neg o \wedge \neg v =$ .

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

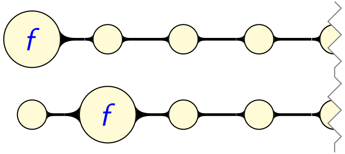
$\Rightarrow$ besoin de faire apparaître le temps

Operateurs LTL

Pour f et g deux formules propositionelles:

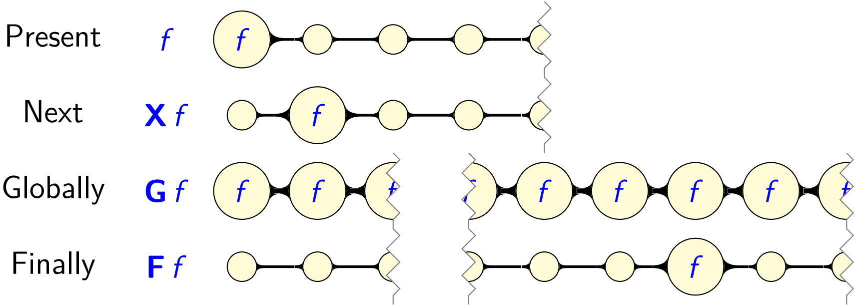
Present f

Next $\mathbf{X}\, f$



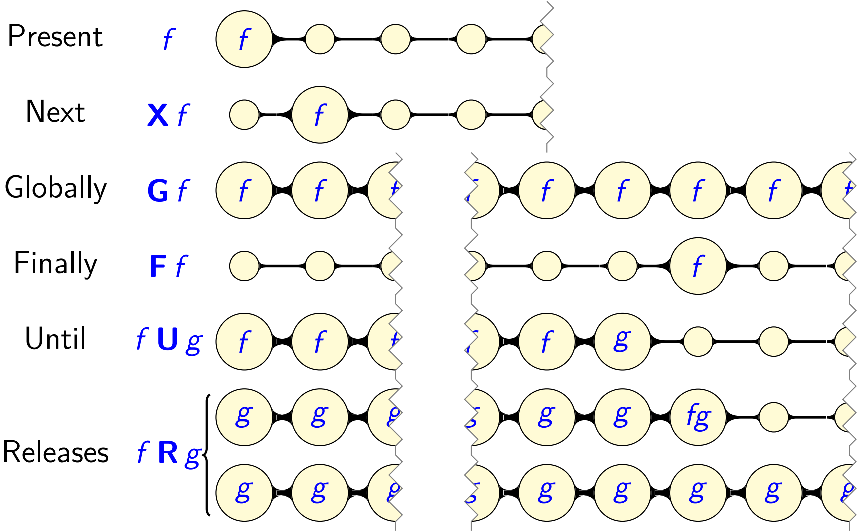
Operateurs LTL

Pour f et g deux formules propositionelles:



Operateurs LTL

Pour f et g deux formules propositionelles:



LTL: Exemples

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

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

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

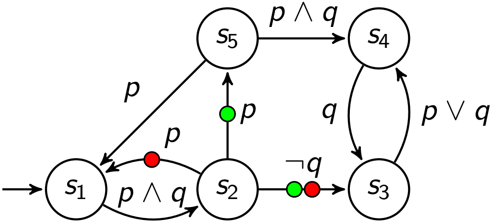
$\mathbf{G}\,\mathbf{F}(\neg r \wedge \neg o \wedge v)$: le système passe infiniment souvent par .

TGBA: Automates de Büchi généralisés

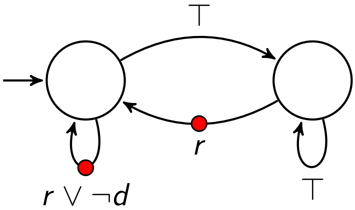
Un automate de Büchi généralisé étiqueté sur les transitions possède:

- un ensemble d'états, avec un état *initial*,
- un ensemble de transitions entre ces états, étiquetées par des formules de logique booléene,
- 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.

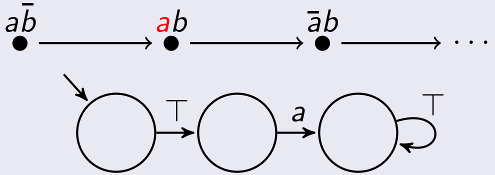


Exemple de TGBA reconnaissant $\mathbf{G}(d \rightarrow \mathbf{F}\, r)$

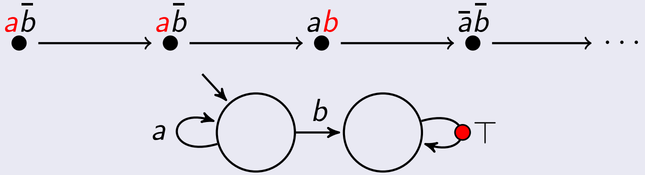


LTL et automates de Büchi

$\mathbf{X}\, a$

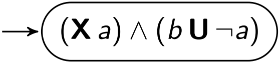


$a\, \mathbf{U}\, b$



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

Tableau Construction for LTL



1. Label the initial state by the formula to translate

Tableau Construction for LTL

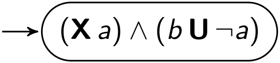
$\longrightarrow \boxed{(\mathbf{X}\, a) \wedge (b\, \mathbf{U} \neg a)}$

1. Label the initial state by the formula to translate
2. Rewrite each state label φ as

$$\bigvee_i \underbrace{\beta_i}_{\text{Boolean formula}} \wedge \mathbf{X}\, \underbrace{\psi_i}_{\text{LTL formula}}$$

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

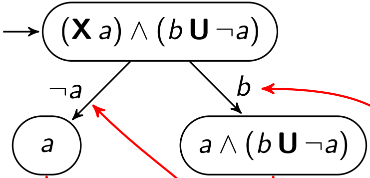


1. Label the initial state by the formula to translate
2. Rewrite each state label φ as

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

Since $f\, \mathbf{U}\, g = g \vee (f \wedge \mathbf{X}(f\, \mathbf{U}\, g))$ we have:
$(\mathbf{X}\, a) \wedge (b\, \mathbf{U} \neg a) = (\neg a \wedge \mathbf{X}\, \mathrm{a}) \vee (b \wedge \mathbf{X}(a \wedge (b\, \mathbf{U} \neg a)))$

Tableau Construction for LTL



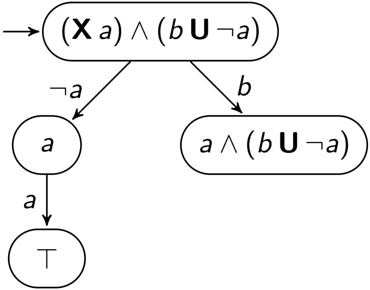
1. Label the initial state by the formula to translate
2. Rewrite each state label φ as

$$\bigvee_i \beta_i \land \mathbf{X}\, \psi_i$$

Then connect φ to each ψ_i using β_i as label

Since $f\, \mathbf{U}\, g = g \lor (f \land \mathbf{X}(f\, \mathbf{U}\, g))$ we have:
$(\mathbf{X}\, a) \land (b\, \mathbf{U}\, \neg a) = (\neg a \land \mathbf{X}\, \mathrm{a}) \lor (b \land \mathbf{X}(a \land (b\, \mathbf{U}\, \neg a)))$

Tableau Construction for LTL



1. Label the initial state by the formula to translate
2. Rewrite each state label φ as

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

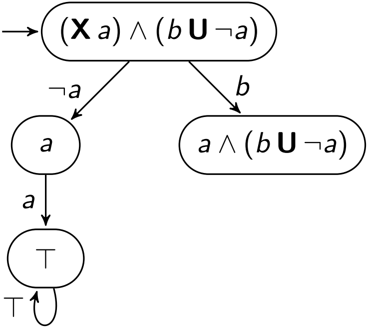
   Then connect φ to each ψ_i using β_i as label

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

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

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

Tableau Construction for LTL



1. Label the initial state by the formula to translate
2. Rewrite each state label φ as

$$\bigvee_i \beta_i \land \mathbf{X}\, \psi_i$$

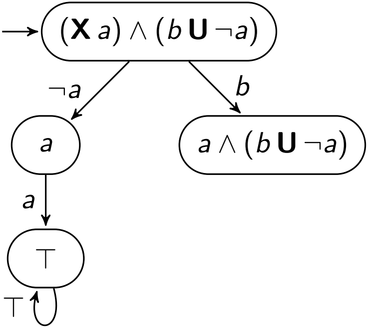
Then connect φ to each ψ_i using β_i as label

Since $f\, \mathbf{U}\, g = g \lor (f \land \mathbf{X}(f\, \mathbf{U}\, g))$ we have:

$(\mathbf{X}\, a) \land (b\, \mathbf{U} \neg a) = (\neg a \land \mathbf{X}\, \mathrm{a}) \lor (b \land \mathbf{X}(a \land (b\, \mathbf{U} \neg a)))$

$a = a \land \mathbf{X}\, \top$; $\top = \top \land \mathbf{X}\, \top$

Tableau Construction for LTL



1. Label the initial state by the formula to translate
2. Rewrite each state label φ as

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

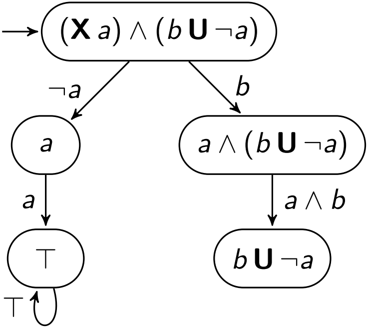
Then connect φ to each ψ_i using β_i as label

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

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

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

Tableau Construction for LTL



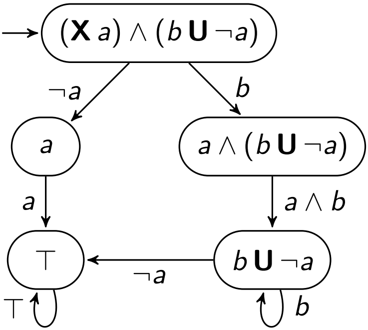
1. Label the initial state by the formula to translate
2. Rewrite each state label φ as

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

Then connect φ to each ψ_i using β_i as label

Since $f\, \mathbf{U}\, g = g \vee (f \wedge \mathbf{X}(f\, \mathbf{U}\, g))$ we have:
$(\mathbf{X}\, a) \wedge (b\, \mathbf{U}\, \neg a) = (\neg a \wedge \mathbf{X}\, \mathrm{a}) \vee (b \wedge \mathbf{X}(a \wedge (b\, \mathbf{U}\, \neg a)))$
$a = a \wedge \mathbf{X}\, \top$; $\top = \top \wedge \mathbf{X}\, \top$; $a \wedge (b\, \mathbf{U}\, \neg a) = a \wedge b \wedge \mathbf{X}\, (b\, \mathbf{U}\, \neg a)$

Tableau Construction for LTL



1. Label the initial state by the formula to translate
2. Rewrite each state label φ as

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

Then connect φ to each ψ_i using β_i as label

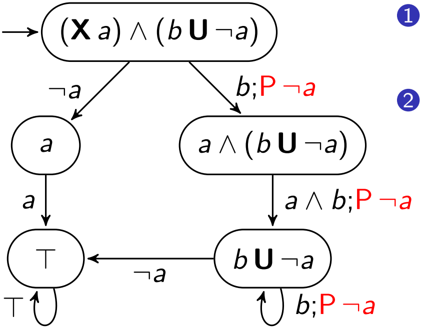
Since $f\,\mathbf{U}\,g = g \vee (f \wedge \mathbf{X}(f\,\mathbf{U}\,g))$ we have:

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

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

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

Tableau Construction for LTL



1. Label the initial state by the formula to translate
2. Rewrite each state label φ as
$$\bigvee_i \left(\beta_i \wedge \mathbf{X}\, \psi_i \wedge \bigwedge_j \mathrm{P}\, \gamma_{ij} \right)$$
Then connect φ to each ψ_i using β_i as label and $\{\mathrm{P}\, \gamma_{ij}\}_j$ as promises

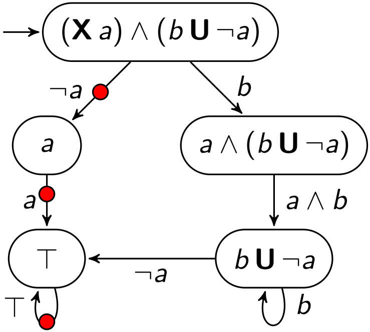
Since $f\, \mathbf{U}\, g = g \vee (f \wedge \mathbf{X}(f\, \mathbf{U}\, g) \wedge \mathrm{P}\, g)$ we have:

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

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

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

Tableau Construction for LTL



1. Label the initial state by the formula to translate
2. Rewrite each state label φ as
$$\bigvee_i \left(\beta_i \wedge \mathbf{X}\, \psi_i \wedge \bigwedge_j \mathrm{P}\, \gamma_{ij} \right)$$
Then connect φ to each ψ_i using β_i as label and $\{\mathrm{P}\,\gamma_{ij}\}_j$ as promises
3. Create Büchi acceptance sets complementing each promise

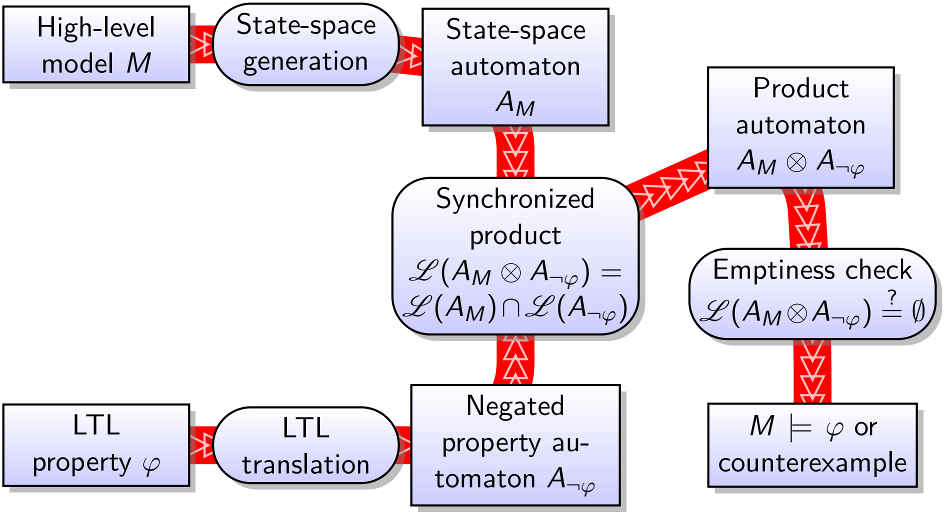
Since $f\,\mathbf{U}\,g = g \vee (f \wedge \mathbf{X}(f\,\mathbf{U}\,g) \wedge \mathrm{P}\, g)$ we have:

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

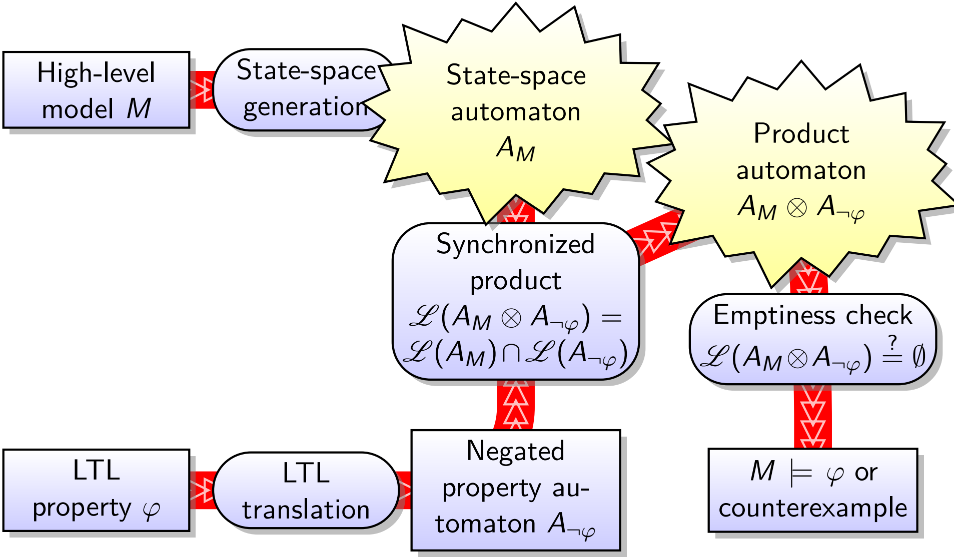
$a = a \wedge \mathbf{X}\, \top$; $\top = \top \wedge \mathbf{X}\, \top$; $a \wedge (b\,\mathbf{U}\,\neg a) = a \wedge b \wedge \mathbf{X}\, (b\,\mathbf{U}\,\neg a) \wedge \mathrm{P}\, \neg a$

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

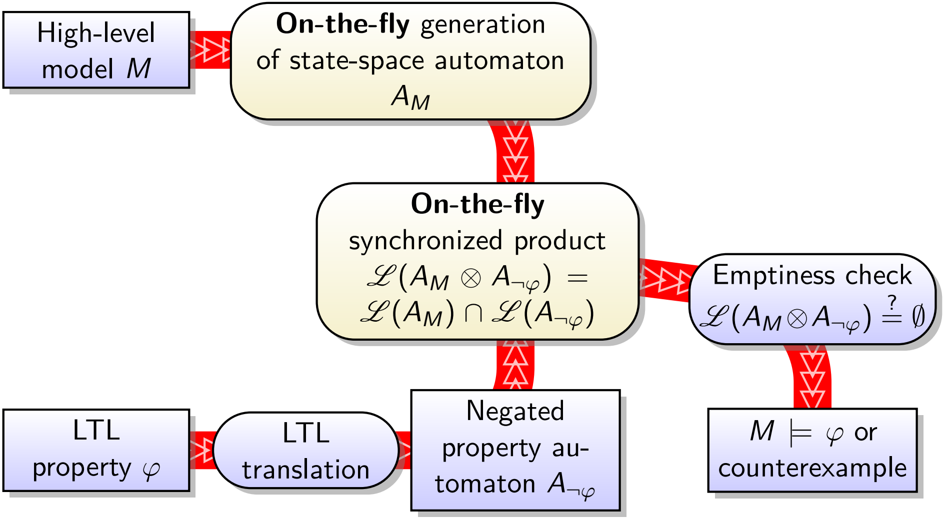
Automata-Theoretic LTL Model Checking



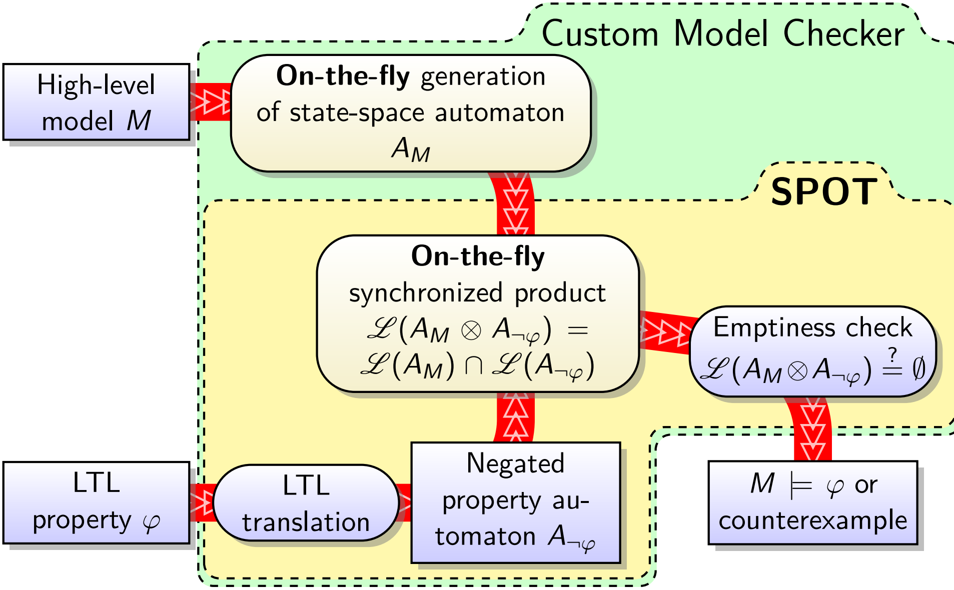
Automata-Theoretic LTL Model Checking



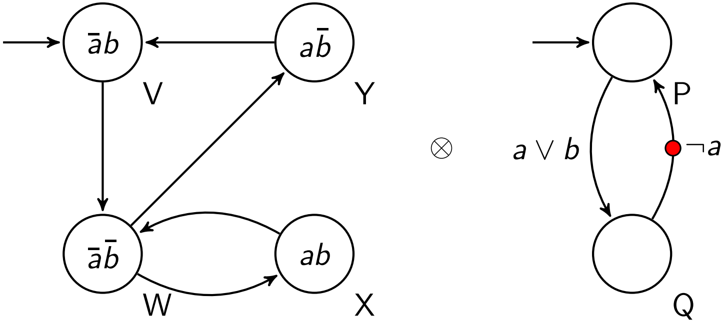
Automata-Theoretic LTL Model Checking



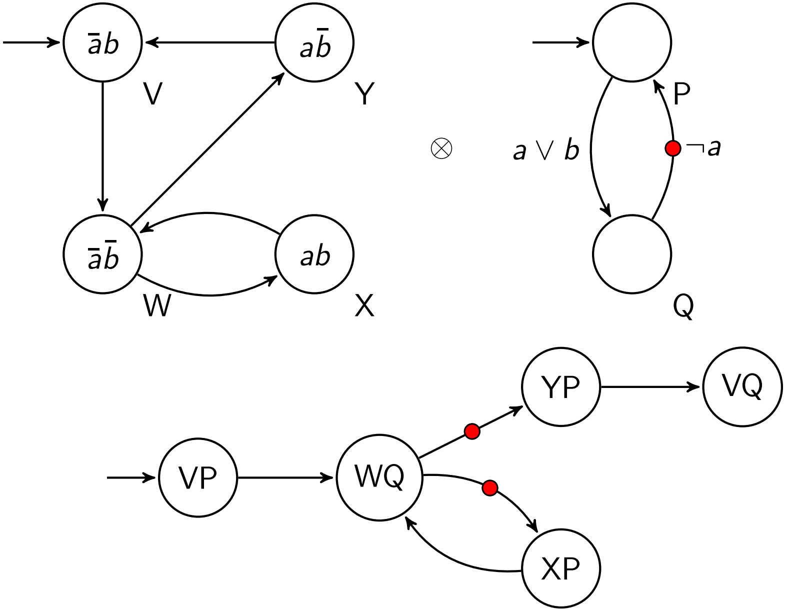
Automata-Theoretic LTL Model Checking



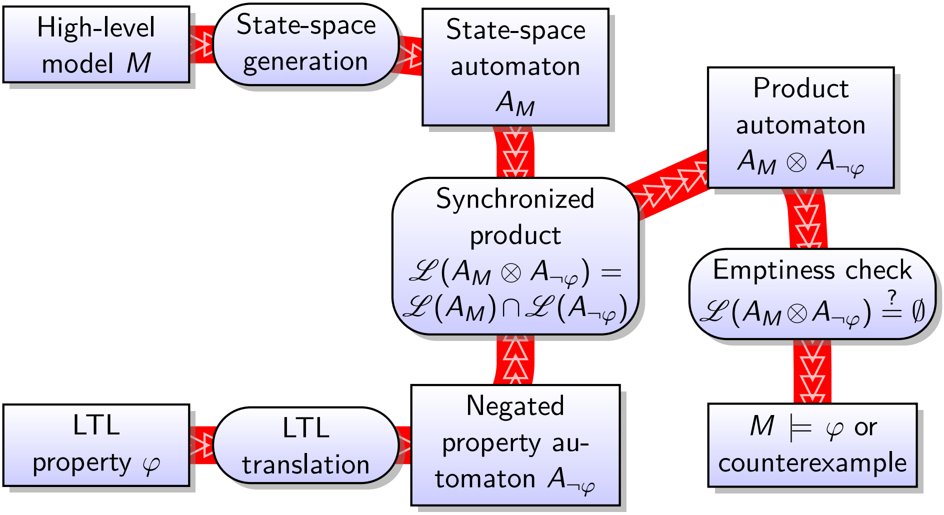
Produit entre structure de Kripke et TGBA



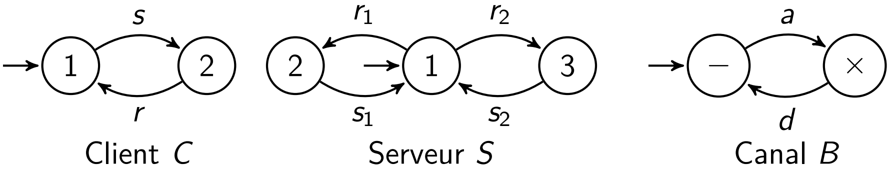
Produit entre structure de Kripke et TGBA



Automata-Theoretic LTL Model Checking



Ex.: clients/serveur par automates synchronisés

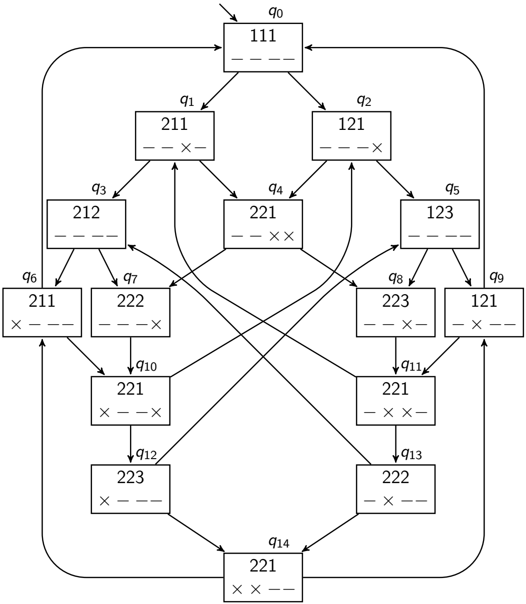


Règles de synchronisation pour le système $\langle C, C, S, B, B, B, B\rangle$:

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

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 ?

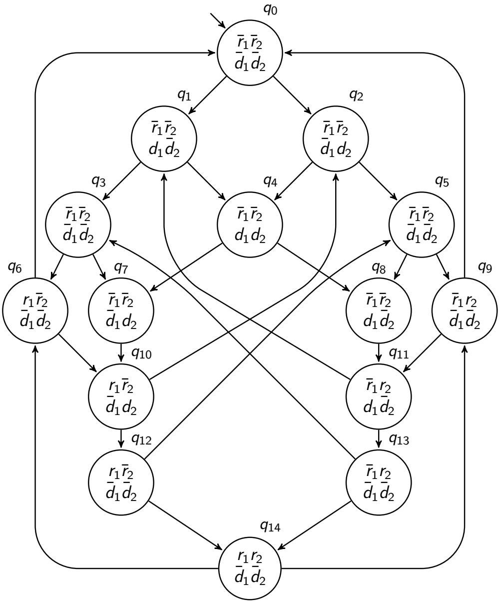
Propositions atomiques pour l'exemple

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

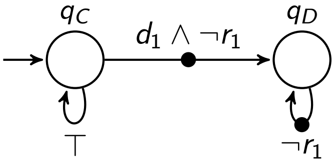


Exploration de la structure de Kripke

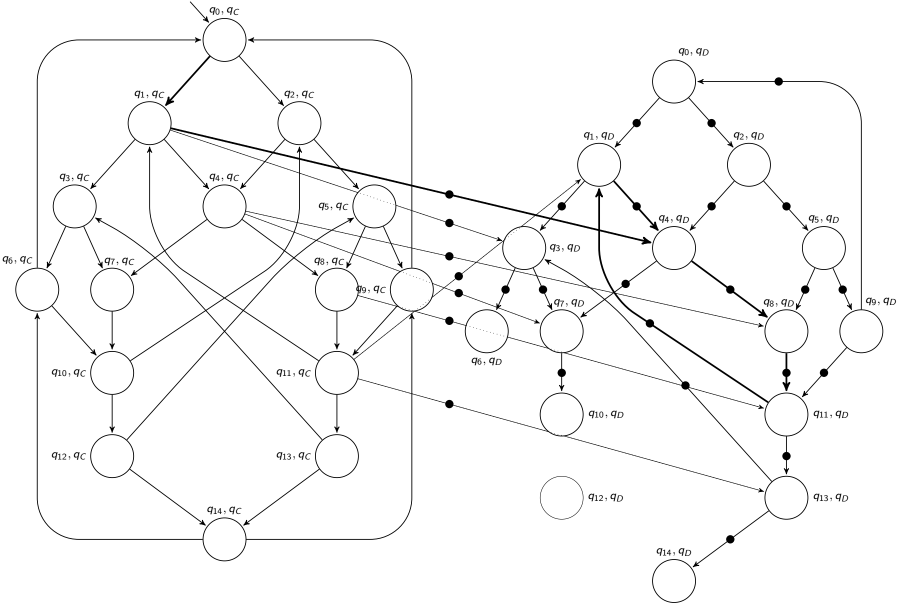
Pour tout $i \in \{1, 2\}$, si un état vérifie d_i alors dans tous ses futurs possibles il possède un successeur qui vérifie r_i.

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 $i = 1$.

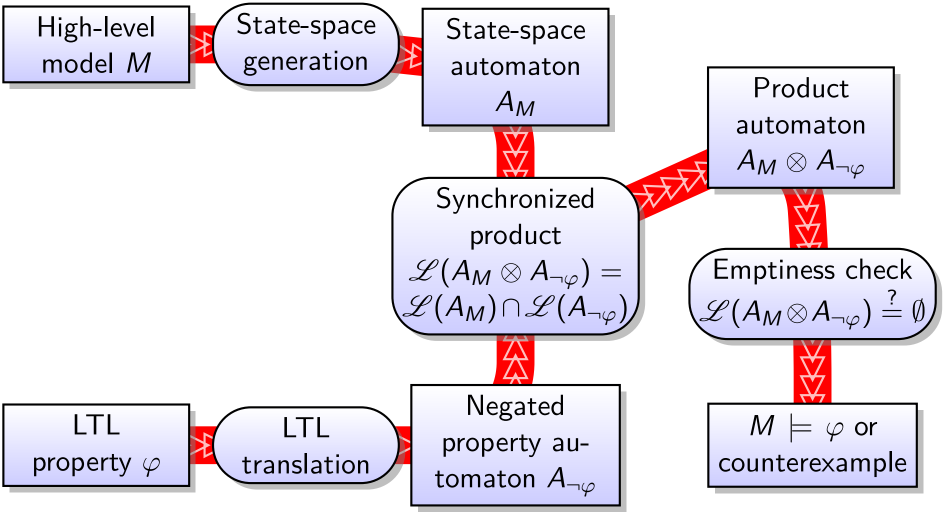
Le chemin qu'on voudrait reconnaître peut être reconnu par l'automate suivant, négation de la formule $\mathbf{G}(d_1 \rightarrow \mathbf{F}\, r_1)$:



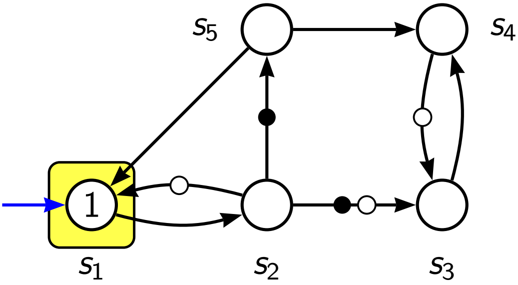
Produit structure de Kripke/Automate



Automata-Theoretic LTL Model Checking

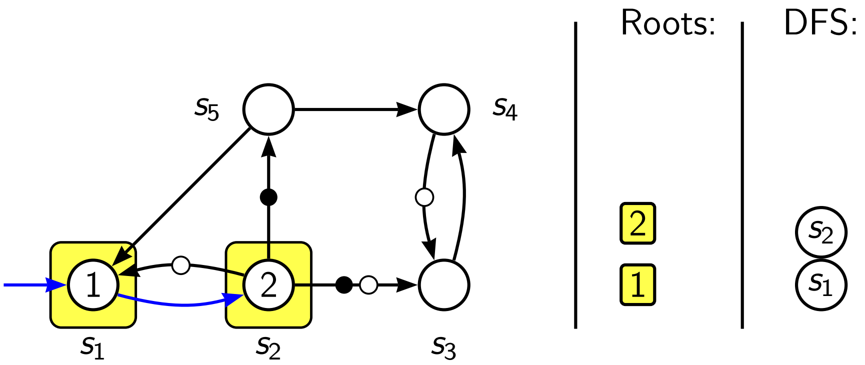


Emptiness check

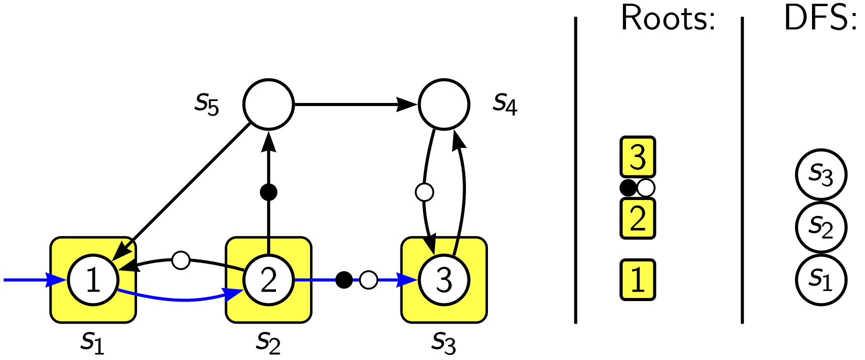


Roots:	DFS:
1	s_1

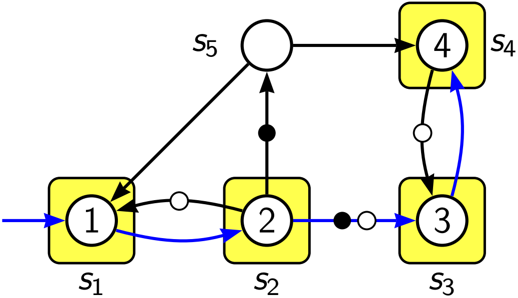
Emptiness check



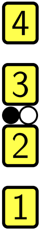
Emptiness check



Emptiness check

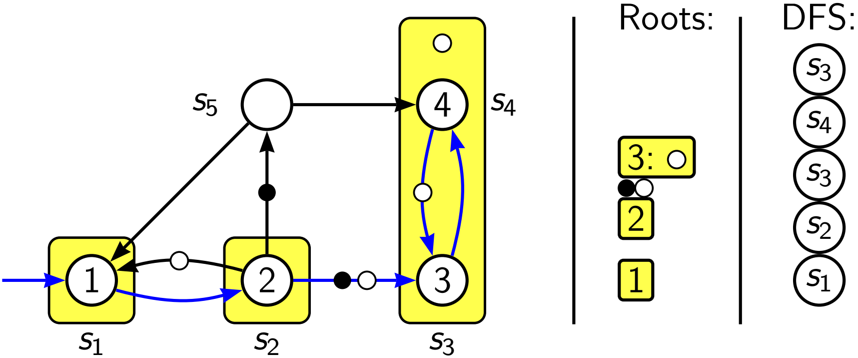


Roots:

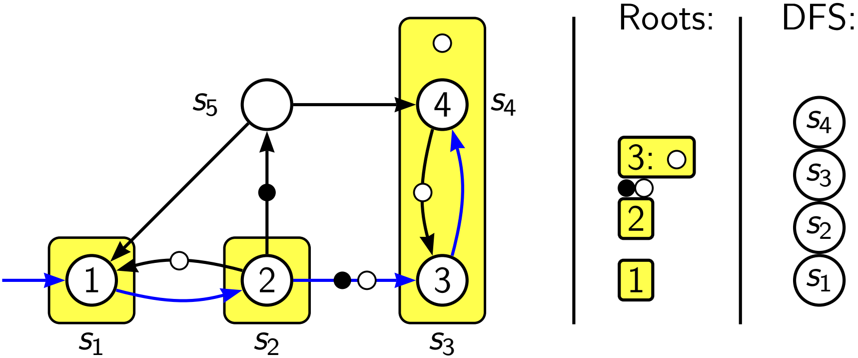


DFS:

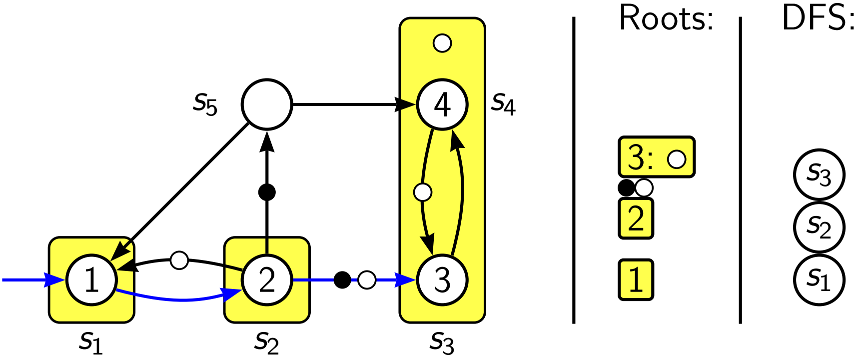
Emptiness check



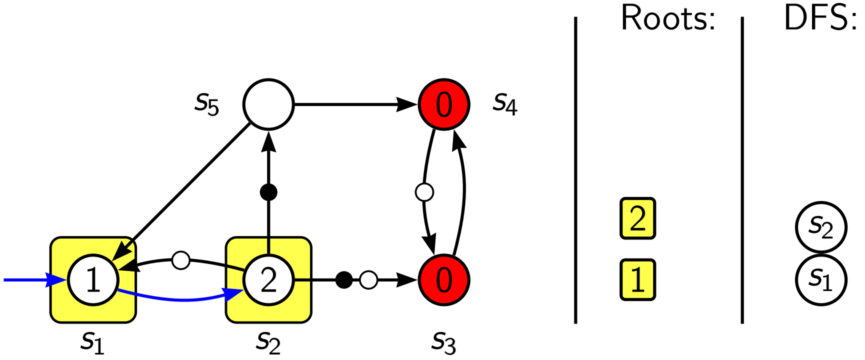
Emptiness check



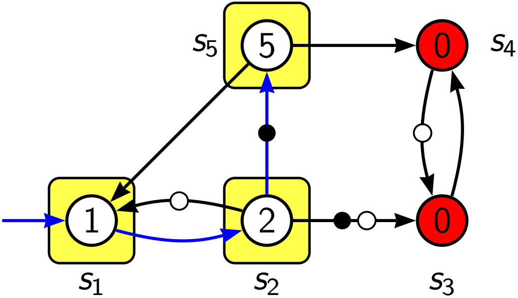
Emptiness check



Emptiness check



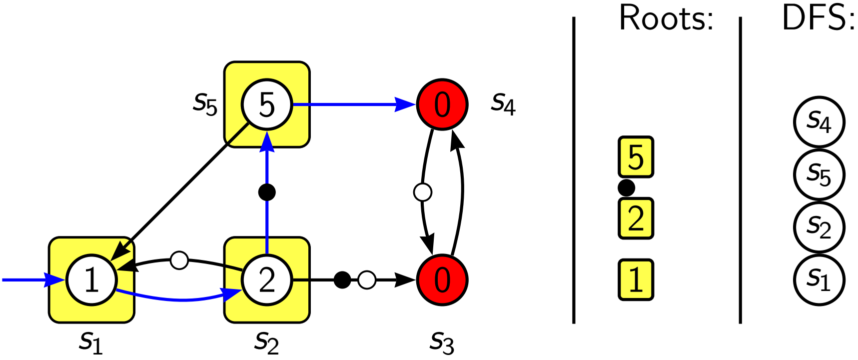
Emptiness check



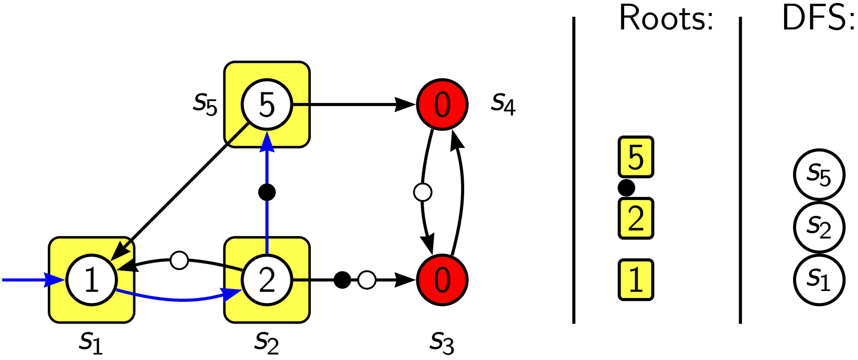
Roots:

DFS:

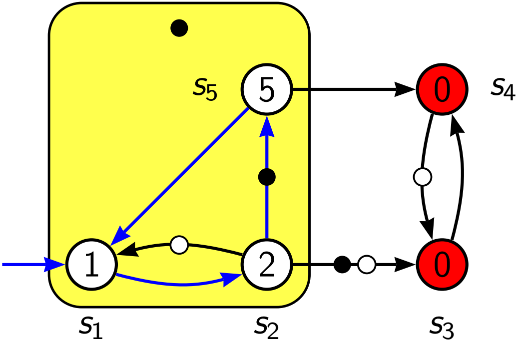
Emptiness check



Emptiness check



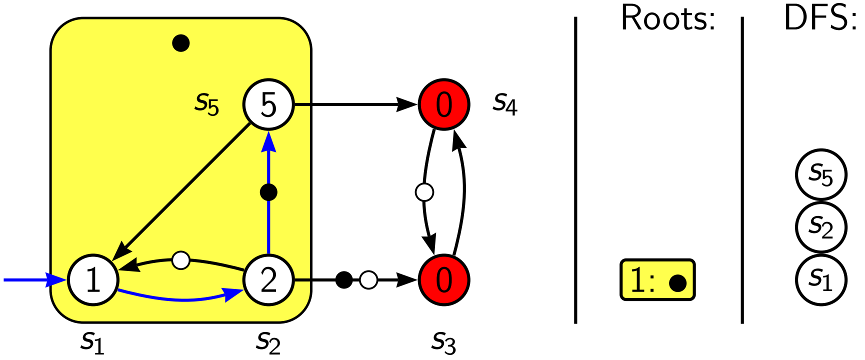
Emptiness check



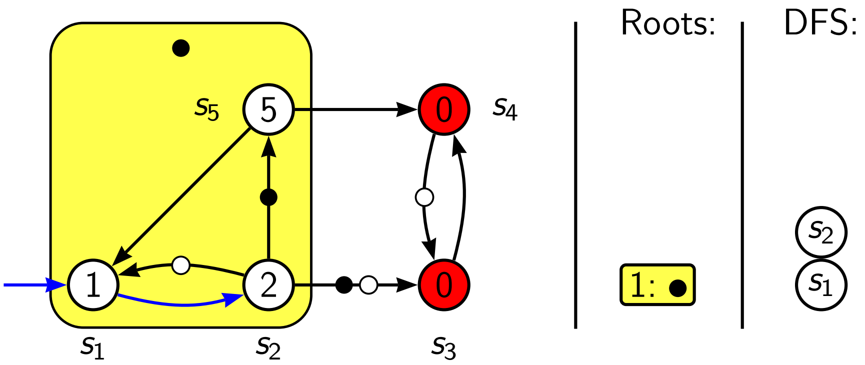
Roots:

DFS:

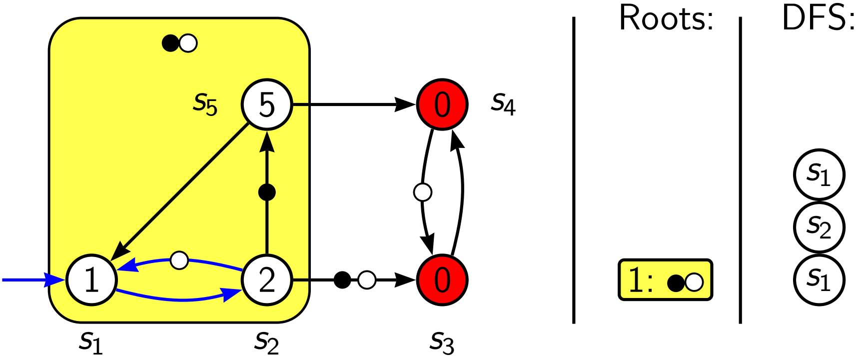
Emptiness check



Emptiness check

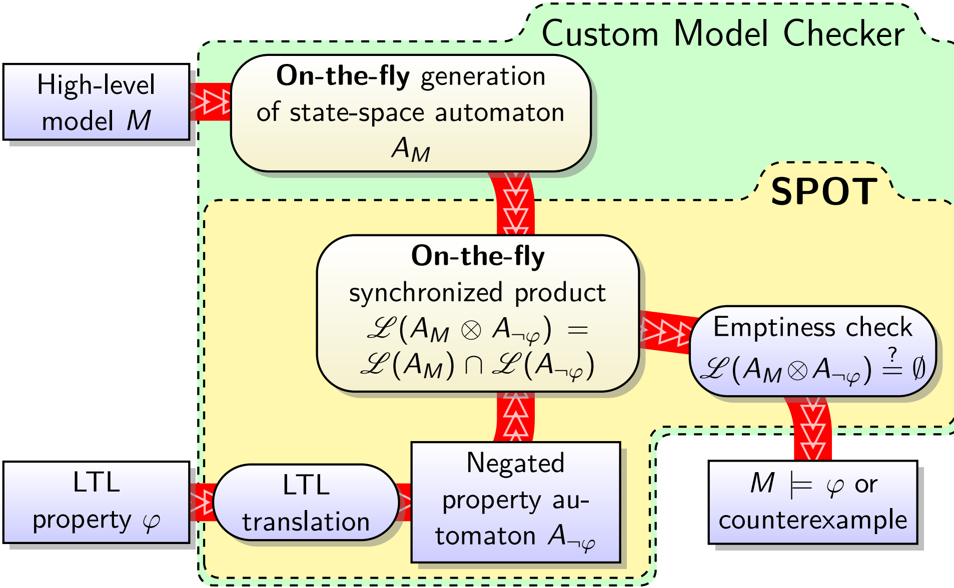


Emptiness check



Found !

Automata-Theoretic LTL Model Checking



Automata-Theoretic LTL Model Checking