Jeux et automates sur les ordres

Soutenance de thèse

Julien Cristau

LIAFA — Université Paris Diderot - Paris 7

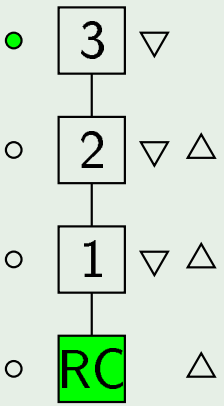
13 décembre 2010

Plan

Exemple

On abstrait un système en un nombre fini d'états.

Exemple

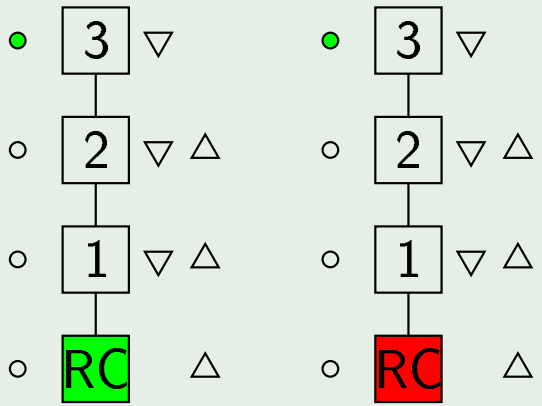


L'évolution du système au cours du temps est représentée par une suite d'états.
$(0, o)$

Exemple

On abstrait un système en un nombre fini d'états.

Exemple



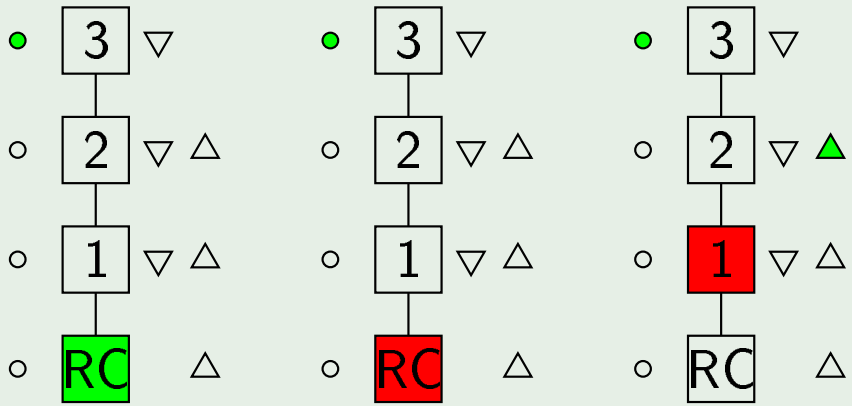
L'évolution du système au cours du temps est représentée par une suite d'états.

$(0, o) \xrightarrow{f} (0, f)$

Exemple

On abstrait un système en un nombre fini d'états.

Exemple



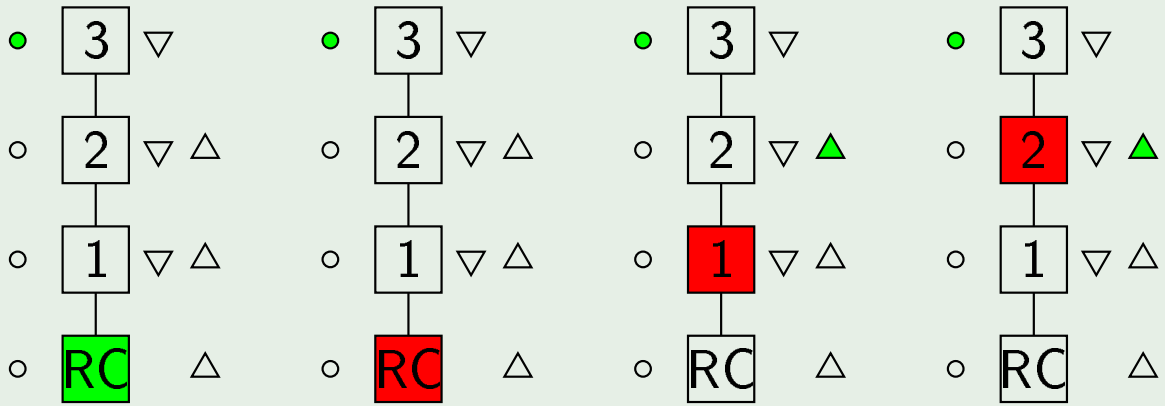
L'évolution du système au cours du temps est représentée par une suite d'états.

$(0, o) \xrightarrow{f} (0, f) \xrightarrow{+} (1, f)$

Exemple

On abstrait un système en un nombre fini d'états.

Exemple



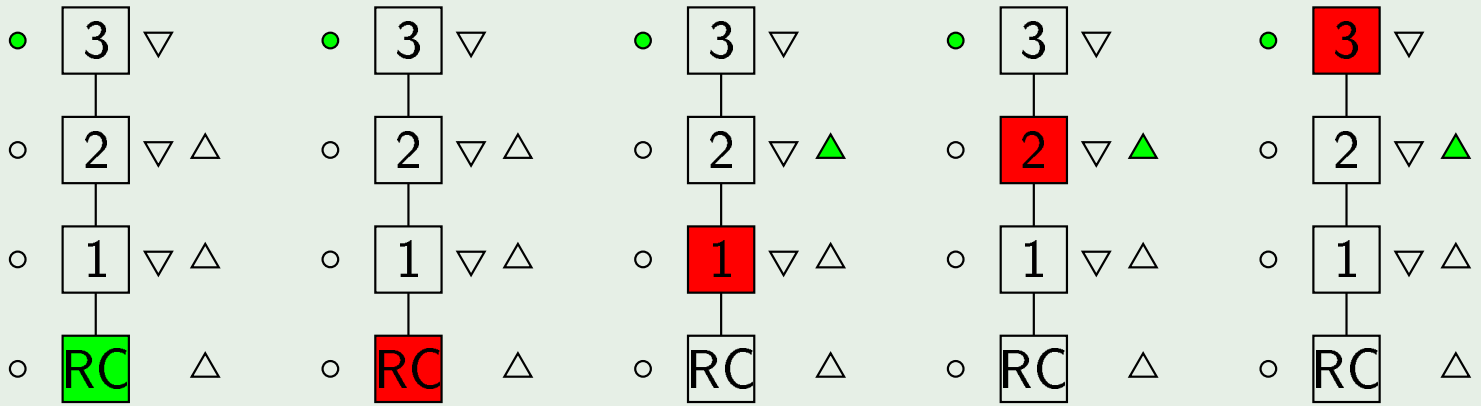
L'évolution du système au cours du temps est représentée par une suite d'états.

$(0, o) \xrightarrow{f} (0, f) \xrightarrow{+} (1, f) \xrightarrow{+} (2, f)$

Exemple

On abstrait un système en un nombre fini d'états.

Exemple



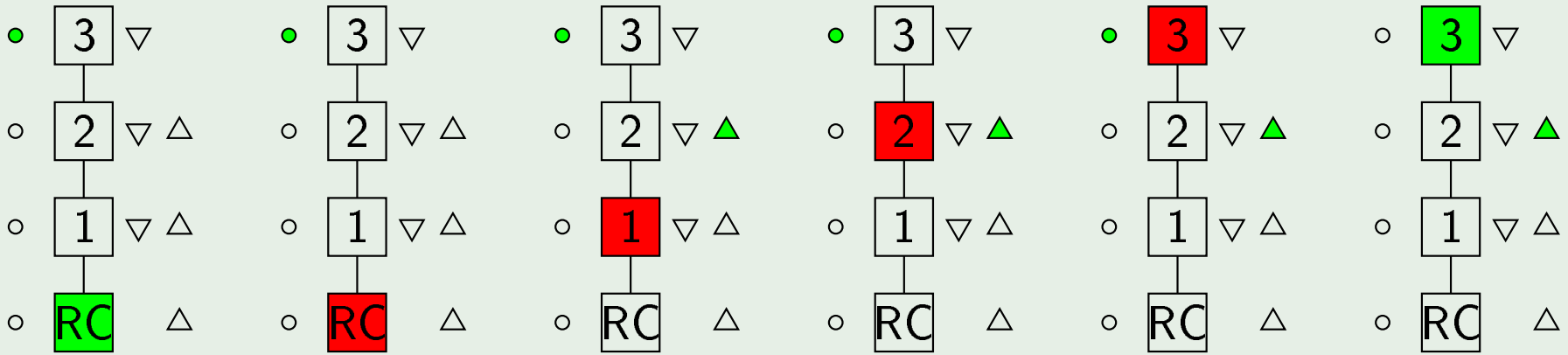
L'évolution du système au cours du temps est représentée par une suite d'états.

$(0, o) \xrightarrow{f} (0, f) \xrightarrow{+} (1, f) \xrightarrow{+} (2, f) \xrightarrow{+} (3, f)$

Exemple

On abstrait un système en un nombre fini d'états.

Exemple

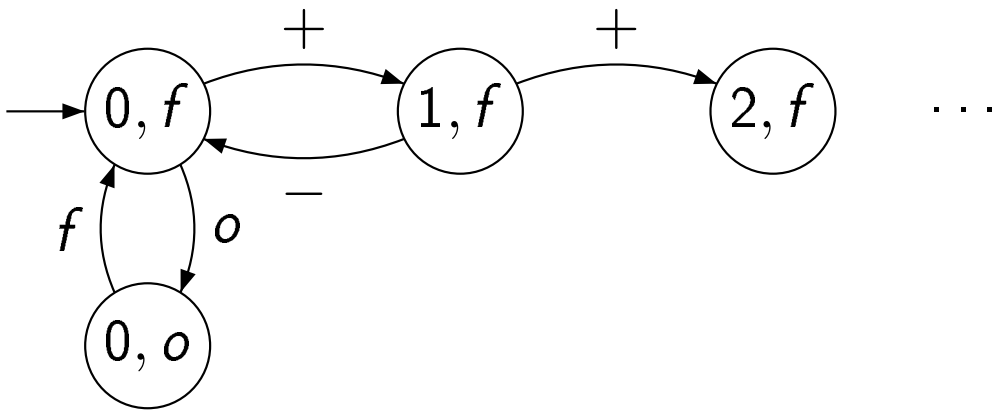


L'évolution du système au cours du temps est représentée par une suite d'états.

$(0, o) \xrightarrow{f} (0, f) \xrightarrow{+} (1, f) \xrightarrow{+} (2, f) \xrightarrow{+} (3, f) \xrightarrow{o} (3, o) \ldots$

Automate

Représentation du système sous la forme d'automate



Exécution : chemin

Exécutions possibles : langage reconnu par l'automate

Logique

Bon fonctionnement du système

- Si l'utilisateur veut aller au 3ème étage il sera atteint (avec les portes ouvertes) en temps fini
- On ne change jamais d'étage avec les portes ouvertes

Logique temporelle linéaire

Formules logiques spécifiant le comportement dans le temps

- $\texttt{Request}_3 \Rightarrow F(3 \wedge o)$
- $o \Rightarrow (\neg + \wedge \neg -)\mathcal{U} f$

Until $(\varphi \mathcal{U} \psi)$

Mots

Alphabet fini $\{a, b, c, d, e, r\}$

Les mots peuvent être :

- finis *abracadabra*, *barre*
- infinis *abracadabraabracadabraabra* $\ldots$, $(ab)^\omega$
- ordinaux $((ab)^\omega bra(bc)^\omega)^\omega$

$$|a|b|a|b|a\cdots \mid b|r|a|b|c|b|c|b\cdots \mid a|b|a|b\cdots \mid b|r|a|b|c|b\cdots \mid \cdots$$

- **cas général**
  - **ensemble totalement ordonné de positions** J
  - **étiquetage** $w = (w_j)_{j\in J}$
- $$\begin{array}{rccl} w: & \mathbb{R} & \to & \{a,b\} \\ & j & \mapsto & \begin{cases} a & \text{si } j \in \mathbb{Q} \\ b & \text{sinon} \end{cases} \end{array}$$

Objectifs

Modèles

Mots indexés par des ordres linéaires quelconques

Formalismes

- Automates finis
- Logique temporelle linéaire

Questions

- Vide du langage reconnu par un automate
- Satisfaisabilité de φ
- Liens entre logique et automates

Objectifs

Modèles

Mots indexés par des ordres linéaires quelconques

Formalismes

- Automates finis
- Logique temporelle linéaire

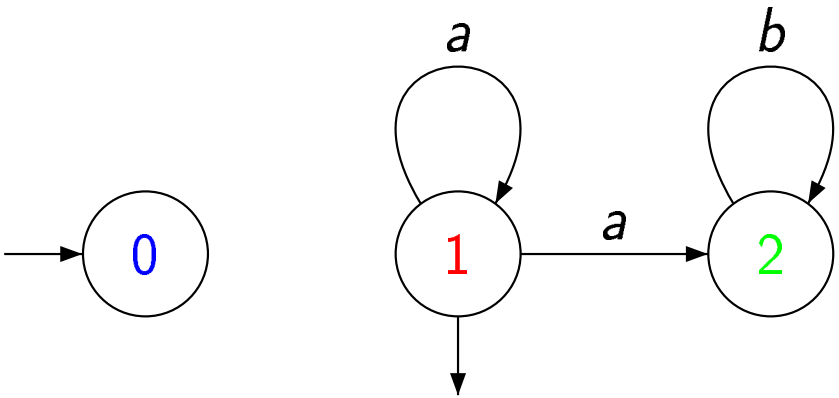
Résultat principal

Satisfaisabilité de LTL

Transformation d'une formule en un automate

Automates sur les ordres

Transitions limites pour gérer les positions sans prédécesseur ou sans successeur.



$\{2\} \to 1$

$0 \to \{1, 2\}$

$$\underset{0}{|} \ldots \underset{1}{|a|}\underset{2}{b|}\underset{2}{b|}\underset{2}{b|}\underset{2}{b|}\underset{2}{b|} \ldots \underset{1}{|a|}\underset{1}{a|}\underset{1}{a|}\underset{2}{b|}\underset{2}{bbb} \ldots \underset{1}{|aabbbb} \ldots \underset{1}{|}$$

L'automate ci-dessus reconnaît $(a^* a b^\omega)^{-\omega}$

Problème du vide

On peut décider l'accessibilité en temps polynomial.

LTL sur les ordres

Syntaxe

$$\varphi, \psi \quad ::= \quad a \mid \varphi \vee \psi \mid \neg\varphi \mid \varphi \mathcal{U} \psi \mid \varphi \mathcal{S} \psi \mid \varphi \mathcal{U}' \psi \mid \varphi \mathcal{S}' \psi$$

$\mathcal{U}'$ et $\mathcal{S}'$ sont nécessaires pour les ordres non complets (par exemple $\mathbb{Q}$) et permettent de parler des « trous ».

Until ($\varphi \mathcal{U} \psi$)

Stavi ($\varphi \mathcal{U}' \psi$)

LTL sur les ordres

Syntaxe

$$\varphi, \psi \quad ::= \quad a \mid \varphi \vee \psi \mid \neg\varphi \mid \varphi \mathcal{U} \psi \mid \varphi \mathcal{S} \psi \mid \varphi \mathcal{U}' \psi \mid \varphi \mathcal{S}' \psi$$

$\mathcal{U}'$ et $\mathcal{S}'$ sont nécessaires pour les ordres non complets (par exemple $\mathbb{Q}$) et permettent de parler des « trous ».

Exemple ($\varphi = (a \vee b)\mathcal{U}c$)

$$w = abcabda^{\omega}ac(ac)^{-\omega} \quad = \quad abcabdaa \ldots \; ac \; \ldots acac$$

Satisfaisabilité

Existe-t-il un mot w et une position de w tels que φ est vraie ?

LTL sur les ordres

Syntaxe

$$\varphi, \psi \quad ::= \quad a \mid \varphi \vee \psi \mid \neg\varphi \mid \varphi \mathcal{U} \psi \mid \varphi \mathcal{S} \psi \mid \varphi \mathcal{U}' \psi \mid \varphi \mathcal{S}' \psi$$

$\mathcal{U}'$ et $\mathcal{S}'$ sont nécessaires pour les ordres non complets (par exemple $\mathbb{Q}$) et permettent de parler des « trous ».

Exemple ($\varphi = (a \vee b)\mathcal{U}c$)

$$w = abcabda^{\omega}ac(ac)^{-\omega} \quad = \quad abcabdaa\ldots \underset{1}{ac} \ldots acac$$

Satisfaisabilité

Existe-t-il un mot w et une position de w tels que φ est vraie ?

LTL sur les ordres

Syntaxe

$$\varphi, \psi \quad ::= \quad a \mid \varphi \vee \psi \mid \neg\varphi \mid \varphi \mathcal{U} \psi \mid \varphi \mathcal{S} \psi \mid \varphi \mathcal{U}' \psi \mid \varphi \mathcal{S}' \psi$$

$\mathcal{U}'$ et $\mathcal{S}'$ sont nécessaires pour les ordres non complets (par exemple $\mathbb{Q}$) et permettent de parler des « trous ».

Exemple ($\varphi = (a \vee b)\mathcal{U}c$)

$$w = abcabda^{\omega}ac(ac)^{-\omega} \quad = \quad abcabdaa\ldots \underset{1}{ac} \ldots \underset{1}{acac}$$

Satisfaisabilité

Existe-t-il un mot w et une position de w tels que φ est vraie ?

LTL sur les ordres

Syntaxe

$$\varphi, \psi \quad ::= \quad a \mid \varphi \vee \psi \mid \neg\varphi \mid \varphi \mathcal{U} \psi \mid \varphi \mathcal{S} \psi \mid \varphi \mathcal{U}' \psi \mid \varphi \mathcal{S}' \psi$$

$\mathcal{U}'$ et $\mathcal{S}'$ sont nécessaires pour les ordres non complets (par exemple $\mathbb{Q}$) et permettent de parler des « trous ».

Exemple ($\varphi = (a \vee b)\mathcal{U}c$)

$$w = abcabda^{\omega}ac(ac)^{-\omega} \quad = \quad abcabdaa\ldots \; ac \; \ldots acac$$
$$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad 1 \qquad 1 \qquad 1$$

Satisfaisabilité

Existe-t-il un mot w et une position de w tels que φ est vraie ?

LTL sur les ordres

Syntaxe

$$\varphi, \psi \quad ::= \quad a \mid \varphi \vee \psi \mid \neg\varphi \mid \varphi \mathcal{U} \psi \mid \varphi \mathcal{S} \psi \mid \varphi \mathcal{U}' \psi \mid \varphi \mathcal{S}' \psi$$

$\mathcal{U}'$ et $\mathcal{S}'$ sont nécessaires pour les ordres non complets (par exemple $\mathbb{Q}$) et permettent de parler des « trous ».

Exemple ($\varphi = (a \vee b)\mathcal{U}c$)

$$w = abcabda^{\omega}ac(ac)^{-\omega} \quad = \quad abcabdaa\ldots \; ac \; \ldots acac$$
$$1 \qquad 1 \qquad 10$$

Satisfaisabilité

Existe-t-il un mot w et une position de w tels que φ est vraie ?

LTL sur les ordres

Syntaxe

$$\varphi, \psi \quad ::= \quad a \mid \varphi \vee \psi \mid \neg\varphi \mid \varphi \mathcal{U} \psi \mid \varphi \mathcal{S} \psi \mid \varphi \mathcal{U}' \psi \mid \varphi \mathcal{S}' \psi$$

$\mathcal{U}'$ et $\mathcal{S}'$ sont nécessaires pour les ordres non complets (par exemple $\mathbb{Q}$) et permettent de parler des « trous ».

Exemple ($\varphi = (a \vee b)\mathcal{U}c$)

$$w = abcabda^{\omega}ac(ac)^{-\omega} \quad = \quad \begin{array}{l} abcabdaa\ldots\ ac\ \ldots acac \\ 11000111\ldots\ 10\ \ldots 1110 \end{array}$$

Satisfaisabilité

Existe-t-il un mot w et une position de w tels que φ est vraie ?

Résultat

Théorème

Pour toute formule LTL φ et tout langage rationnel L de mots sur les ordres, on peut décider s'il existe un mot $w \in L$ et une position i de w tels que $w, i \models \varphi$.

Idée

- Construire un transducteur $\mathcal{A}_\varphi$: $\mathcal{A}_\varphi(w)$ est le mot de vérité de φ sur w
- Décider SAT(φ) $\simeq$ problème d'accessibilité sur $\mathcal{A}_\varphi$

Construction

- Transducteurs pour
  - prédicats atomiques
  - connecteurs logiques
  - opérateurs temporels
- Opérations sur les tranducteurs
  - produit
  - composition

Pour toute formule φ on construit $\mathcal{A}_\varphi$

- entrée : mot w
- sortie : mot de vérité de φ sur w

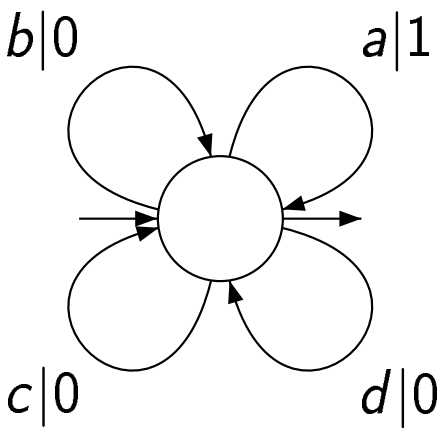
Transducteurs

w $\quad abcabdaa \ldots \; ac \; \ldots acac$

$(a \vee b)\mathcal{U}c$ $\quad 11000111 \ldots \; 10 \; \ldots 1110$

Transducteurs

w $abcabdaa \ldots ac \ldots acac$



a $10010011 \ldots 10 \ldots 1010$

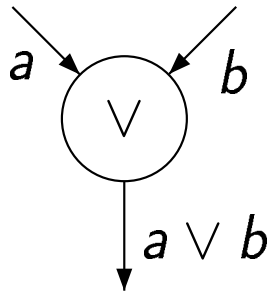
$(a \vee b)\mathcal{U}c$ $11000111 \ldots 10 \ldots 1110$

Transducteurs

w	$abcabdaa\ldots$	ac	$\ldots acac$
a	$10010011\ldots$	10	$\ldots 1010$
b	$01001000\ldots$	00	$\ldots 0000$
$(a \vee b)\mathcal{U}c$	$11000111\ldots$	10	$\ldots 1110$

Transducteurs

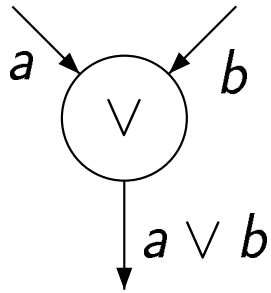
w	$abcabdaa \ldots ac \ldots acac$
a	$10010011 \ldots 10 \ldots 1010$
b	$01001000 \ldots 00 \ldots 0000$



$a \vee b$	$11011011 \ldots 10 \ldots 1010$
$(a \vee b)\mathcal{U}c$	$11000111 \ldots 10 \ldots 1110$

Transducteurs

w $\quad abcabdaa \ldots \; ac \; \ldots acac$



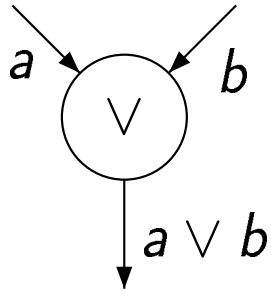
$a \vee b$ $\quad 11011011 \ldots \; 10 \; \ldots 1010$

c $\quad 00100000 \ldots \; 01 \; \ldots 0101$

$(a \vee b)\mathcal{U}c$ $\quad 11000111 \ldots \; 10 \; \ldots 1110$

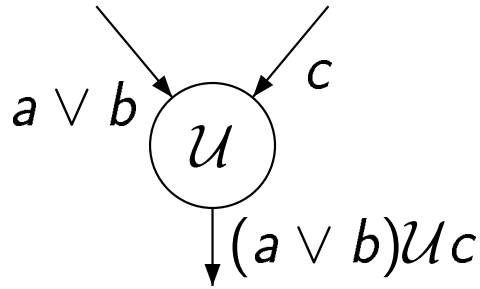
Transducteurs

w $\quad abcabdaa \ldots\ ac\ \ldots acac$



$a \vee b$ $\quad 11011011 \ldots\ 10\ \ldots 1010$

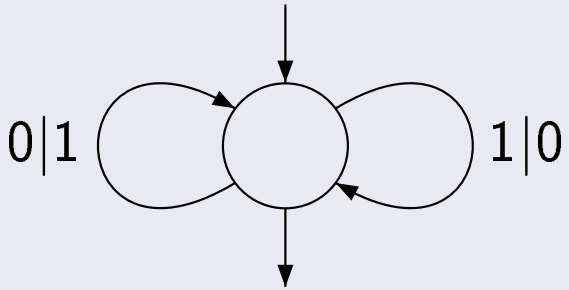
c $\quad 00100000 \ldots\ 01\ \ldots 0101$



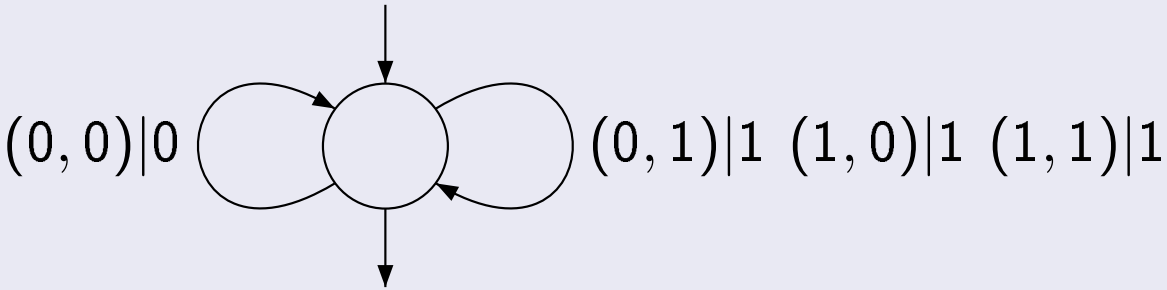
$(a \vee b)\mathcal{U}c$ $\quad 11000111 \ldots\ 10\ \ldots 1110$

Opérateurs logiques

Négation : entrée φ, sortie $\neg\varphi$

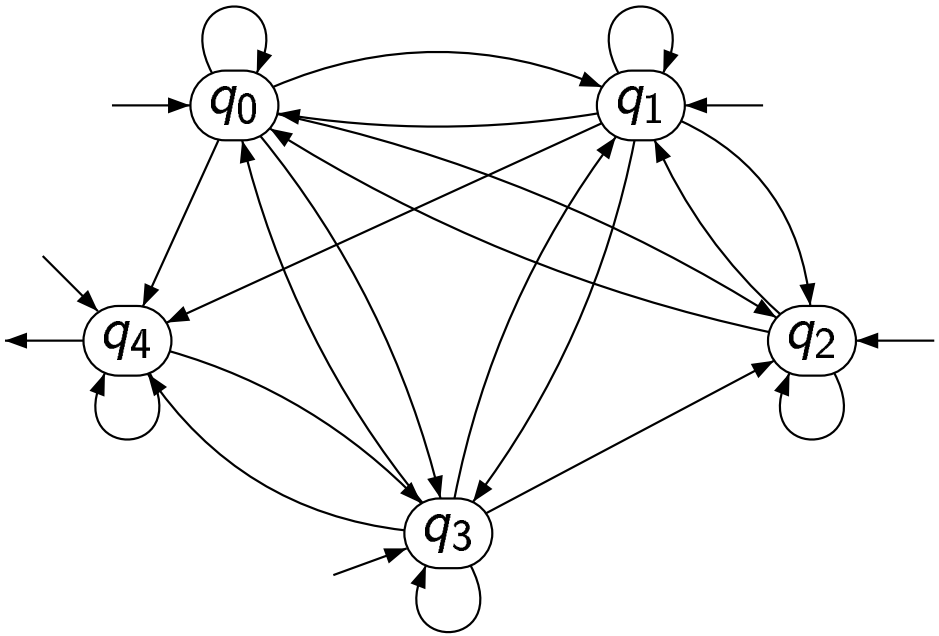


Disjonction : entrée (φ, ψ), sortie $\varphi \vee \psi$



Opérateurs temporels (1)

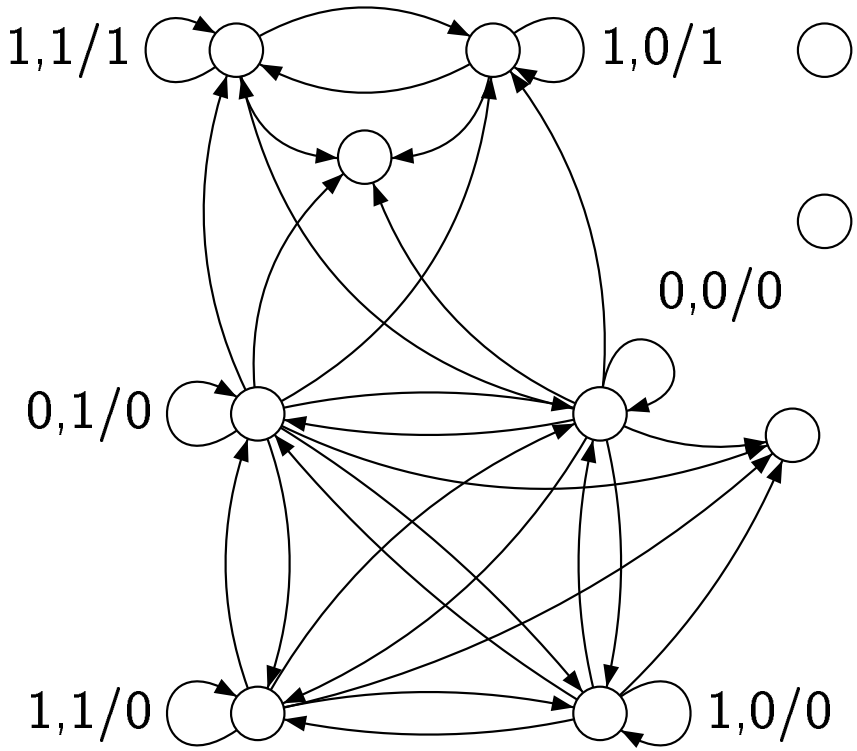
Until : 



Entrée (φ, ψ), sortie $\varphi \mathcal{U} \psi$

Opérateurs temporels (2)

Stavi :



Entrée (φ, ψ), sortie $\varphi \mathcal{U}' \psi$

Propriétés du transducteur

Par composition et produit de transducteurs de base on obtient $\mathcal{A}_\varphi$ tel que :

- le mot de sortie est l'évaluation de φ à chaque position de w
- pour tout mot d'entrée w il existe un et un seul chemin acceptant (donc un seul mot de sortie)

φ est satisfiable si et seulement s'il existe un chemin acceptant dans $\mathcal{A}_\varphi$ qui passe par une transition dont la sortie est 1

Complexité et perspectives

Complexité

- La construction explicite donnée ici est 2-EXPSPACE
- Rabinovich s'est servi de cette construction pour obtenir une borne PSPACE

Expressivité

Les automates (même non ambigus) permettent de décrire des langages au-delà de LTL.

Jeux de longueur ordinale

Pas vraiment de modèle existant. Comment définir des parties ordinales :

- arène finie
- automate avec contrôle partagé

Questions qui se posent :

- les jeux sont-ils déterminés ?
- si oui, peut-on trouver le vainqueur ?
- quelles sont les stratégies ?

Modèle

Pour passer à des parties ordinales on ajoute des transitions limites.
Condition de victoire : accessibilité (pour Eve), sûreté (pour Adam)

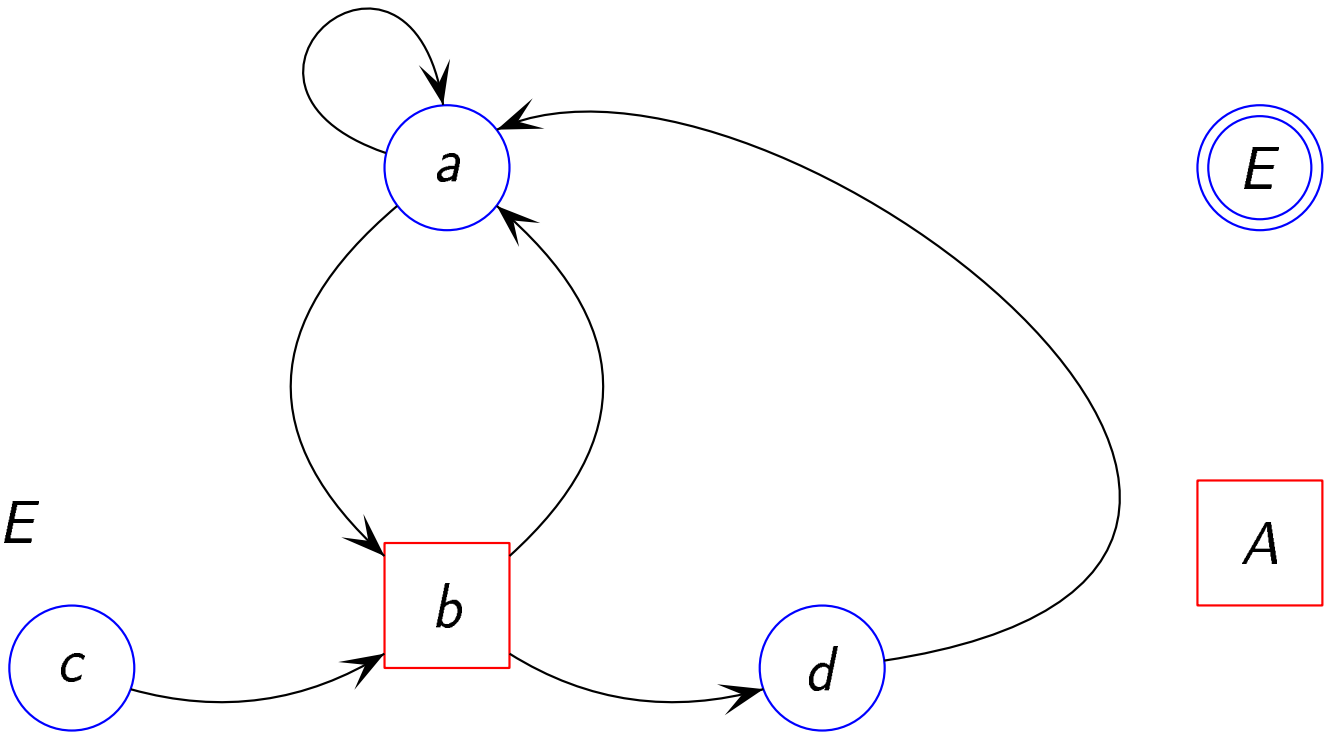
$\{a\} \to c$

$\{a, b\} \to d$

$\{a, b, c\} \to A$

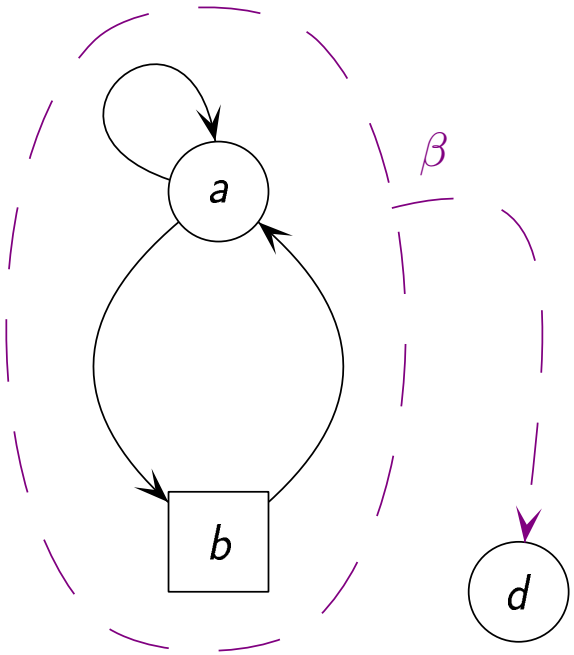
$\{a, b, d\} \to A$

$\{a, b, c, d\} \to E$



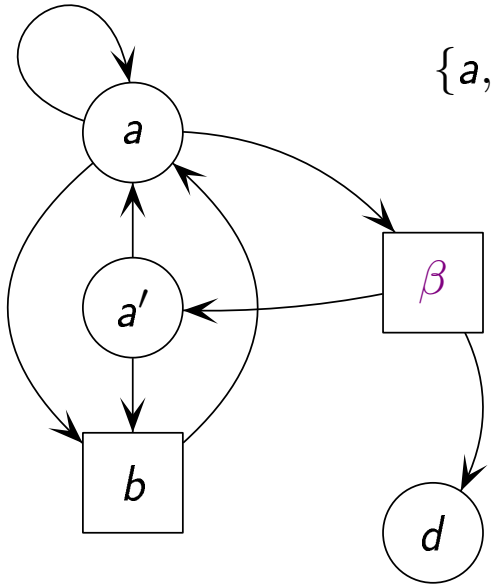
Réduction

Jeu ordinal



Condition de victoire : accessibilité
Longueur des parties : ordinal

Jeu de Muller



$\{a, a', b, \beta\} \in \mathcal{F}$

Condition de victoire : Muller
Longueur des parties : ω

Résultats

Théorème

Ces jeux sont déterminés et on peut déterminer le vainqueur en PSPACE.

Idée

Réduction vers un jeu de Muller et traduction des stratégies.

Stratégies

Les stratégies données par l'algorithme doivent se souvenir d'un préfixe d'un jeu de Muller : mémoire non bornée.

Peut-on obtenir des stratégies à mémoire finie ?

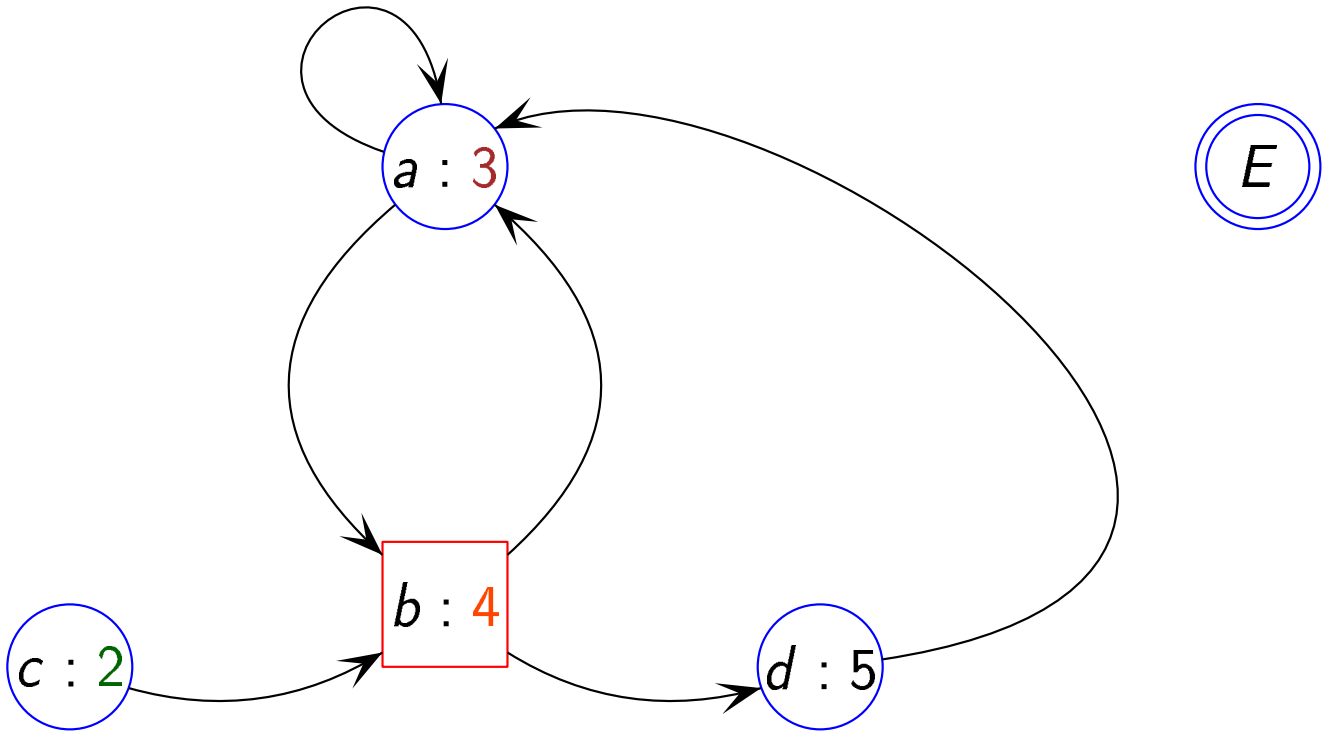
Jeux sur les ordinaux avec priorités

Les transitions limites sont définies par les priorités

$2 \to E$

$3 \to d$

$4 \to c$



Condition de victoire : Eve doit atteindre E

Résultats

Théorème

Les jeux de priorité sont positionnels.

- Adaptation de l'algorithme de Zielonka pour les jeux de parité ($\mathrm{NP} \cap \mathrm{coNP}$).

Théorème

Dans le cas général le vainqueur peut gagner avec une mémoire finie.

- Réduction vers des jeux de priorité (à partir des LAR)
- Algorithme EXPSPACE pour décider le vainqueur et construire une stratégie gagnante.

Résumé et perspectives

Résumé des contributions

- Modèle de jeux ordinaux sur des graphes finis
- Algorithme PSPACE pour décider le vainqueur
- Algorithme EXPSPACE pour calculer des stratégies gagnantes à mémoire finie

Conditions de victoire

Conditions plus complexes qu'accessibilité et sûreté

Jeux temporisés

Est-ce qu'on peut ajouter des horloges explicites ?

Conclusion

Automates et logique sur les ordres (FSTTCS'09)

- traduction d'une formule LTL en un transducteur qui calcule le mot de vérité
- accessibilité dans les automates sur les ordres
- satisfaisabilité de LTL sur des ordres arbitraires
- intersection d'une formule avec un langage rationnel

Jeux de longueur ordinale (SOFSEM'08, FSTTCS'08)

- modèle de jeux de longueur ordinale déterminés
- réduction d'un jeu ordinal en jeu de Muller
- étude des jeux ordinaux à priorité
- réduction du cas général au cas à priorité
- calcul de stratégies à mémoire finie