

Concours CR CNRS 2023

Ismaël Jecker

Projet de recherche :
Synthèse automatique de programmes

EPFL**Bachelor - Master
en Mathématiques****ULB** UNIVERSITÉ
LIBRE
DE BRUXELLES**Doctorat en
Informatique****IST AUSTRIA****Postdoctorat****Postdoctorat****2015****2019****2021**

Collaborations :

 Emmanuel Filiot
 Jean-François Raskin
 Christof Löding
 Nathan Lhote
 Pierre-Alain Reynier
 Marie van den Bogaard
 ...

 Krishnendu Chatterjee
 Henning Fernau
 Orna Kupferman
 Karoliina Lehtinen
 Shibashis Guha
 Martin Zimmermann
 ...

 Wojciech Czerwiński
 Filip Mazowiecki
 Sławomir Lasota
 David Purser
 Anca Muscholl
 Gabriele Puppis
 ...

Publications :

LICS'17 ICALP'17
 ICALP'16 FoSSaCS'17
 MFCS'18 CSL'18
 IJFCS(×2) DLT'15'16

SODA'21 STACS'21
 MFCS'21 best paper award
 CONCUR'21 MFCS'20 (×2)
 FSTTCS'21

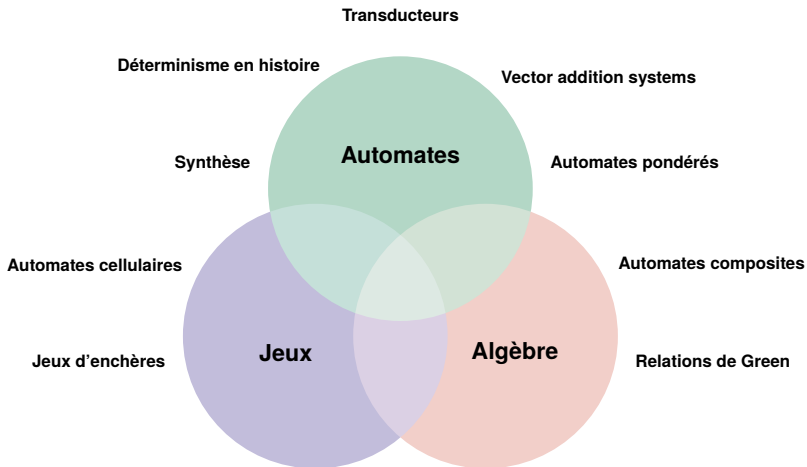
AAAI'23 STACS'23
 Acta Informatica
 FSTTCS'22 (×2)

Talks invités :

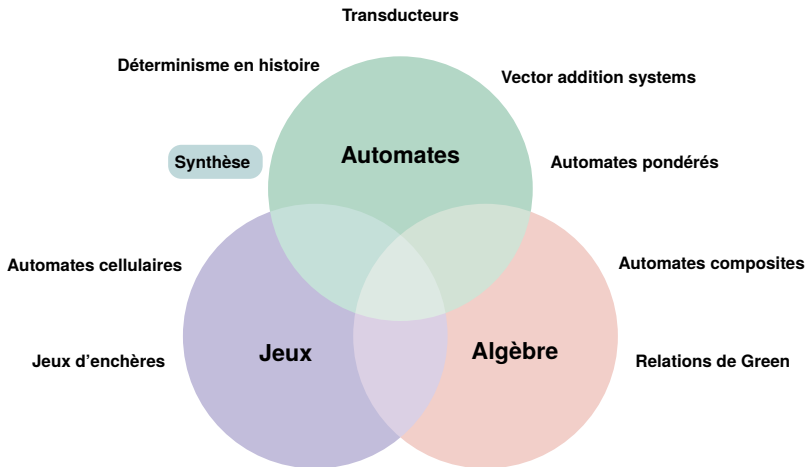
TRENDS 2019

DLT 2023

Thématiques de recherche



Thématiques de recherche

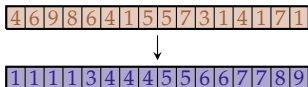


Synthèse automatique de programmes

Problème : Comment concevoir des programmes informatiques **corrects** ?

- Protocoles de sécurité
- Équipements médicaux
- Véhicules autonomes

Solution : En construisant **automatiquement** des systèmes à partir de leurs spécifications



```
public static void sort(int[] a)
{
    if(a.length <= 1) {return; }
    int[] first = new int[a.length / 2];
    int[] second = new int[a.length - first.length];

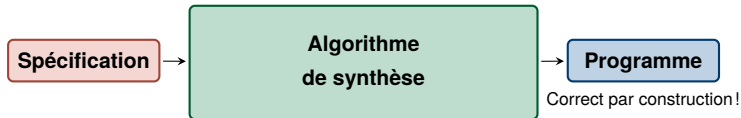
    for(int i = 0; i < first.length; i++)
    {
        first[i] = a[i];
    }
    for(int i = 0; i < second.length; i++)
    {
        second[i] = a[i];
    }
    sort(first);
    sort(second);
    merge(first, second, a);
}
```

Synthèse automatique de programmes

Problème : Comment concevoir des programmes informatiques **corrects** ?

- Protocoles de sécurité
- Équipements médicaux
- Véhicules autonomes

Solution : En construisant **automatiquement** des systèmes à partir de leurs spécifications

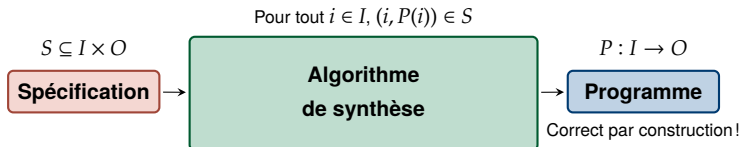


Synthèse automatique de programmes

Problème : Comment concevoir des programmes informatiques **corrects** ?

- Protocoles de sécurité
- Équipements médicaux
- Véhicules autonomes

Solution : En construisant **automatiquement** des systèmes à partir de leurs spécifications

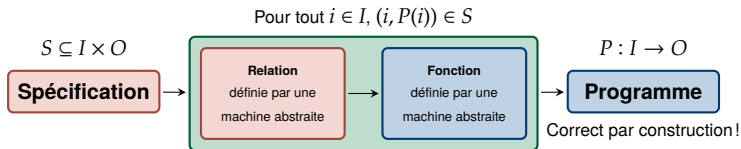


Synthèse automatique de programmes

Problème : Comment concevoir des programmes informatiques **corrects** ?

- Protocoles de sécurité
- Équipements médicaux
- Véhicules autonomes

Solution : En construisant **automatiquement** des systèmes à partir de leurs spécifications



Motive l'étude des machines reconnaissant des transformations mot-vers-mot

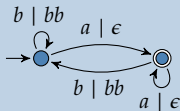
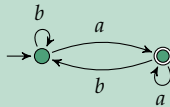
Résultat significatif :

[STACS 2023] A Regular and Complete Notion of Delay for Streaming String Transducers

Emmanuel Filiot, **Ismaël Jecker**, Christof Löding, Sarah Winter

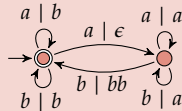
Reconnaît un **langage**

Automate fini



Transducteur déterministe

Reconnaît une **fonction**



Transducteur non-déterministe

Reconnaît une **relation**

Reconnaît un **langage**

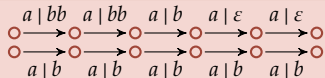
Automate fini

Équivalence : **DÉCIDABLE**

Équivalence : **DÉCIDABLE**

Transducteur déterministe

Reconnaît une **fonction**



Équivalence : **INDÉCIDABLE**

Transducteur non-déterministe

Reconnaît une **relation**

Reconnaît un **langage**

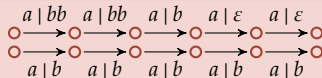
Automate fini

Équivalence : **DÉCIDABLE**

Équivalence : **DÉCIDABLE**

Transducteur déterministe

Reconnaît une **fonction**



Équivalence : **INDÉCIDABLE**

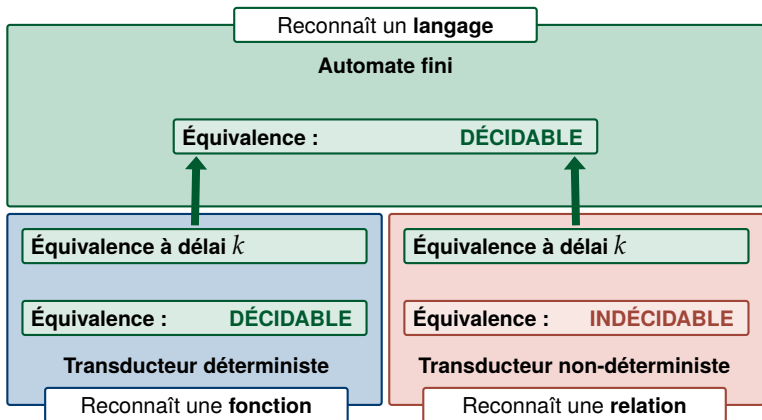
Transducteur non-déterministe

Reconnaît une **relation**

Le **délai** mesure à quel point deux exécutions équivalentes sont différentes

Régulier : transforme tout problème en un problème sur les automates

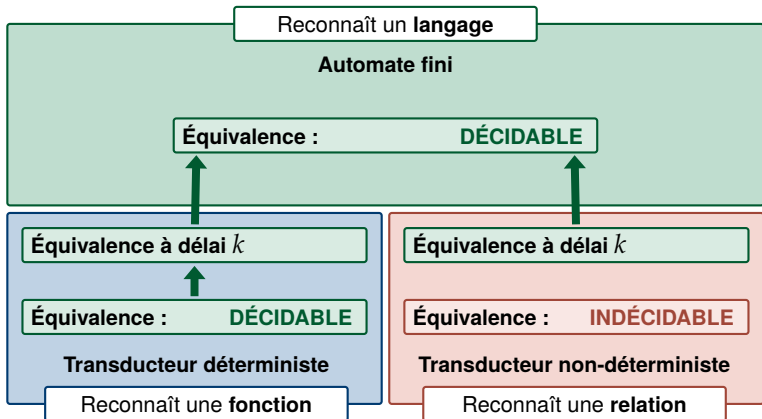
Complet : capable d'identifier les transducteurs déterministes équivalents



Le **délai** mesure à quel point deux exécutions équivalentes sont différentes

Régulier : transforme tout problème en un problème sur les automates

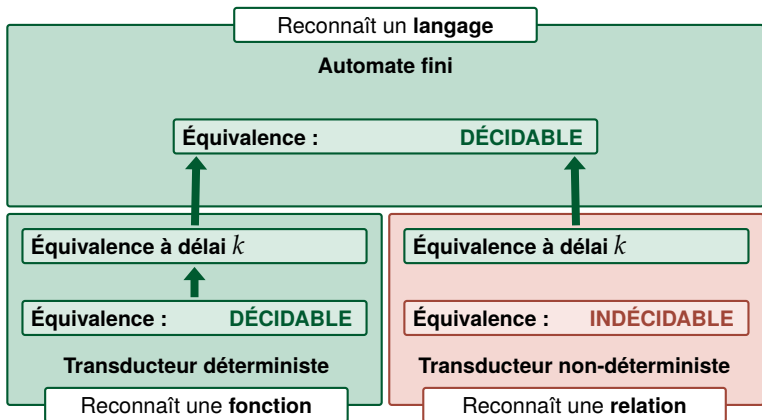
Complet : capable d'identifier les transducteurs déterministes équivalents



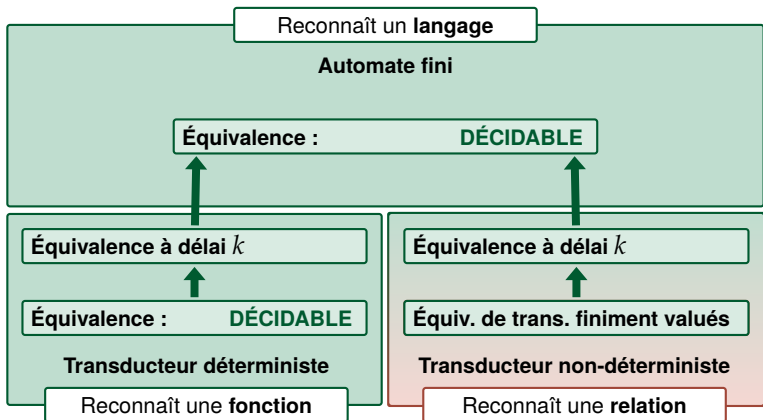
Le **délai** mesure à quel point deux exécutions équivalentes sont différentes

Régulier : transforme tout problème en un problème sur les automates

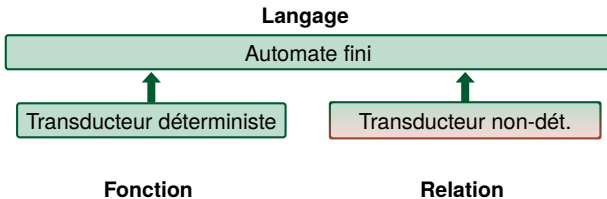
Complet : capable d'identifier les transducteurs déterministes équivalents

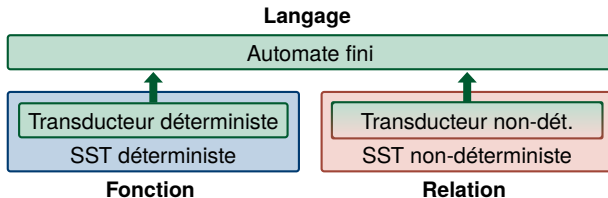


Le **délai** mesure à quel point deux exécutions équivalentes sont différentes
Les résultats majeurs sur les transducteurs reposent sur la notion de délai :
Équivalence, déterminisabilité, minimisation, ...



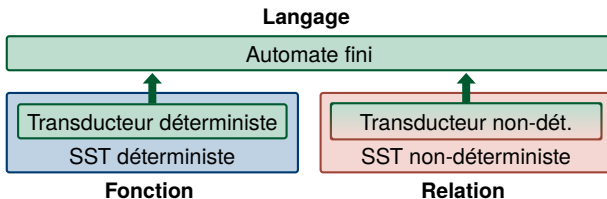
Le **délai** mesure à quel point deux exécutions équivalentes sont différentes
Les résultats majeurs sur les transducteurs reposent sur la notion de délai :
Équivalence, déterminisabilité, minimisation, ...





Les formalismes suivants reconnaissent la même classe de fonctions :

- Transducteurs bidirectionnels déterministes [Shepherdson, 1959]
- Transductions monadiques du second ordre [Courcelle, 1991]
- Streaming string transducers déterministes [Alur, Černý, 2010]
- Regular Combinators [Alur, Freilich, Raghothaman, 2014]
- Transducteurs bidirectionnels réversibles [Dartois, Fournier, Jecker, Lhote, 2017]
- Regular list functions [Bojańczyk, Daviaud, Krishna, 2018]



Existe-t-il une notion de délai **régulier et complet** pour les SST ?

[FSTTCS 2018] Origin-Equivalence of Two-Way Word Transducers Is in PSpace

Sougata Bose, Anca Muscholl, Vincent Penelle, Gabriele Puppis

[MFCS 2019] On Synthesis of Resynchronizers for Transducers

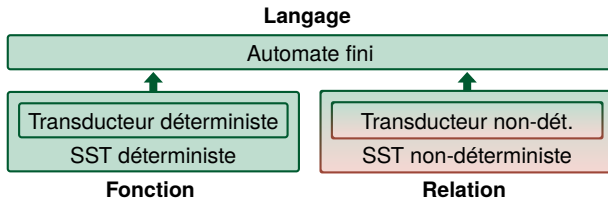
Sougata Bose, Shankara Narayanan Krishna, Anca Muscholl, Vincent Penelle, Gabriele Puppis

[FoSSaCS 2021] One-way Resynchronizability of Word Transducers

Sougata Bose, Shankara Narayanan Krishna, Anca Muscholl, Gabriele Puppis

[STACS 2023] A Regular and Complete Notion of Delay for SST

Emmanuel Filiot, **Ismaël Jecker**, Christof Löding, Sarah Winter



Existe-t-il une notion de délai **régulier et complet** pour les SST ?

[FSTTCS 2018] Origin-Equivalence of Two-Way Word Transducers Is in PSpace

Sougata Bose, Anca Muscholl, Vincent Penelle, Gabriele Puppis

[MFCS 2019] On Synthesis of Resynchronizers for Transducers

Sougata Bose, Shankara Narayanan Krishna, Anca Muscholl, Vincent Penelle, Gabriele Puppis

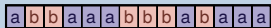
[FoSSaCS 2021] One-way Resynchronizability of Word Transducers

Sougata Bose, Shankara Narayanan Krishna, Anca Muscholl, Gabriele Puppis

[STACS 2023] A Regular and Complete Notion of Delay for SST

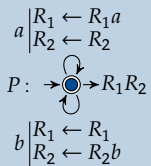
Emmanuel Filiot, **Ismaël Jecker**, Christof Löding, Sarah Winter

Tri stable

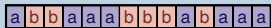


$R_1 :$

$R_2 :$

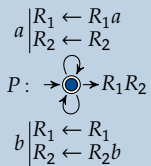


Tri stable

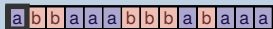


$R_1 :$

$R_2 :$

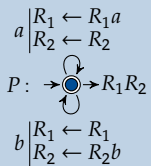


Tri stable

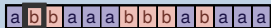


$R_1 :$

$R_2 :$

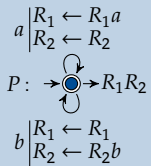


Tri stable

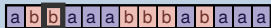


R_1 : a

R_2 :

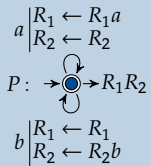


Tri stable

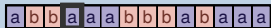


R_1 : a

R_2 : b

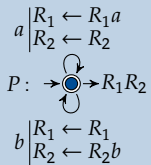


Tri stable

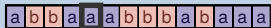


R_1 : a

R_2 : b b

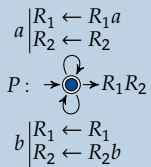


Tri stable

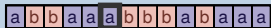


R_1 : a a

R_2 : b b

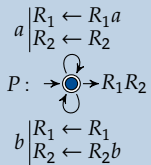


Tri stable

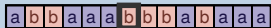



R_1 : a a a


R_2 : b b

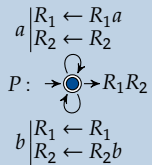


Tri stable

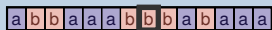


R_1 : 

R_2 : 

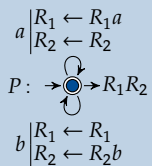


Tri stable

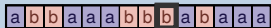



R_1 : a a a a


R_2 : b b b

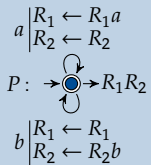


Tri stable

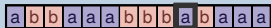


R_1 : 

R_2 : 

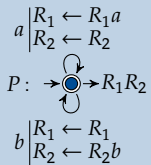


Tri stable

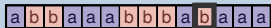


R_1 : a a a a

R_2 : b b b b b

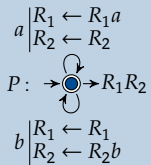


Tri stable

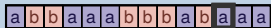


R_1 : a a a a a

R_2 : b b b b b

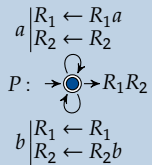


Tri stable

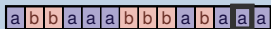


R_1 :

R_2 :

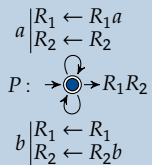


Tri stable

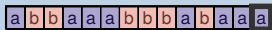


R_1 :

R_2 :

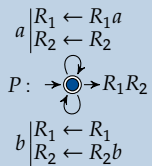


Tri stable



R_1 : a a a a a a a

R_2 : b b b b b b

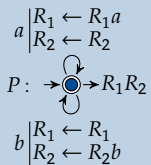


Tri stable

a b b a a a b b a b a a a

R_1 : a a a a a a a a

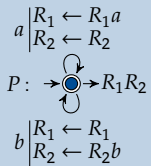
R_2 : b b b b b b



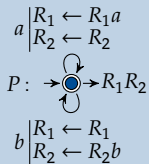
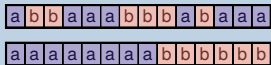
Tri stable

a b b a a a b b b a b a a a

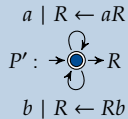
R_1R_2 : a a a a a a a a b b b b b b



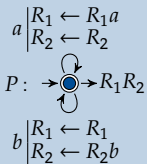
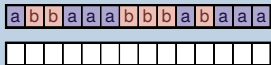
Tri stable



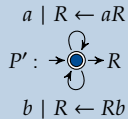
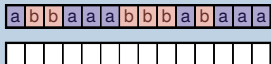
Tri instable



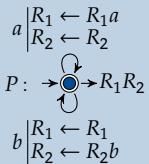
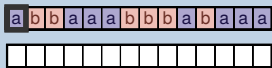
Tri stable



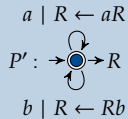
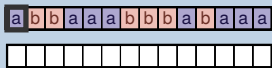
Tri instable



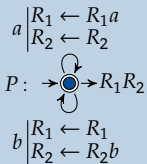
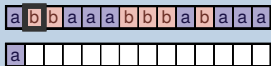
Tri stable



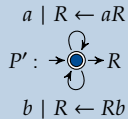
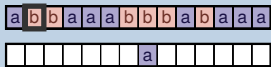
Tri instable



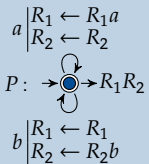
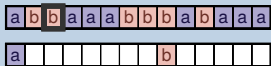
Tri stable



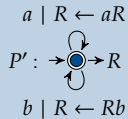
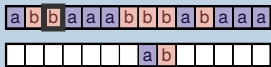
Tri instable



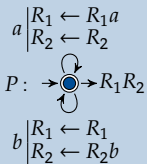
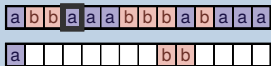
Tri stable



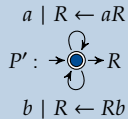
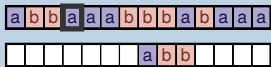
Tri unstable



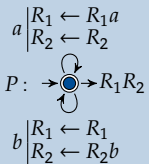
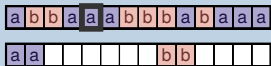
Tri stable



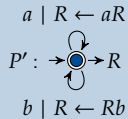
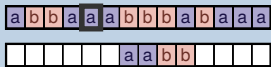
Tri unstable



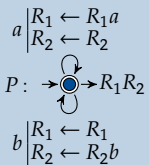
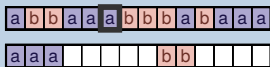
Tri stable



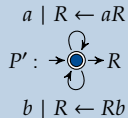
Tri instable



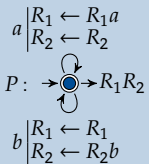
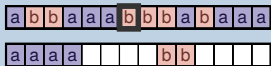
Tri stable



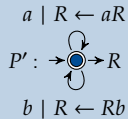
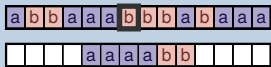
Tri unstable



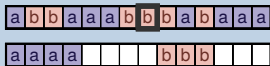
Tri stable



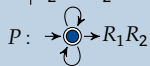
Tri instable



Tri stable

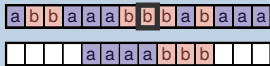


$$a \begin{cases} R_1 \leftarrow R_1 a \\ R_2 \leftarrow R_2 \end{cases}$$

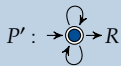


$$b \begin{cases} R_1 \leftarrow R_1 \\ R_2 \leftarrow R_2 b \end{cases}$$

Tri instable

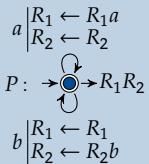
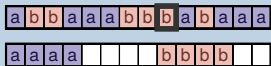


$$a \mid R \leftarrow aR$$

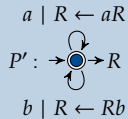


$$b \mid R \leftarrow Rb$$

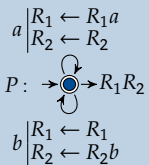
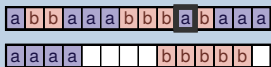
Tri stable



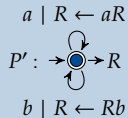
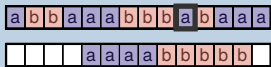
Tri instable



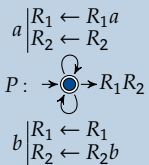
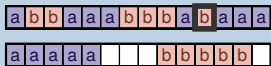
Tri stable



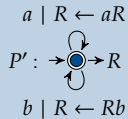
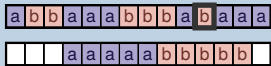
Tri unstable



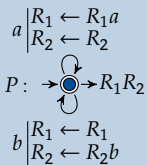
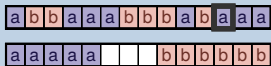
Tri stable



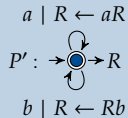
Tri unstable



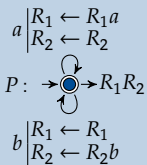
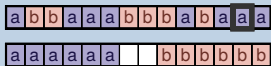
Tri stable



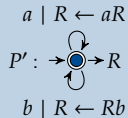
Tri unstable



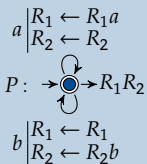
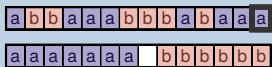
Tri stable



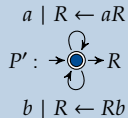
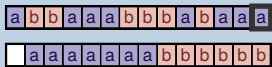
Tri unstable



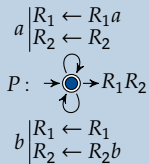
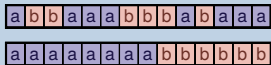
Tri stable



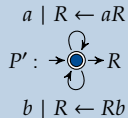
Tri unstable



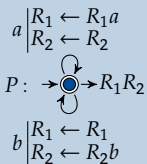
Tri stable



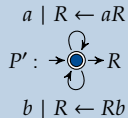
Tri unstable



Tri stable



Tri instable



Le bloc de a est rempli différemment, mais à la même **cadence**

Le délai mesure la **distance** entre deux exécutions produisant la même sortie

Le délai est **petit** si lorsque l'on décompose la sortie en blocs puissances de **petits** mots, chacun de ces blocs est rempli à une cadence **proche**

$u =$ 

$u =$ 

Le délai mesure la **distance** entre deux exécutions produisant la même sortie

Le délai est **petit** si lorsque l'on décompose la sortie en blocs puissances de **petits** mots, chacun de ces blocs est rempli à une cadence **proche**

$u =$ 

$u =$ 

Le délai mesure la **distance** entre deux exécutions produisant la même sortie

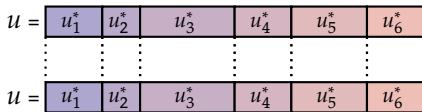
Le délai est **petit** si lorsque l'on décompose la sortie en blocs puissances de **petits** mots, chacun de ces blocs est rempli à une cadence **proche**

$$u = \begin{array}{|c|c|c|c|c|c|} \hline u_1^* & u_2^* & u_3^* & u_4^* & u_5^* & u_6^* \\ \hline \end{array}$$

$$u = \begin{array}{|c|c|c|c|c|c|} \hline u_1^* & u_2^* & u_3^* & u_4^* & u_5^* & u_6^* \\ \hline \end{array}$$

Le délai mesure la **distance** entre deux exécutions produisant la même sortie

Le délai est **petit** si lorsque l'on décompose la sortie en blocs puissances de **petits** mots, chacun de ces blocs est rempli à une cadence **proche**



Le délai mesure la **distance** entre deux exécutions produisant la même sortie

Le délai est **petit** si lorsque l'on décompose la sortie en blocs puissances de **petits** mots, chacun de ces blocs est rempli à une cadence **proche**

$$\begin{array}{l} u = \begin{array}{|c|c|c|c|c|c|} \hline u_1^* & u_2^* & u_3^* & u_4^* & u_5^* & u_6^* \\ \hline \end{array} \\ \vdots \\ u = \begin{array}{|c|c|c|c|c|c|} \hline u_1^* & u_2^* & u_3^* & u_4^* & u_5^* & u_6^* \\ \hline \end{array} \end{array}$$

Régulier : transforme tout problème en un problème sur les automates

Complet : capable d'identifier les SST déterministes équivalents

Exemple concret d'application : le **Problème de Décomposition**

- Énoncé par **Rajeev Alur** et **Jyotirmoy Deshmukh** en **2011**

Peut-on décomposer tout SST **finiment valué** en SST déterministes ?

- Résolu en **2017** pour les SST avec **un unique registre** :

[STACS 2017] On the Decomposition of Finite-Valued SST

Paul Gallot, Anca Muscholl, Gabriele Puppis, Sylvain Salvati

- Finalement résolu dans le cadre général **grâce au délai** :

Théorème de Décomposition [Filiot, Jecker, Löding, Muscholl, Puppis, Winter]

Tout SST **finiment valué** peut être décomposé en SST déterministes

Corollaire : l'équivalence des SST finiment valués est décidable

Exemple concret d'application : le **Problème de Décomposition**

- Énoncé par **Rajeev Alur** et **Jyotirmoy Deshmukh** en **2011**

Peut-on décomposer tout **SST finiment valué** en SST déterministes ?

une vaste classe de SST **non-déterministes**

- Résolu en **2017** pour les SST avec **un unique registre** :

[STACS 2017] On the Decomposition of Finite-Valued SST

Paul Gallot, Anca Muscholl, Gabriele Puppis, Sylvain Salvati

- Finalement résolu dans le cadre général **grâce au délai** :

Théorème de Décomposition [Filiot, Jecker, Löding, Muscholl, Puppis, Winter]

Tout SST **finiment valué** peut être décomposé en SST déterministes

Corollaire : l'équivalence des SST finiment valués est décidable

Projet de recherche :

Synthèse automatique de programmes

Axe 1 : Développer de meilleurs algorithmes de synthèse

Axe 2 : Étendre le cadre de la synthèse

1. Développer de meilleurs algorithmes de synthèse

1.1 : Identifier la source de la complexité théorique

Peut-on évaluer la complexité d'un système à l'aide des **relations de Green** ?

1.2 : Décomposer les instances compliquées

Comment décomposer efficacement les transducteurs **non-déterministes** ?

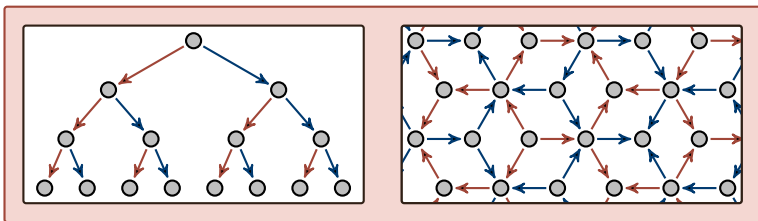
1.3 : Établir des algorithmes efficaces en pratique

Quel est l'impact de la **bisimulation à congruence près** sur la synthèse ?

1.1 Identifier la source de la complexité théorique

Spécification définie par un transducteur synchrone \Rightarrow ExpTime-complet

Toutefois, les systèmes très **grands** sont parfois très **simples**

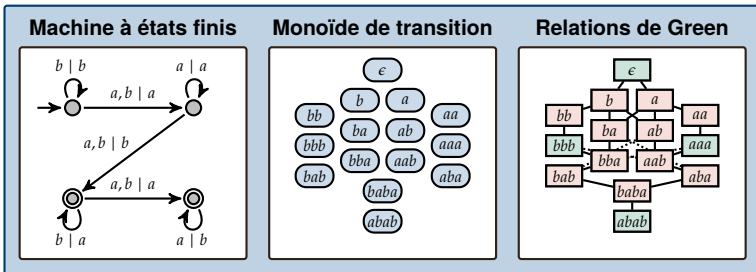


Quel **paramètre** permet de mesurer la **complexité intrinsèque** d'un système ?

1.1 Identifier la source de la complexité théorique

Spécification définie par un transducteur synchrone \Rightarrow ExpTime-complet

Proposition : Analyser les systèmes à l'aide des **monoïdes**



Peut-on évaluer la complexité d'un système grâce aux **relations de Green** ?

Premiers résultats : Étude de la \mathcal{J} -hauteur régulière

[STACS 2021]

2. Étendre le cadre de la synthèse

2.1 : Étudier la synthèse des streaming string transducers

Peut-on synthétiser des SST utilisant un nombre **minimal** de registres ?

2.2 : Approfondir l'étude du déterminisme en histoire

Quelles sont les limites du dét. en histoire pour les systèmes à états **infinis** ?

2.3 : Explorer les variantes quantitatives de la synthèse

Comment synthétiser le programme le plus **proche** de la spécification ?

2.3 Explorer les variantes quantitatives de la synthèse

Un algorithme de synthèse standard répond soit :

- **Oui**, la spécification est réalisable, voici un programme **P** qui la satisfait
- **Non**, la spécification n'est pas réalisable

L'utilisation de **spécifications quantitatives** permet d'être plus précis

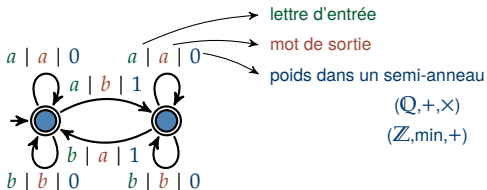
2.3 Explorer les variantes quantitatives de la synthèse

Un algorithme de synthèse standard répond soit :

- **Oui**, la spécification est réalisable, voici un programme **P** qui la satisfait
- **Non**, la spécification n'est pas réalisable

L'utilisation de **spécifications quantitatives** permet d'être plus précis

Modèle proposé : les transducteurs **pondérés**



Comment synthétiser le programme le plus **proche** de la spécification ?

Premiers résultats : Krishnendu Chatterjee et Ege Saraç (ISTA)

Intégration

Aix Marseille Université, LIS, équipe MOVE

Thématiques : Langages et transformations, vérification

Collaborations : K. Lehtinen, N. Lhote, B. Monmege, P.-A. Reynier, J.-M. Talbot



ENS Lyon, LIP, équipe PLUME

Thématiques : Programmes corrects par construction

Collaborations : A. Doumane, D. Kuperberg, D. Pous



Université Gustave Eiffel, LIGM, équipe BAAM

Thématiques : Théorie des modèles finis et infinis, Théorie des jeux

Collaborations : A. Carayol, M. van den Bogaard



EPFL**Bachelor - Master
en Mathématiques****ULB** UNIVERSITÉ
LIBRE
DE BRUXELLES**Doctorat en
Informatique****IST AUSTRIA****Postdoctorat****Postdoctorat****2015****2019****2021**

Collaborations :

 Emmanuel Filiot
 Jean-François Raskin
 Christof Löding
 Nathan Lhote
 Pierre-Alain Reynier
 Marie van den Bogaard
 ...

 Krishnendu Chatterjee
 Henning Fernau
 Orna Kupferman
 Karoliina Lehtinen
 Shibashis Guha
 Martin Zimmermann
 ...

 Wojciech Czerwiński
 Filip Mazowiecki
 Sławomir Lasota
 David Purser
 Anca Muscholl
 Gabriele Puppis
 ...

Publications :

LICS'17 ICALP'17
 ICALP'16 FoSSaCS'17
 MFCS'18 CSL'18
 IJFCS(×2) DLT'15'16

SODA'21 STACS'21
 MFCS'21 best paper award
 CONCUR'21 MFCS'20 (×2)
 FSTTCS'21

AAAI'23 STACS'23
 Acta Informatica
 FSTTCS'22 (×2)

Talks invités :

TRENDS 2019

DLT 2023