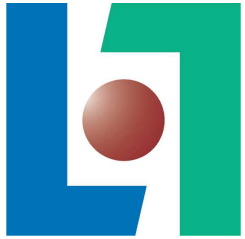


Un problème de langages formels

Géraud Sénizergues

LaBRI, Université de Bordeaux,

11 Décembre 2014

La face cachée (aux étudiants) de l'enseignant-chercheur

contents

Égalité de deux objets

Deux mots

u= abaaabaaabaaabaaabaaababbabab
v= abaaabaaabaaabaaabaaababbabab

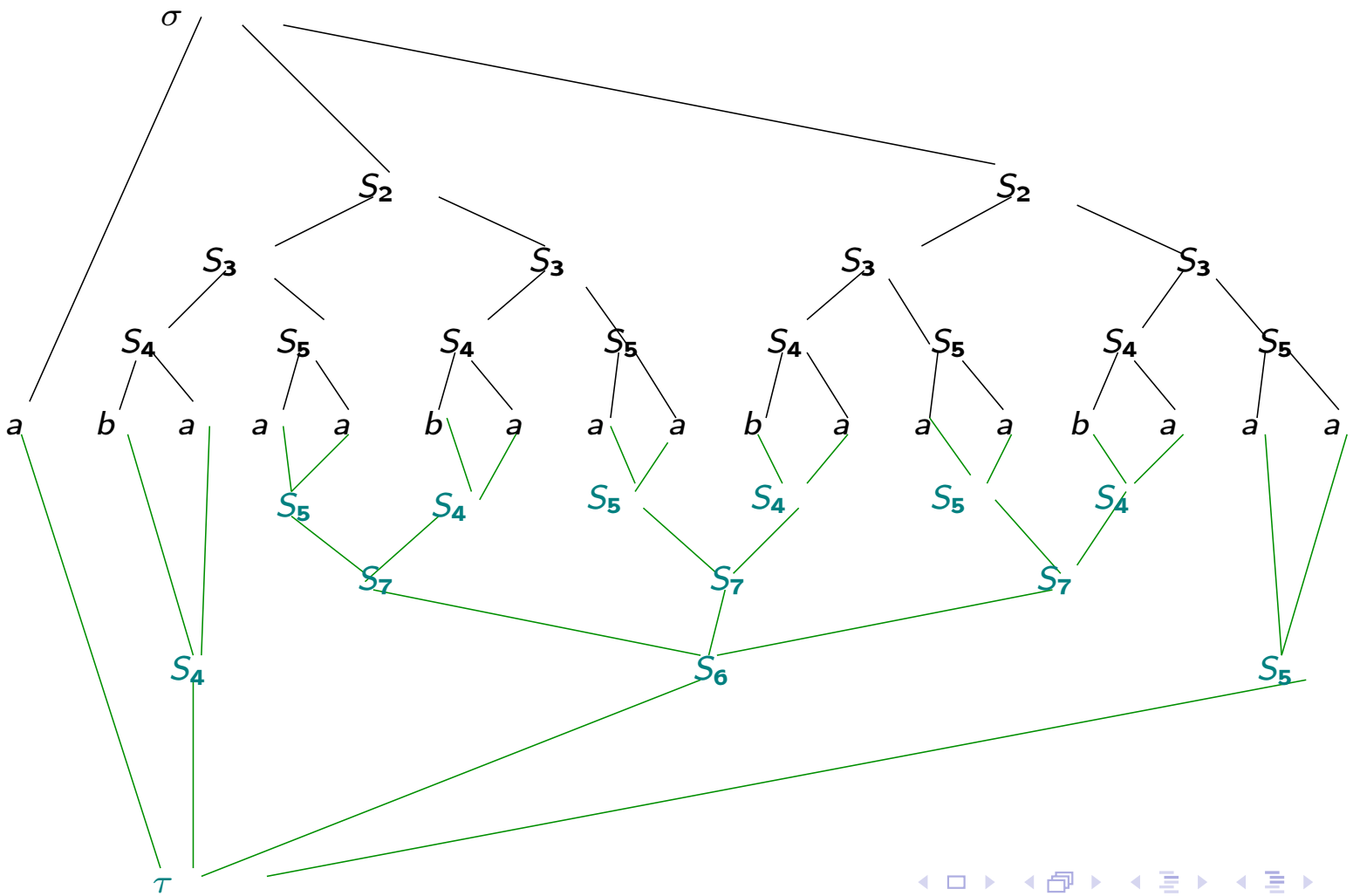
Deux mots

Deux mots sur l'alphabet $\{a, b\}$:

$$
\begin{aligned}
\sigma &\to aS_1 \\
\tau &\to aS_4S_6S_5 \\
S_1 &\to S_2S_2 \\
S_2 &\to S_3S_3 \\
S_3 &\to S_4S_5 \\
S_4 &\to ba \\
S_5 &\to aa \\
S_6 &\to S_7S_7S_7 \\
S_7 &\to S_5S_4.
\end{aligned}
$$

Deux mots

On développe :



Deux langages rationnels

Exemple 1 :
$G =< \{a, b\}, \{S_1, S_2, S_3, S_4\}, P >$ où P consiste en les règles :

$$\begin{aligned} S_1 &\rightarrow aS_1 + bS_2 \\ S_2 &\rightarrow \varepsilon \\ S_3 &\rightarrow aS_4 + bS_2 \\ S_4 &\rightarrow aS_4 + bS_2 \end{aligned}$$

$S_1 \equiv S_3$ admet une preuve dans G-EQ :

Deux langages rationnels

Exemple 1 :

$$\dfrac{\dfrac{\dfrac{}{a \equiv a}\,ref \quad \dfrac{}{S_1 \equiv S_4}\,sj}{aS_1 \equiv aS_4} \times \dfrac{}{bS_2 \equiv bS_2}\,ref}{aS_1 + bS_2 \equiv aS_4 + bS_2} + \dfrac{\dfrac{}{S_4 \equiv aS_4 + bS_2}\,gr}{aS_4 + bS_2 \equiv S_4}\,sym}{aS_1 + bS_2 \equiv S_4}\,trans}{S_1 \equiv aS_1 + bS_2}\,gr}{S_1 \equiv S_4}\,trans \times \dfrac{}{bS_2 \equiv bS_2}\,ref}{aS_1 + bS_2 \equiv aS_4 + bS_2} + \dfrac{}{aS_4 + bS_2 \equiv S_3}\,gr}{aS_1 + bS_2 \equiv S_3}\,trans}{S_1 \equiv aS_1 + bS_2}\,gr}{S_1 \equiv S_3}\,trans$$

Deux langages algébriques

Exemple 2 :
$G =< \{a, b, c\}, \{S, T\}, P >$ où P consiste en les deux règles :

$$\begin{aligned} S &\rightarrow aSb + c \\ T &\rightarrow aTb + c \end{aligned}$$

$S \equiv T$ admet une preuve dans G-EQ :

Deux langages algébriques

Exemple 2

$$\dfrac{\dfrac{S \equiv aSb + c}{S \equiv T}\,gr \quad \dfrac{\dfrac{\dfrac{\dfrac{}{a \equiv a}\,ref \quad \dfrac{}{Sb \equiv Tb}\,sj}{aSb \equiv aTb}\times \quad \dfrac{}{c \equiv c}\,ref}{aSb + c \equiv aTb + c}+ \quad \dfrac{T \equiv aTb + c}{aTb + c \equiv T}\,gr}{aSb + c \equiv T}\,sym}{S \equiv T}\,trans \quad \dfrac{}{b \equiv b}\,ref}{Sb \equiv Tb}\,trans \times$$

Deux langages algébriques

Exemple 3 :la grammaire sur l'alphabet $\{a, b, c, x, D, F, \$, \#\}$:

$$
\begin{aligned}
\sigma &\to S + T \\
S &\to \$xFxbxaxaxaxaxaxaxaxcxDx\$S\$x + \$xSxDxFx\$ \\
&\quad + axSax + bxSbx + cxScx + DxSDx + FxSFx \\
&\quad + axbxbxSaxbx + cxcxbxScxbx + FxSaxFx \\
&\quad + bxSDxbx + FxScxFx + \# \\
U &\to \alpha + \alpha U \text{ pour } \alpha \in \{a, b, c, x, D, F, \$\} \\
T &\to \tau + \# \\
\tau &\to \alpha\tau\alpha + \alpha\tau'\beta \text{ pour } \alpha, \beta \in \{a, b, c, x, D, F, \$\}, \alpha \neq \beta \\
&\quad + \#U + U\# \\
\tau' &\to U\#U + U\# + \#U + \#
\end{aligned}
$$

Deux langages algébriques

Question :

$$\sigma \equiv T ?$$

Réponse : non
Témoin : w tel que

$$S \rightarrow^* w, \quad T \not\rightarrow^* w$$

Longueur du plus court témoin : $2^{2^8} \simeq 10^{60}$
Ecrire w , 1 caractere par top d'horloge, pour un processeur à 10 Gbits/s :
10^{50} secondes
Année : $365 \times 24 \times 3600 \simeq 4.10^7$ secondes
Ecriture du témoin : $\geq 10^{43}$ années $>>$ 5 milliards d'années !

Le problème

Théorème

Le problème de l'égalité entre deux langages rationnels est décidable.

référence : années 50.

Théorème

Le problème de l'égalité entre deux langages algébriques est indécidable.

référence : années 60.

Déterminisme

Une grammaire G est dite *déterministe* lorsque :
1- tout mot w engendré par G a un seul arbre de dérivation dans G
2- cet arbre peut être calculé en une seule passe de lecture de w de gauche à droite.
N.B. Les langages de programmation sont définis par des grammaires déterministes.

Le problème

Théorème

Le problème de l'inclusion entre deux langages algébriques déterministes est indécidable.

PB de l'égalite [Ginsburg-Greibach Inf. and Control, 1966] :
Le problème de l'égalité entre deux langages algébriques déterministes est-il indécidable ?

La recherche

- $\simeq$100 publications entre 1966 et 1996
- solutions *partielles* (approche "par le bas")
- Hopcroft-Korenjac : un seul état, pas de transitions spontanées
- Harrison-Havel-Yehudai : un seul état, pas de transitions spontanées (pour un des deux automates)
- Oyamaguchi-Honda-Inagaki : pas de transitions spontanées
- idées *générales* (approche "par le haut")
- Valiant : simulation en parallèle–> transformation de paires
- Courcelle : systèmes de preuve complets
- Meitus : analogie avec l'algèbre lineaire

La solution

Le Système de preuves d'égalité
G-EQ :

$$\frac{x \equiv y}{y \equiv x}\, sym \qquad \frac{x \equiv y \;\; y \equiv z}{x \equiv z}\, trans \qquad \frac{x \equiv x' \;\; y \equiv y'}{x + y \equiv x' + y'}\, +$$

G-EQ : (schemas de) règles strictes :

$$\frac{}{x \equiv x}\, ref \qquad \frac{x \equiv x' \;\; y \equiv y'}{x \cdot y \equiv x' \cdot y'}\, \times$$

G-EQ : (schemas de) règles non-strictes :

$$\frac{}{S \equiv \sum_{i=1}^{n} w_i}\, gr$$

La solution

Théorème

Le système de preuve G-EQ est complet i.e.
toute équivalence vraie *entre deux mots de la grammaire G est* prouvable *dans ce système.*

[GS TCS2001]

Corollaire

Le problème de l'égalité de deux langages algébriques déterministes est décidable.

[GS ICALP97-TCS2001]

Deux langages algébriques déterministes

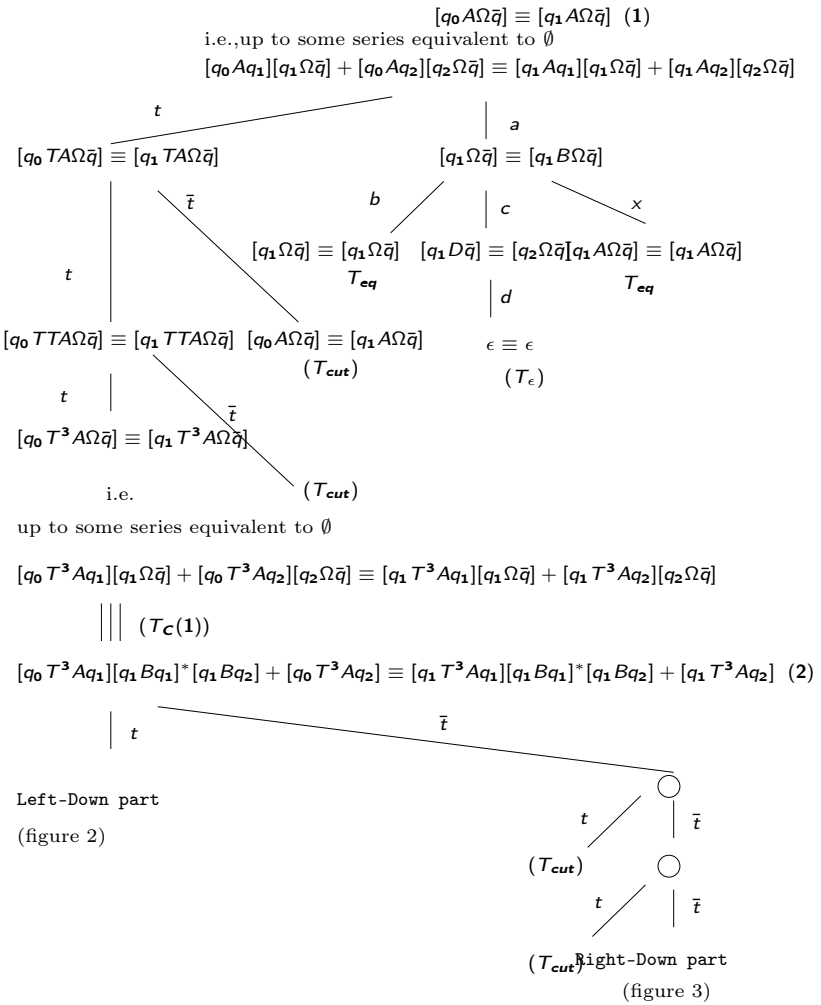


Figure: example 4 : haut

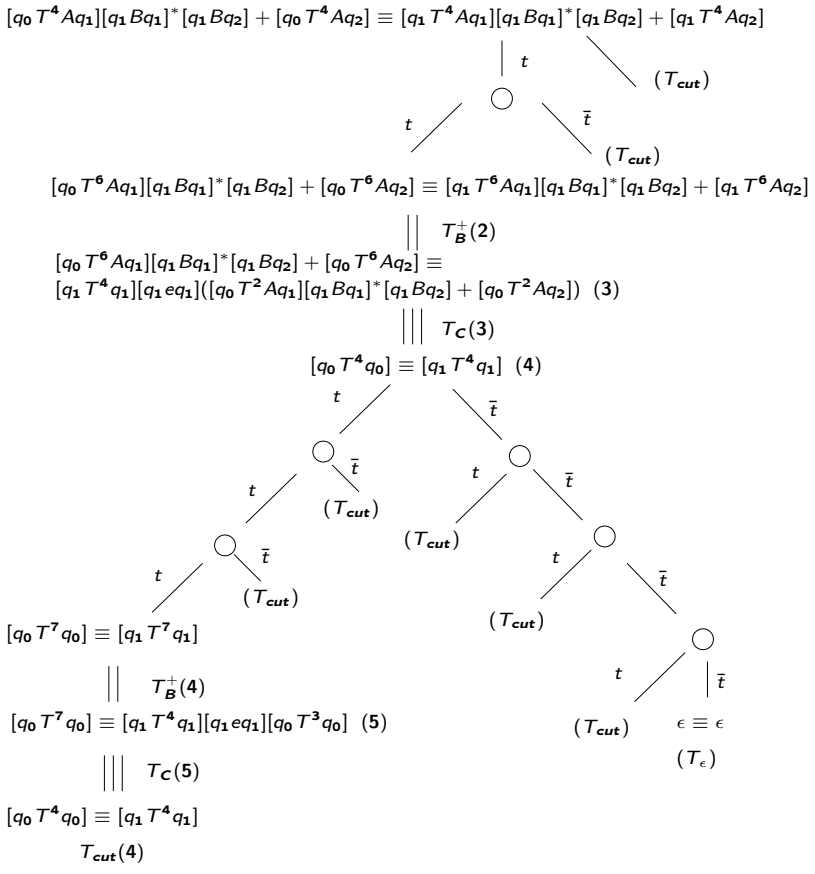


Figure: example 4 : bas-gauche

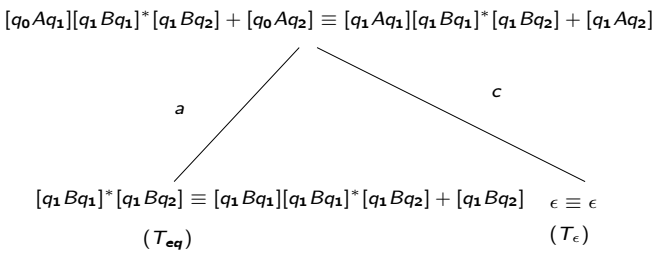


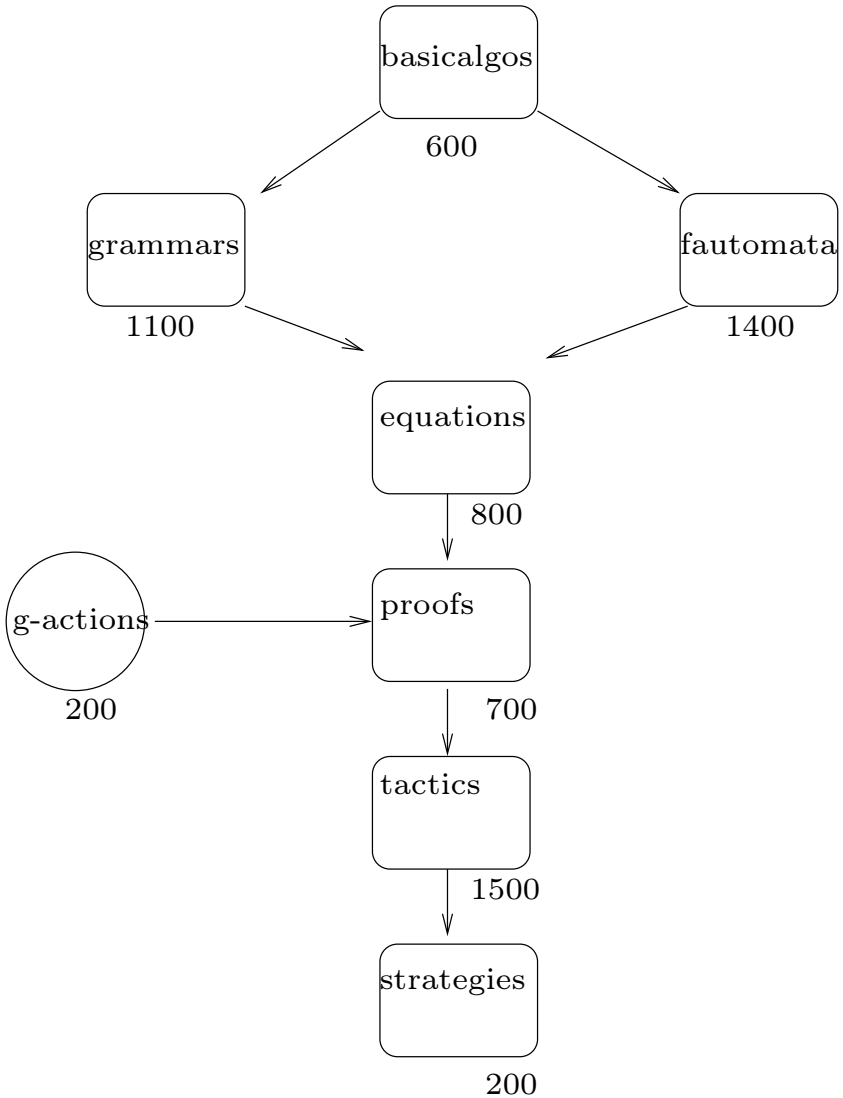
Figure: example 4 : bas-droit

De plus en plus effectif

Le programme

ENTRÉE :
Une grammaire strict-déterministe et deux mots W_1, W_2.
SORTIE :
- un témoin de $W_1 \not\equiv W_2$
- ou une preuve (compactée) de $W_1 \equiv W_2$

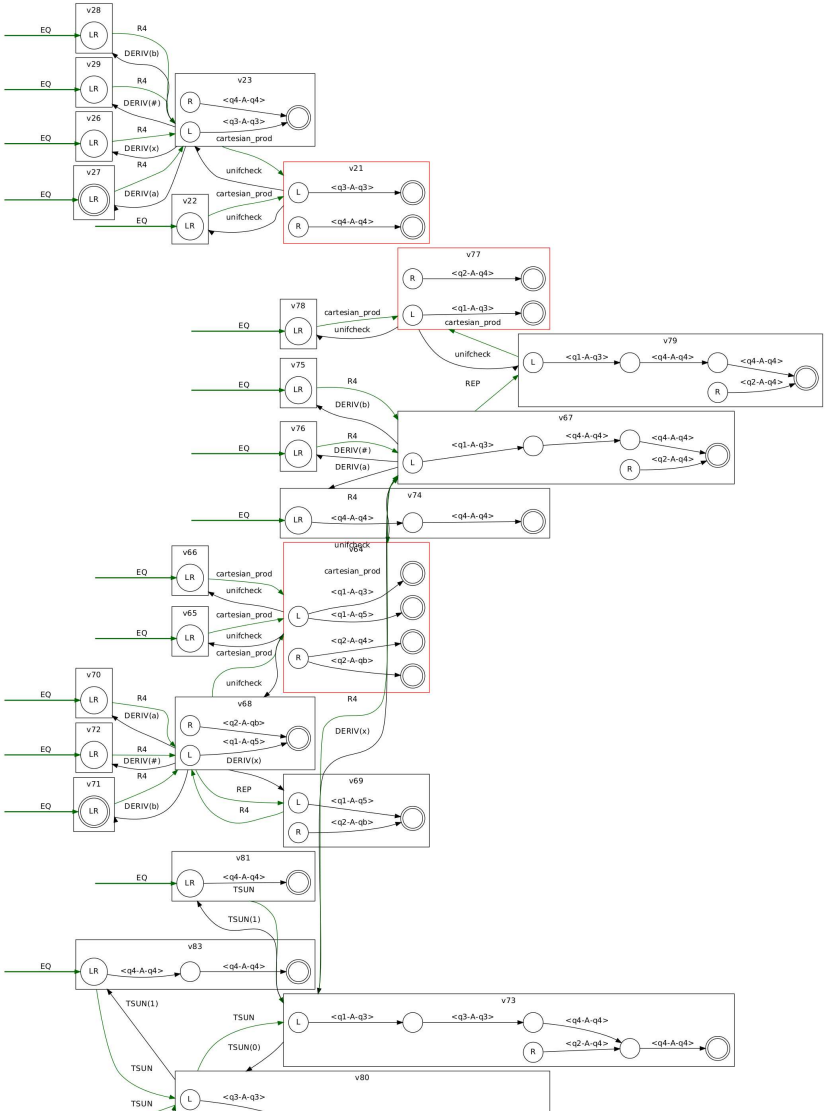
Le programme



Le programme en action

```
Terminal symbols : a b x
Rewriting rules:
<q1-A-q3>::=a
<q1-A-q3>::=x<q1-A-q3><q3-A-q3>
<q1-A-q5>::=b
<q1-A-q5>::=x<q1-A-q5>
<q1-O-q3>::=<q1-A-q3><q3-O-q3>
<q1-O-q3>::=<q1-A-q5>
<q2-A-q4>::=a<q4-A-q4><q4-A-q4>
<q2-A-q4>::=x<q2-A-q4><q4-A-q4>
<q2-A-qb>::=b
<q2-A-qb>::=x<q2-A-qb>
<q2-O-q3>::=<q2-A-q4><q4-O-q3>
<q2-O-q3>::=<q2-A-qb>
<q3-A-q3>::=a
<q3-O-q3>::=a<q3p-O-q3>
<q3b-O-q3>::=a
<q3p-O-q3>::=a<q3b-O-q3>
<q4-A-q4>::=a
<q4-O-q3>::=a
```

Le programme en action



Le programme en action

```
["U1",["x","V1","C1","Y","U2","X","U1"]], ["U1",["x","V1","C1","Y","U2","X","U2
["U1",["x","V1","C1","Y","U3","X","U1"]], ["U1",["x","V1","C1","Y","U3","X","U2
["U1",["x","V1","C1","Z","X","U1"]], ["U1",["x","V1","C1","Z","X","U2"]],
["U1",["x","V2","Y","U2","X","U1"]], ["U1",["x","V2","Y","U2","X","U2"]],
["U1",["x","V2","Y","U3","X","U1"]], ["U1",["x","V2","Y","U3","X","U2"]],
["U1",["x","V2","Z","X","U1"]], ["U1",["x","V2","Z","X","U2"]], ["U1",["x","V3"
["U1",["x","V3","Y","U2","X","U2"]], ["U1",["x","V3","Y","U3","X","U1"]],
["U1",["x","V3","Y","U3","X","U2"]], ["U1",["x","V3","Z","X","U1"]], ["U1",["x"
["U1",["y","Y","U2","X","U1"]], ["U1",["y","Y","U2","X","U2"]], ["U1",["y","Y",
["U1",["y","Y","U3","X","U2"]], ["U1",["y","Z","X","U1"]], ["U1",["y","Z","X","
["U2",["x","V1","C1","Y","U2","Y","U4","C1"]], ["U2",["x","V1","C1","Y","U2","Y
["U2",["x","V1","C1","Y","U3","Y","U4","C1"]], ["U2",["x","V1","C1","Y","U3","Y
```

Le programme en action

["U2",["x","V1","C1","Z","Y","U4","C1"]], ["U2",["x","V1","C1","Z","Y","U2"]],
["U2",["x","V2","Y","U2","Y","U4","C1"]], ["U2",["x","V2","Y","U2","Y","U2"]],
["U2",["x","V2","Y","U3","Y","U4","C1"]], ["U2",["x","V2","Y","U3","Y","U2"]],
["U2",["x","V2","Z","Y","U4","C1"]], ["U2",["x","V2","Z","Y","U2"]], ["U2",["x"
["U2",["x","V3","Y","U2","Y","U2"]], ["U2",["x","V3","Y","U3","Y","U4","C1"]],
["U2",["x","V3","Y","U3","Y","U2"]], ["U2",["x","V3","Z","Y","U4","C1"]],
["U2",["x","V3","Z","Y","U2"]], ["U2",["y","Y","U2","Y","U4","C1"]], ["U2",["y"
["U2",["y","Y","U3","Y","U4","C1"]], ["U2",["y","Y","U3","Y","U2"]], ["U2",["y"
["U2",["y","Z","Y","U2"]], ["U2",["x","V1","C1","Y","U2","Z"]], ["U2",["x","V1"
["U2",["x","V1","C1","Z","Z"]], ["U2",["x","V2","Y","U2","Z"]], ["U2",["x","V2"
["U2",["x","V2","Z","Z"]], ["U2",["x","V3","Y","U2","Z"]], ["U2",["x","V3","Y",

Le programme en action

["U2",["x","V3","Z","Z"]], ["U2",["y","Y","U2","Z"]], ["U2",["y","Y","U3","Z"]]
["U2",["y","Z","Z"]], ["U3",["x","V1","C1","Y","U4","C1"]], ["U3",["x","V1","C1
["U3",["x","V2","Y","U4","C1"]], ["U3",["x","V2","Y","U4","C2"]], ["U3",["x","V
["U3",["x","V3","Y","U4","C2"]], ["U3",["y","Y","U4","C1"]], ["U3",["y","Y","U4
["U4",["x","V1","C2"]], ["U4",["x","V4"]], ["V1",["x","U1","Y","V1","C2"]],
["V1",["x","U1","Y","V2"]], ["V1",["x","U1","Z"]], ["V1",["x","U3","Y","V1","C2
["V1",["x","U3","Y","V2"]], ["V1",["x","U3","Z"]], ["V1",["x","U4","C1","Y","V1
["V1",["x","U4","C1","Y","V2"]], ["V1",["x","U4","C1","Z"]], ["V1",["y","Y","V1
["V1",["y","Y","V2"]], ["V1",["y","Z"]], ["V2",["x","U1","Y","V3"]], ["V2",["x"

Le programme en action

```
["V2",["x","U3","Y","V3"]], ["V2",["x","U3","Y","V4"]], ["V2",["x","U4","C1","Y
["V2",["x","U4","C1","Y","V4"]], ["V2",["y","Y","V3"]], ["V2",["y","Y","V4"]],
["V3",["x","U2","X","V1","C1"]], ["V3",["x","U2","X","V1","C2"]], ["V3",["x","U
["V3",["x","U4","C2","X","V1","C2"]], ["V4",["x","U2","Y","V2"]], ["V4",["x","U
["V4",["x","U2","Z"]], ["V4",["x","U4","C2","Y","V2"]], ["V4",["x","U4","C2","Y
["V4",["x","U4","C2","Z"]], ["C1",["x","U1"]], ["C1",["x","U2"]], ["C2",["y","U
["C2",["y","U2"]], ["C2",["z"]]],
```

Le programme en action

$$U_1 \equiv V_1 C_1 ?$$

$$[2, ['x', 'y', 'y', 'x', 'y', 'z', 'y', 'y', 'z', 'z', 'y', 'y', 'z', 'z', 'z', 'x', 'y', 'z', 'z']]$$

Utiliser le programme

cpp2dpa

Abstract

cpp2dpa is a tool to convert cryptographic protocols into deterministic pushdown automata, enabling proofs of equivalence and discovery of attacks. The procedure is detailed in the journal version of this article (http://www.lsv.ens-cachan.fr/Publis/PAPERS/PDF/CCD-icalp13.pdf), which contains detailed proofs, implementation details and additional examples.

cpp2dpa is coded in Python 3. Here are the sources (cpp2dpa.tar.gz).

To carry the proofs or find attacks, cpp2dpa is interfacing with a modified version of the LALBLC (http://dept-info.labri.u-bordeaux.fr/~ges/LALBLC/lalblc.html) tool which requires the Numpy package. This version includes optimized conversion from automata to grammars and the possibility to store grammars. The source is available here (LALBLC.tar.gz).

Installing cpp2dpa

You first need to install Python3 (https://www.python.org/downloads/), along with the Numpy module, available here (http://sourceforge.net/projects/numpy/files/). To install Numpy, assuming you run a Unix system and extracted the sources:

```
python3 setup.py build
python3 setup.py install
```

Then, to install LALBLC, you first need to extract the archive

Utiliser le programme

Application à la Cryptologie.
[Chrétien-Cortier-Delaune ICALP'13]
Principe :
Correction d'un protocole cryptographique –> une grammaire strict-déterministe et deux non-terminaux S, T tels que :

$$\text{protocole est correct } \Leftrightarrow S \equiv T.$$

Programme de conversion des protocoles en grammaires :cpp2dpa
`http://www.lsv.ens-cachan.fr/ chretien/cpp2dpa.php`
Exemple : protocole de e-passeport Francais (2011)

$$S \not\equiv T$$

Version corrigée :

$$S \equiv T$$

Utiliser le programme

output pattern). Here are the two protocols of the first running example from the article.

```
passport1 = Protocol(['x'],
                [
                    # Making mackm(2) a signature-like primitive
                    ('cmac','x.mackm','x'),
                    ('cmac','x.mackm2','x'),
                    # Giving a message from a previous session
                    ('c0', 'start', 'nr0.kr0.np0.ke.mackm'),
                    # The protocol
                    # The reader is not modeled
                    ('c1', 'start', 'nr.kr.np.ke.mackm'),
                    ('c2', 'x.ke.mackm', 'x.qmacok'),
                    ('c3', 'x.np.qmacok', 'x.np.kp.mackm'),
                    ])

passport2 = Protocol(['x'],
                [
                    # Making mackm(2) a signature-like primitive
                    ('cmac','x.mackm','x'),
                    ('cmac','x.mackm2','x'),
                    # Giving a message from a previous session
                    ('c0', 'start', 'nr0.kr0.np0.ke.mackm'),
                    # The protocol
                    # The reader is not modeled
                    ('c1', 'start', 'nr.kr.nq.ke2.mackm2'),
                    ('c2', 'x.ke2.mackm2', 'x.qmacok'),
                    ('c3', 'x.nq.qmacok', 'x.nq.kq.mackm2'),
                    ])
```

Other examples are avaiblable in the Examples folder, containing,

Statut des résultats

Statut des résultats

CHAÎNE

d'arguments	(mathématiques),
d'instructions	(algorithmes, programme),
de calculs	(physiques),
de résultats	(preuves).

Statut des résultats : Logique

Preuve de DÉCIDABILITÉ
[TCS 2001, pages 11-81 V]érifiée par des “recenseurs”.
Preuve de bonne-fondation et COMPLÉTUDE de G-EQ.
[TCS 2001, pages 82-100 P]ublic - Vérifiable-
Vérifiée par des “recenseurs” ?.

Statut des résultats : Algorithmique

ALGORITHMES de construction de preuves :
* calcul des coordonnees
* “ remontée” du témoin
* usage de nouvelles stratégies $\neq$ stratégie utilisée dans la preuve de complétude.
* ...

Spécification [P. Henry-G.Sénizergues].
Décrits dans [CIAA 2013].
mais pas prouvés.

Statut des résultats : Programme

PROGRAMME.
Public : adresse web
Testé : 3000 lignes de tests.
Utilisé par des cryptologues.
Pas de preuve de correction du programme.

Statut des résultats : Exécutions

EXÉCUTION du programme
Interprète Python :
public ;testé ;non-prouvé
PREUVES dans G-EQ :
Non produites explicitement (version concise) ; Format exotique.

Développer le programme

- produire une preuve dans G-EQ (ou un témoin)
- (petit) module qui teste qu'un ensemble de paires est une preuve pour G-EQ (qu'un mot est un témoin).
- mieux : produire une preuve en DEDUKTI.

Coopération avec INRIA-Paris (Deducteam).

De plus en plus général

Transducteurs

Transducteurs vers les mots, vers les groupes linéaires.
oui : [G. Sénizergues ICALP 1999]

Non-déterminisme

Automates non-deterministes, mais equivalence plus restrictive (bisimulation)
oui : [Sénizergues SIAM 2005]
mais tres tres complexe [Goeller 2013]
(programme improbable ...)

Arbres

Automates deterministes à pile sur les arbres :
cas descendant : oui.
cas ascendant : ouvert.

Piles de piles

Automates deterministes à pile de piles, sur les mots : ouvert.


À LA RECHERCHE
DU TEMPS PERDU
MARCEL PROUST
Format Kindle

Jacques Hadamard : la psychologie de l'invention, 1945
(psychologie, cité par S. Dehaene "le code de la conscience", 2014)

Jean-Francois Sabouret : Chercher jour apres jour, les aventuriers du savoir, 2000
(interview de chercheurs par des élèves-journalistes)

David Lodge : Un tout petit monde, 1984
(roman)

Bruno Latour : La Vie de laboratoire. La production des faits scientifiques, 1988.
(sociologie des sciences)