INRIA

Test d'automates temporisés

Amélie Stainer
M2RI, IFSIC, Rennes

encadrée par Nathalie Bertrand et Thierry Jéron
Vertecs

22 juin 2010

Introduction

- Les automates temporisés (TA) [AD94]
  - modèle pour les systèmes temps-réel
  - automates finis munis d'horloges continues
- Le test de conformité de systèmes temps-réel
  - instance :
    - un système temps-réel dont on ne connaît que l'interface (supposé modélisable par un TA)
    - une spécification donnée (sous forme d'un TA)
  - problème :
    - le système est-il conforme à la spécification ?

[AD94] Alur and Dill. A theory of Timed Automata. 1994

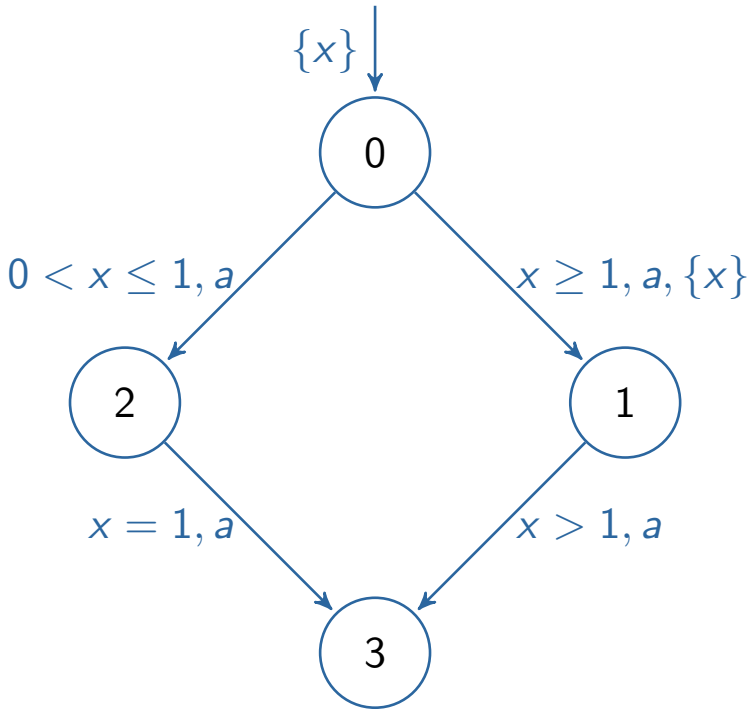
Introduction

- Observation : la génération de tests nécessite une déterminisation de la spécification.
  - *on-line* : temps de calcul
  - *off-line* : généralité des cas de test générés
- Objectif : utiliser et compléter la théorie de la déterminisation des TA pour générer des tests de conformité.

INRIA

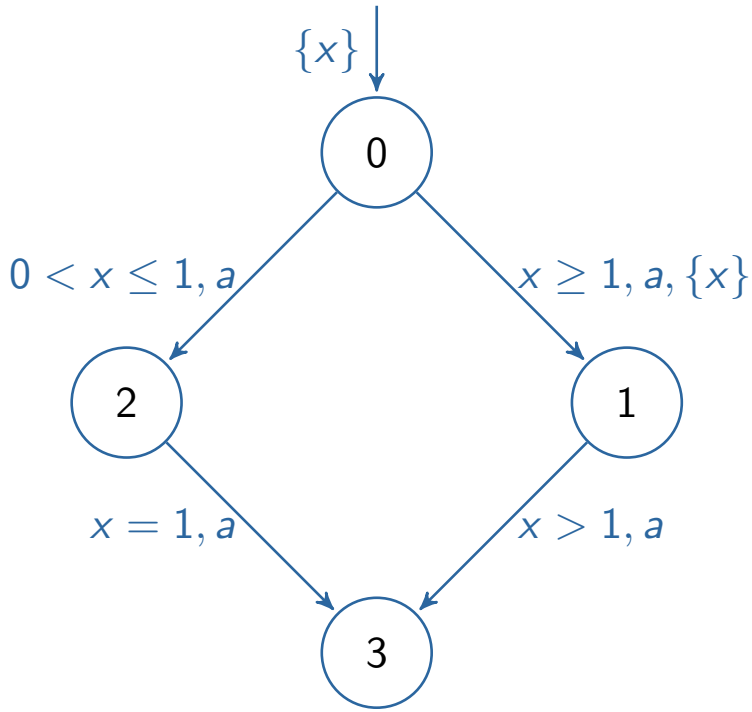
Automates temporisés

Exemple (Un automate temporisé $\mathcal{A}$)



Traces

Exemple (Des traces de $\mathcal{A}$)



Traces de $\mathcal{A}$:

- 1.a.0.a.12
- 2.a.1.a.2

Rq : $\mathcal{A}$ est non-déterministe.

Déterminisabilité

- Les TA ne sont pas tous déterminisables.
- La déterminisabilité d'un TA est indécidable.

$\Rightarrow$ 2 approches :

- déterminisation exacte mais sans assurer la terminaison [BBBB09]
- terminaison mais sur-approximation déterministe [KT09]

[BBBB09] Baier, Bertrand, Bouyer and Brihaye. When are Timed Automata Determinizable? 2009
[KT09] Krichen and Tripakis. Conformance testing for real-time systems. 2009

Une procédure de déterminisation [BBBB09]

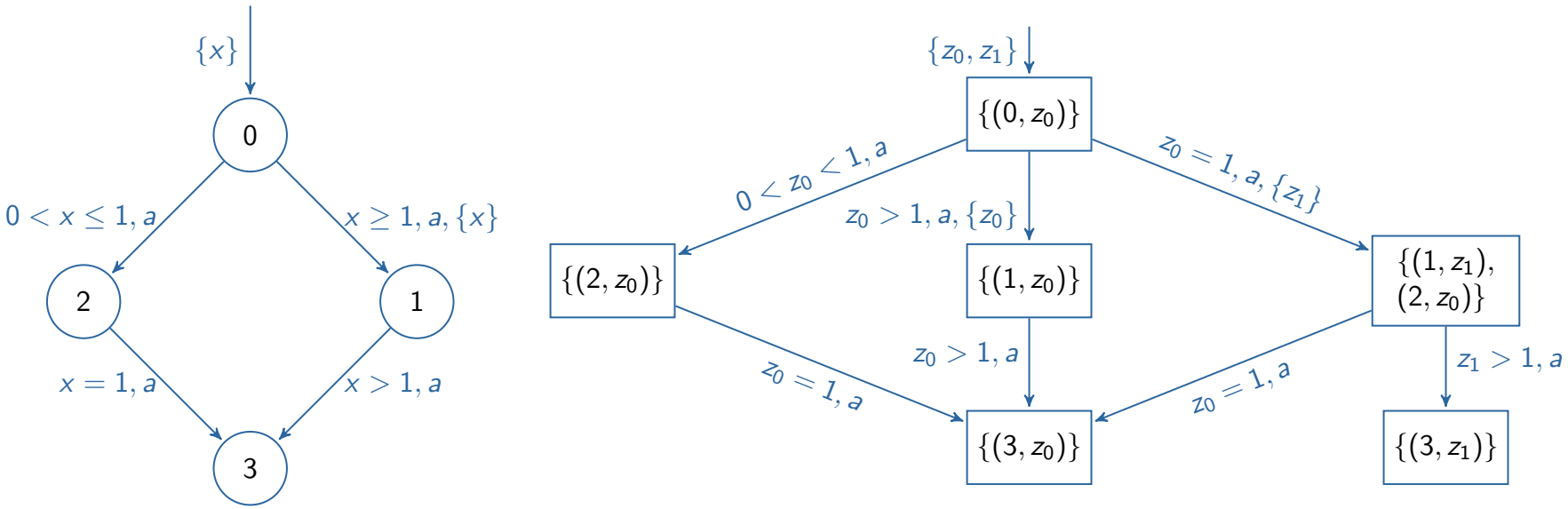
1. Dépliage en utilisant une nouvelle horloge à chaque étape
2. Déterminisation
3. Repliage en supprimant les horloges redondantes et en fusionnant les noeuds isomorphes

Terminaison sous certaines hypothèses.

Dans chaque localité, les horloges de l'automate initial sont représentées par les horloges de la construction.

Un exemple

Exemple (Déterminisation de $\mathcal{A}$ par [BBBB09])



Rq : cette procédure produit un TA à 2 horloges.

Une sur-approximation déterministe [KT09]

Construction pas à pas d'un DTA qui reconnait un langage contenant celui du TA initial.

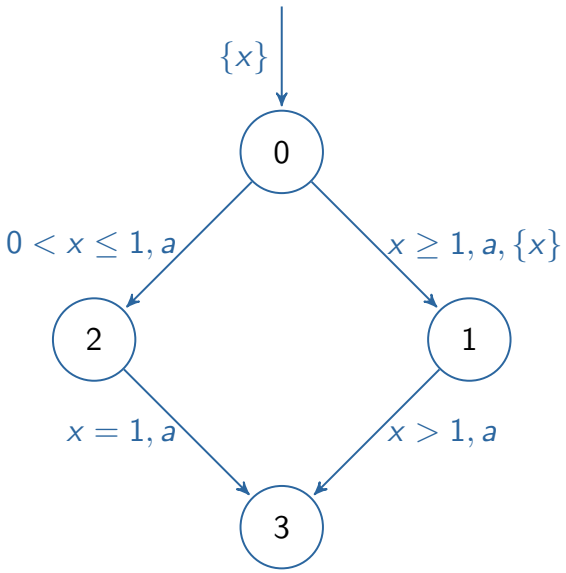
Politique de réinitialisation fixée :

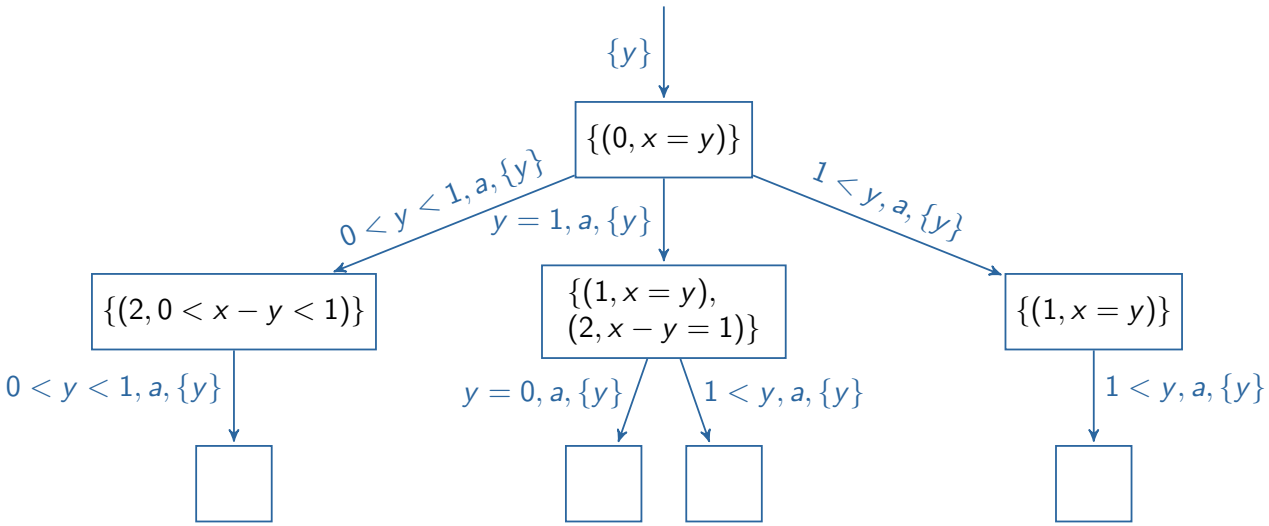
- Sur-approximation des gardes des transitions tirables grâce à l'horloge d'observation
- Estimation sur-approximée de l'ensemble des états atteignables

Dans chaque localité, on code des relations entre les horloges du TA initial et l'horloge d'observation.

Un exemple

Exemple (sur-approximation à une horloge correspondant à $\mathcal{A}$)





Rq : la sur-approximation est stricte (ex : $0, 8.a.0, 9.a.12$).

Notre approche

- But : combiner les deux méthodes.
  - ressources fixées (nombre d'horloges et constante maximale),
  - déterminisation si possible,
  - sur-approximation déterministe sinon.
- Méthode :
  - mêmes calculs que Krichen et Tripakis
  - un jeu pour choisir quand réinitialiser les horloges.
  - inspirée de l'approche de [BCD05] pour le diagnostic des TA

Dans chaque état du jeu, on code les relations entre les horloges d'observation et celles du TA initial.

[BCD05] Bouyer, Chevalier, D'Souza. Fault diagnosis using timed automata. 2005

Définition du jeu

- Un jeu fini *turn-based* de sûreté
- Deux joueurs : Spoiler et Determinisator
  - Spoiler choisit une action et quand la tirer
  - Determinisator choisit quelles horloges réinitialiser
    $\Rightarrow$ un choix de Spoiler + un choix de Determinisator = une transition
- Determinisator veut éviter les états construits en faisant une sur-approximation.
  - Un état contient une estimation des configurations possibles.
  - Chaque configuration peut être marquée, les états grisés sont ceux dont toutes les configurations sont marquées.

Propriétés du jeu

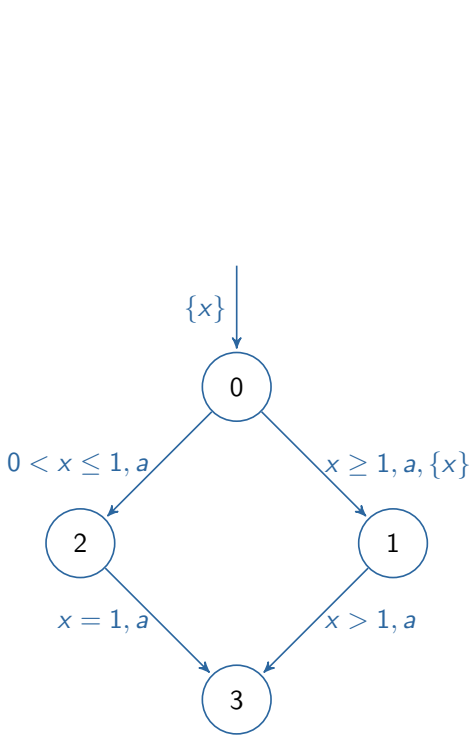
- À une stratégie S de Determinisator correspond un automate temporisé déterministe $Aut(S)$ (en fusionnant chaque choix de Spoiler avec le choix de Determinisator suivant).

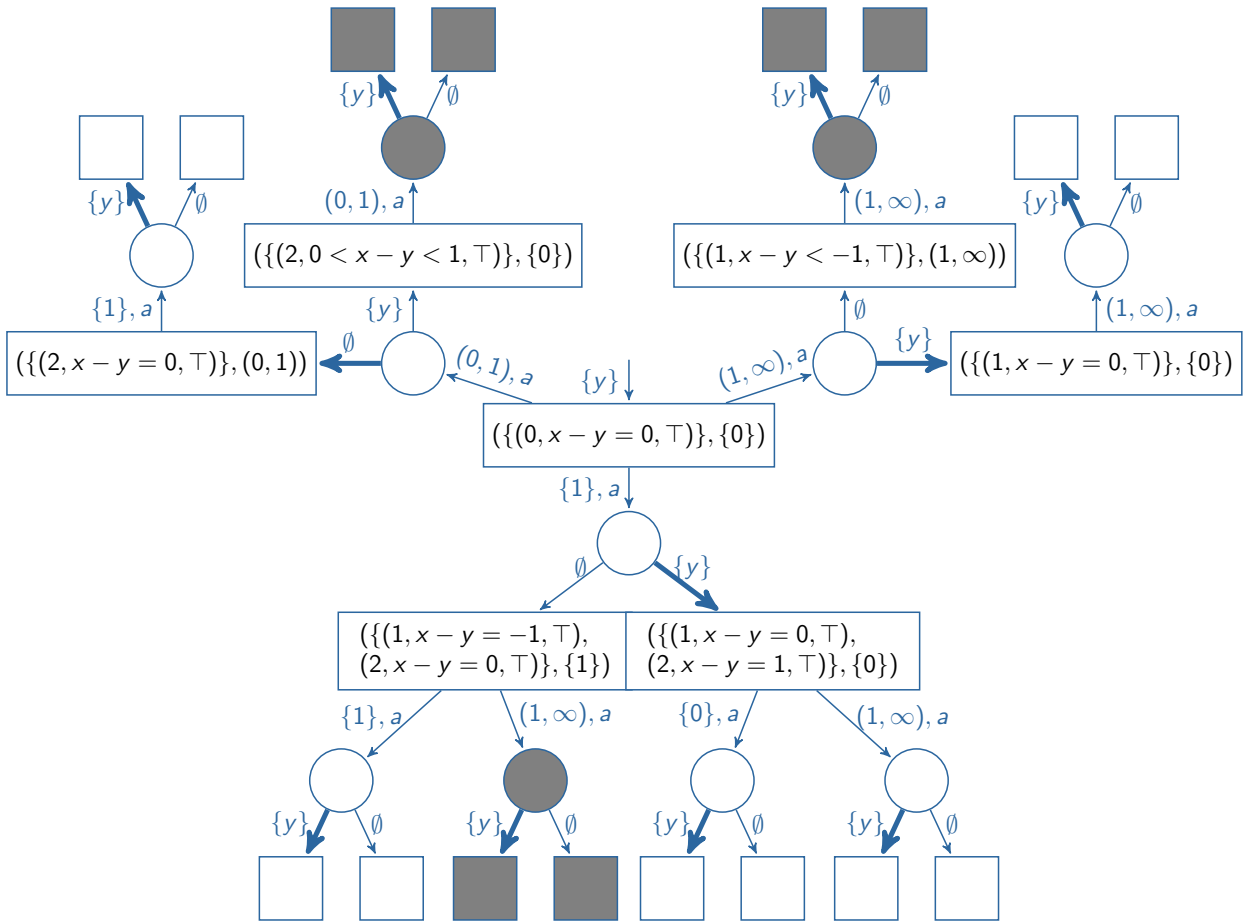
Propriétés

- À une stratégie de Determinisator correspond une sur-approximation.
- À une stratégie **gagnante** de Determinisator correspond un déterminisé **exact**.

Un exemple

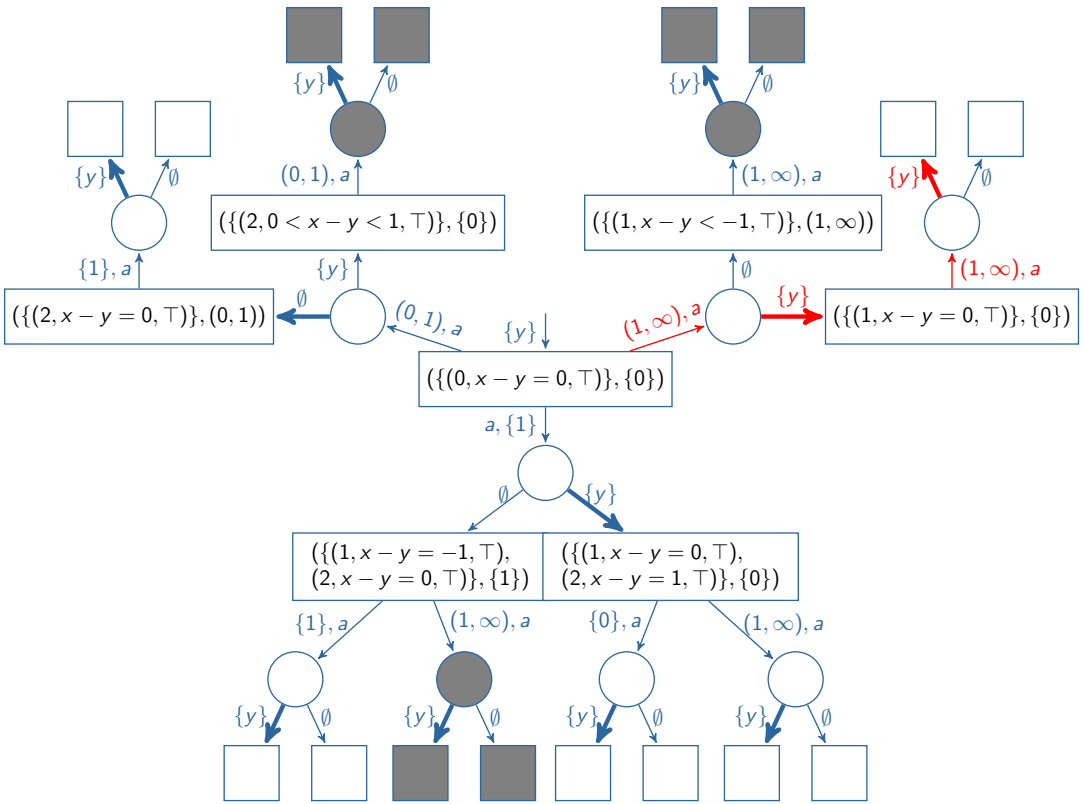
Exemple (Jeu correspondant à $\mathcal{A}$ et une stratégie gagnante pour Determinisator)

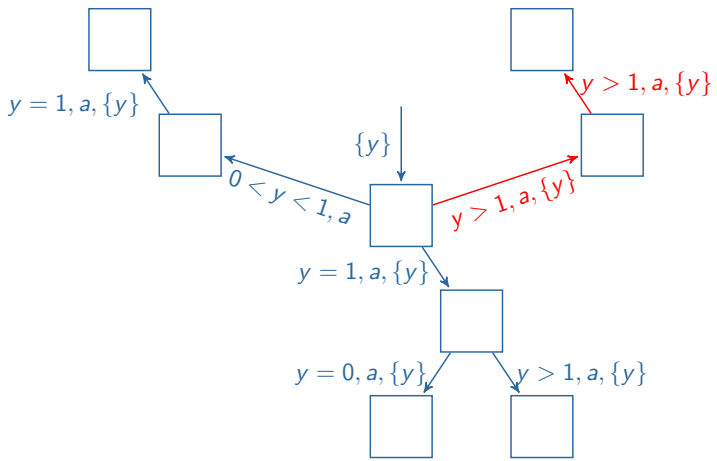




Un exemple

Exemple (DTA correspondant à une stratégie gagnante)





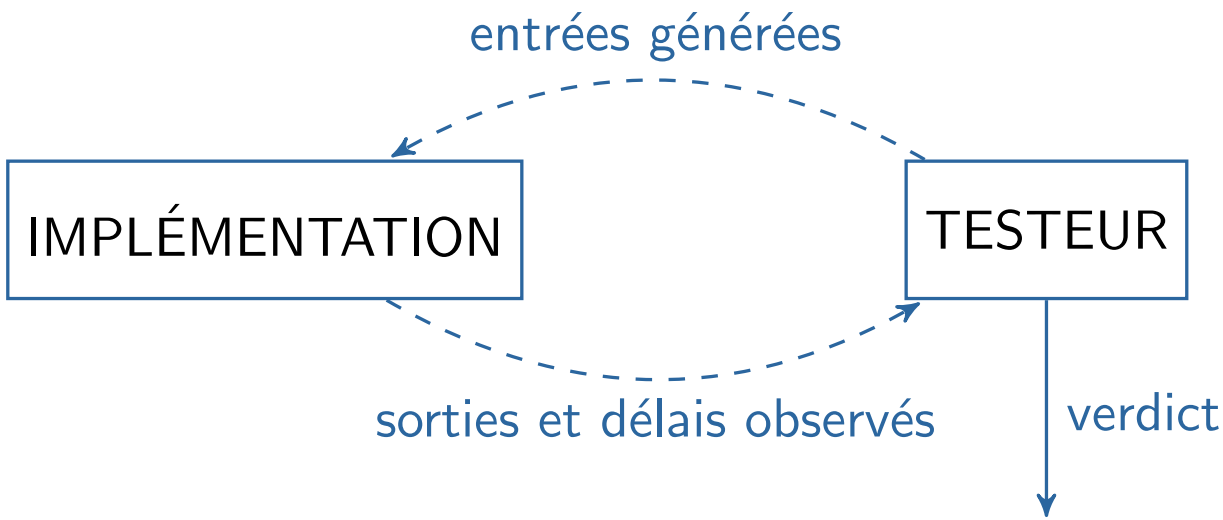
Comparaison

- Plus précise que la sur-approximation [KT09]
  $\Rightarrow$ toute construction de [KT09] correspond à une stratégie de Determinisator.
  $\Rightarrow$ on préserve le déterminisme.
- Strictement plus général que la procédure de déterminisation [BBBB09] :
  - nos relations entre horloges sont plus expressives que le mapping
    $\Rightarrow$ extension des classes de TA que l'on sait traiter
  - marquage des états par configurations
    $\Rightarrow$ on traite des inclusions de traces entre branches d'un TA.

Test : introduction

Le test de conformité :

- instance :
  - un système temps-réel dont on ne connaît que l'interface (supposé modélisable par un TA)
  - une spécification donnée (sous forme d'un TA non-déterministe)
- problème :
  - le système est-il conforme à la spécification ?

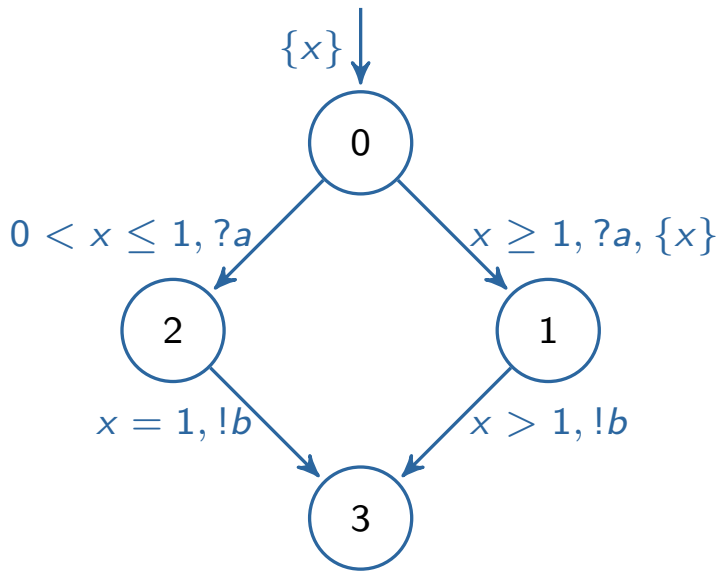


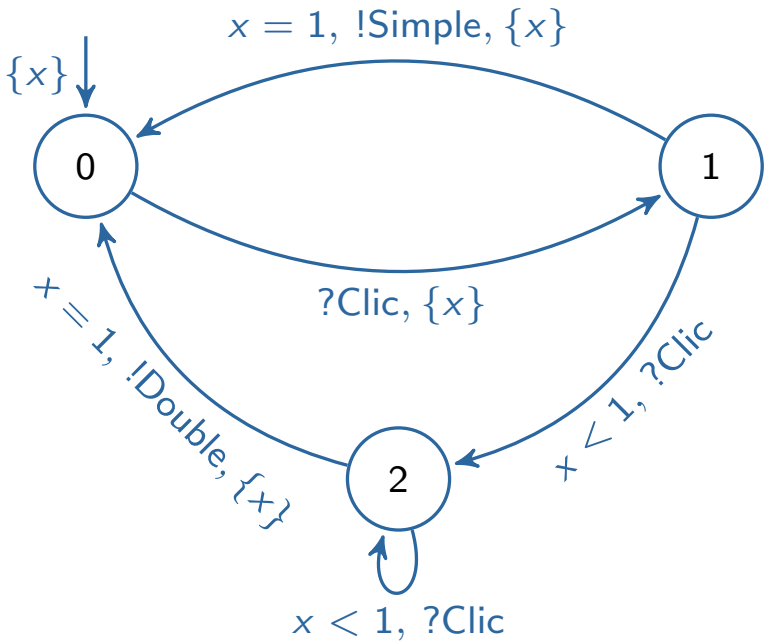
Les TA entrée/sortie

Alphabet partitionné :

- entrées marquées par ?
- sorties marquées par !

Exemple (Automates temporisés entrée/sortie (TAIO))





Relation de conformité

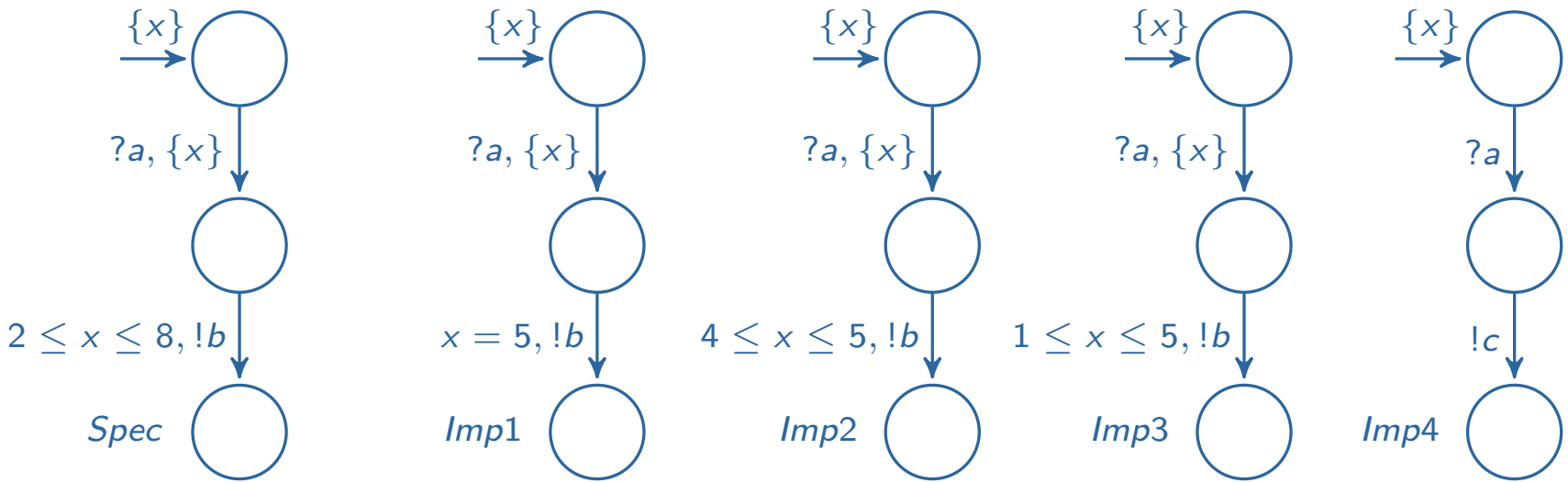
Formalisation de la conformité [KT09]

Définition (**tioco**, une relation de conformité)

Imp **tioco** Spec $\Leftrightarrow$ les sorties et délais observables de Imp après les traces de Spec sont inclus dans ceux de Spec.

Exemple (**tioco**)

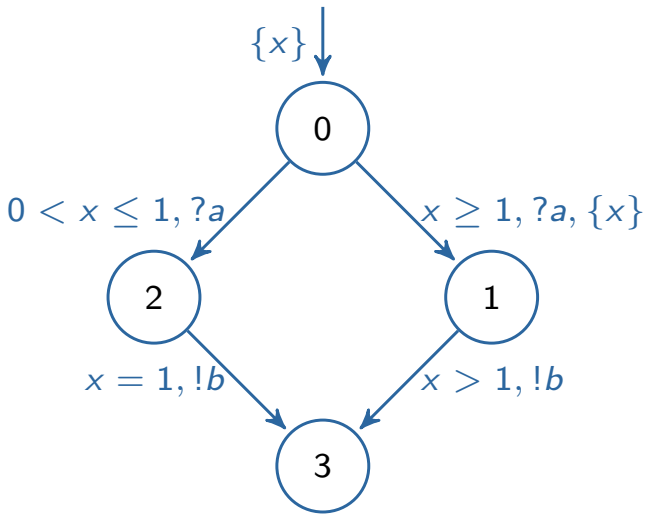
Imp1 **tioco** Spec Imp2 **tioco** Spec $\neg$(Imp3 **tioco** Spec) $\neg$(Imp4 **tioco** Spec)

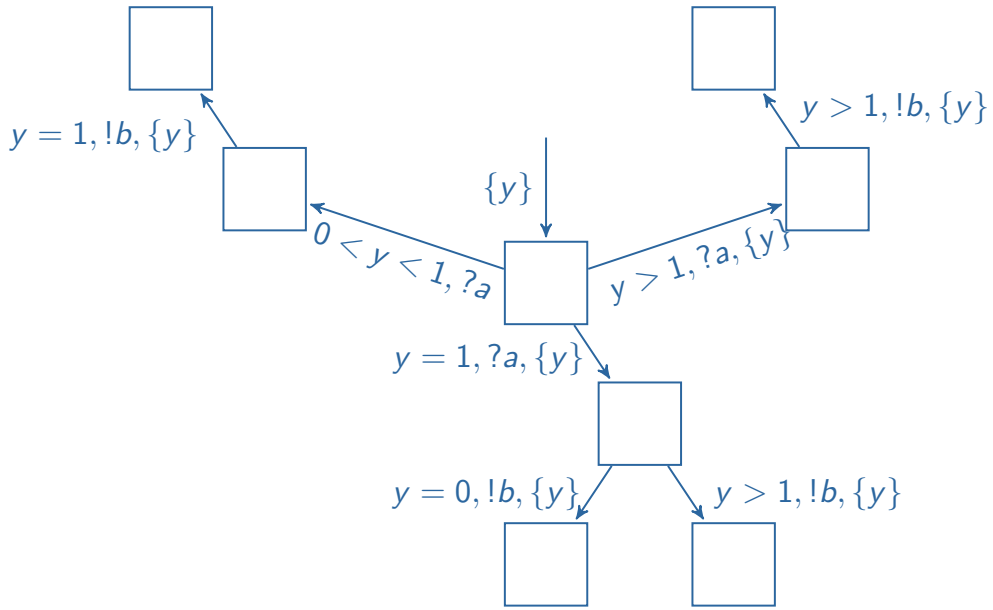


Génération de tests

Exemple (Construction du testeur canonique)

Déterminisation :

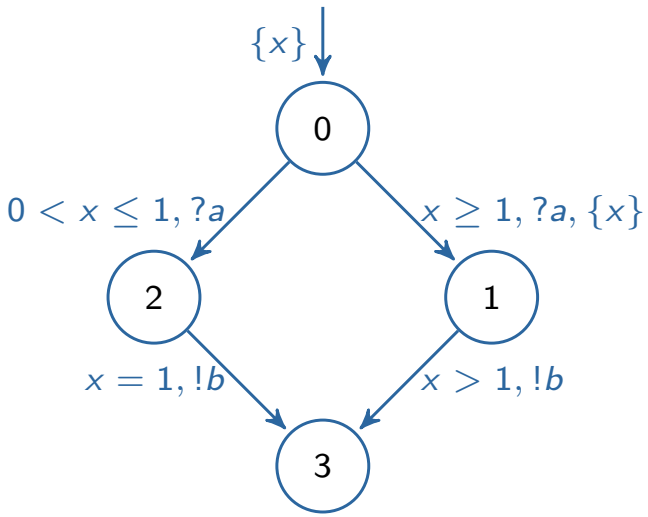


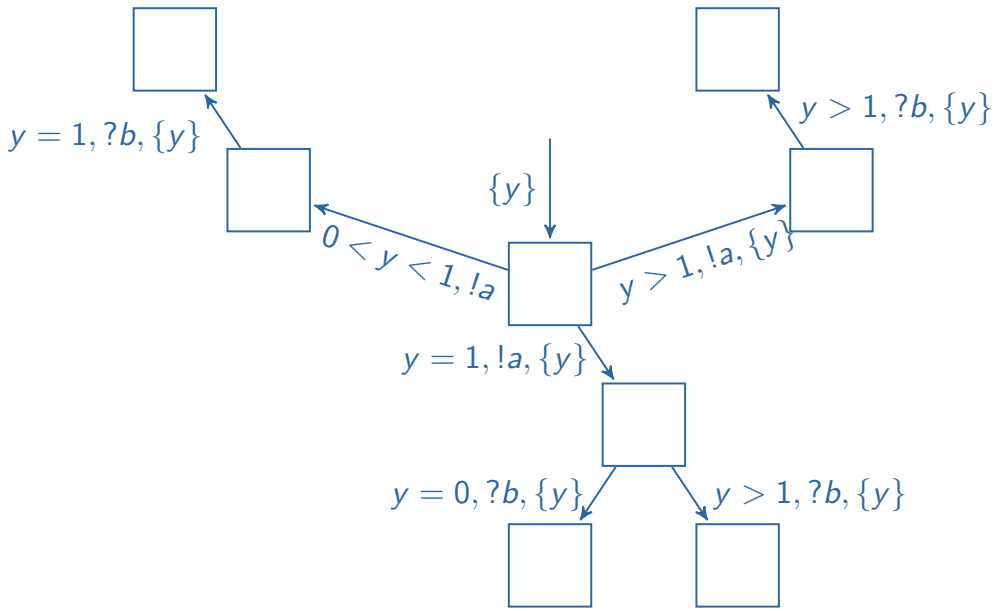


Génération de tests

Exemple (Construction du testeur canonique)

Inversion entrées-sorties :

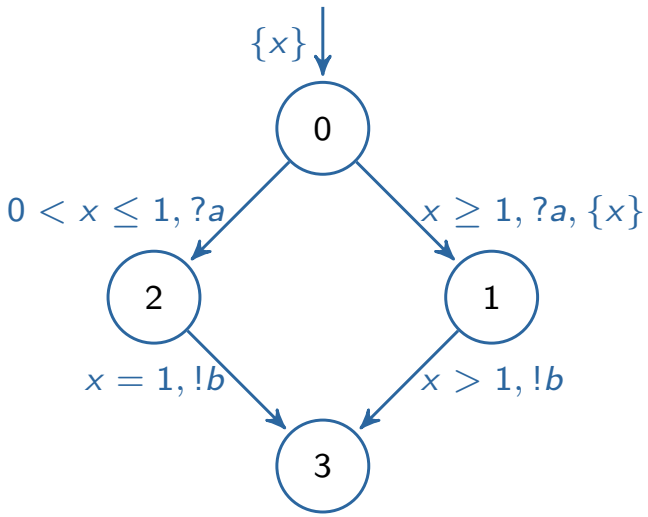


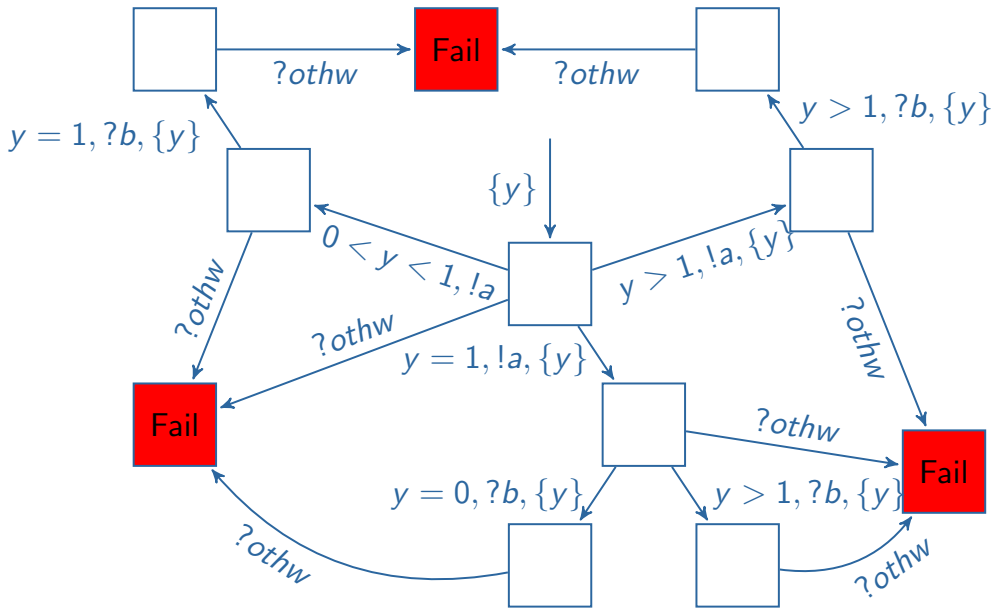


Génération de tests

Exemple (Construction du testeur canonique)

Ajout des verdicts :





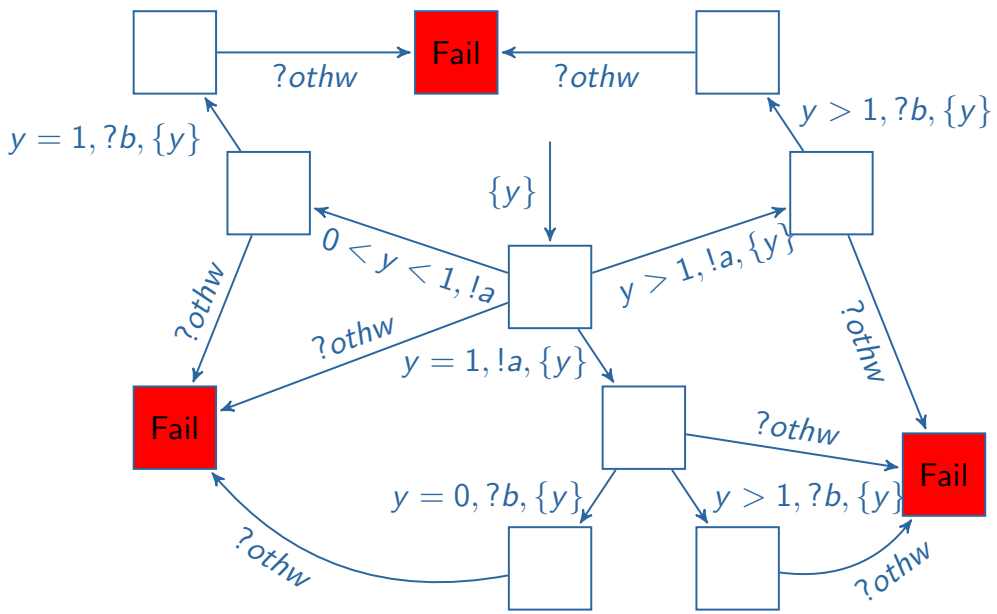
Rq : besoin du déterminisme pour ajouter les verdicts.

Génération de tests

Exemple (Construction du testeur canonique)

Génération de test :

- Parcours du testeur canonique
- Verdict : *échec* si on finit en Fail, *réussi* sinon.



Adaptation de notre approche

- [KT09] : spécification supposée complète en entrée
  $\Rightarrow$ **tioco** = inclusion de traces,
  $\Rightarrow$ la sur-approximation préserve **tioco**.
- Relaxation de cette hypothèse tout en préservant **tioco**
  $\Leftarrow$ on sous-approxime sur les entrées.

Propriété de notre adaptation

Étant donnés un TA $\mathcal{A}$ et une stratégie S de Determinisator pour notre jeu :

pour toute implémentation Imp, (Imp **tioco** $\mathcal{A} \Rightarrow$ Imp **tioco** $Aut(S)$).

Conclusion

- But du stage :
  - Déterminisation des automates temporisés
  - Application au test
- Résultats :
  - Déterminisation des automates temporisés
    - Amélioration de deux approches de déterminisation existantes
    - Présentation de mon travail à MOVEP 2010 (école d'été)
    - Soumission d'un article (FSTTCS'10)
  - Application au test
    - Formalisation de la génération de test avec objectifs
    - Adaptation de notre méthode de déterminisation pour préserver **tioco**
    - Poursuite de ce travail cet été avec la collaboration de Moez Krichen
    - Adaptation de notre approche pour la déterminisation à des modèles plus riches