Some Applications of the Decidability of DPDA's Equivalence

Géraud Sénizergues

LaBRI, Université de Bordeaux I
ges@labri.u-bordeaux.fr
http://www.dept-info.labri.u-bordeaux.fr/∼ges/

Abstract. The *equivalence* problem for deterministic pushdown automata has been shown decidable in [Sén97c,Sén97a,Sén97b,Sén01,Sti99]. We give some applications of this decidability result to other problems arising in the following areas of theoretical computer science:

- programming languages theory
- infinite graph theory
- Thue-systems

1 Introduction

We describe, in this extended abstract, several applications of the decidability of the equivalence problem for dpda (we denote this problem by Eq(D,D) in the sequel), in the following areas

- programming languages theory
- infinite graph theory
- Thue-systems

We have left out of this description applications to other equivalence problems in formal language theory; such applications are already described in [Sén01, p.155-156].

What we call an *application* may be either a true application of theorem 1 or a result obtained by a method which is a variant of the method used in [Sén01]. In many cases the "application" to a problem P will follow from a reduction from problem P to Eq(D,D) which already existed in the literature (i.e. the reduction was established before Eq(D,D) was solved). In some cases (corollary 5, theorem 14, theorem 16) we shall give a *new* reduction of some problem P to Eq(D,D). In all cases we try to sketch the landscape around every application.

2 Deterministic Pushdown Automata

In this section we just recall the basic definitions on dpda and the result about decidability of the equivalence problem. More details about either the proof of this result or the historical development of the works around this problem can be found elsewhere (see [Sén97a] for the solution, [Sén01, section 1.1] for a general historical outline and [Sén00a] for an historical perspective on the proof-systems used in the solution).

M. Margenstern and Y. Rogozhin (Eds.): MCU 2001, LNCS 2055, pp. 114–132, 2001.

2.1 Pushdown Automata

A *pushdown automaton* on the alphabet X is a 7-tuple $\mathcal{M}=<X, Z, Q, \delta, q_0, z_0, F>$ where Z is the finite stack-alphabet, Q is the finite set of states, $q_0 \in Q$ is the initial state, z_0 is the initial stack-symbol, F is a finite subset of QZ^*, the set of *final* configurations, and δ, the transition function, is a mapping $\delta : QZ \times (X \cup \{\epsilon\}) \to \mathcal{P}_f(QZ^*)$.

Let $q, q' \in Q, \omega, \omega' \in Z^*, z \in Z, f \in X^*$ and $a \in X \cup \{\epsilon\}$; we note $(qz\omega, af) \longmapsto_{\mathcal{M}} (q'\omega'\omega, f)$ if $q'\omega' \in \delta(qz, a)$. $\overset{*}{\longmapsto}_{\mathcal{M}}$ is the reflexive and transitive closure of $\longmapsto_{\mathcal{M}}$.

For every $q\omega, q'\omega' \in QZ^*$ and $f \in X^*$, we note $q\omega \overset{f}{\longrightarrow}_{\mathcal{M}} q'\omega'$ iff $(q\omega, f) \overset{*}{\longmapsto}_{\mathcal{M}} (q'\omega', \epsilon)$.

$\mathcal{M}$ is said *deterministic* iff, for every $z \in Z, q \in Q, x \in X$:

$$\mathrm{Card}(\delta(qz, \epsilon)) \in \{0, 1\} \tag{1}$$

$$\mathrm{Card}(\delta(qz, \epsilon)) = 1 \Rightarrow \mathrm{Card}(\delta(qz, x)) = 0, \tag{2}$$

$$\mathrm{Card}(\delta(qz, \epsilon)) = 0 \Rightarrow \mathrm{Card}(\delta(qz, x)) \leq 1. \tag{3}$$

A configuration $q\omega$ of $\mathcal{M}$ is said ϵ-bound iff there exists a configuration $q'\omega'$ such that $(q\omega, \epsilon) \longmapsto_{\mathcal{M}} (q'\omega', \epsilon)$; $q\omega$ is said ϵ-free iff it is not ϵ-bound.
A pda $\mathcal{M}$ is said *normalized* iff, it fulfills conditions (1), (2) (see above) and (4),(5),(6):

$$q_0 z_0 \text{ is } \epsilon - \text{free} \tag{4}$$

and for every $q \in Q, z \in Z, x \in X$:

$$q'\omega' \in \delta(qz, x) \Rightarrow \mid \omega' \mid \leq 2, \tag{5}$$

$$q'\omega' \in \delta(qz, \epsilon) \Rightarrow \mid \omega' \mid = 0. \tag{6}$$

The *language recognized* by $\mathcal{M}$ is

$$\mathrm{L}(\mathcal{M}) = \{w \in X^* \mid \exists c \in F, q_0 z_0 \overset{w}{\longrightarrow}_{\mathcal{M}} c\}.$$

We call *mode* every element of $QZ \cup \{\epsilon\}$. For every $q \in Q, z \in Z$, qz is said *ϵ-bound* (respectively ϵ-free) iff condition (2) (resp. condition (3)) in the above definition of deterministic automata is realized. The mode ϵ is said ϵ-free. We define a mapping $\mu : QZ^* \to QZ \cup \{\epsilon\}$ by

$$\mu(\epsilon) = \epsilon \text{ and } \mu(qz \cdot \omega) = qz,$$

for every $q \in Q, z \in Z, \omega \in Z^*$. For every $c \in QZ^* \cup \{\epsilon\}$, $\mu(c)$ is called the mode of c. The configuration c is said ϵ-bound (resp. ϵ-free) if and only if $\mu(c)$ has the corresponding property.

2.2 The Equivalence Problem

The *equivalence problem* for dpda was raised in [GG66]. It is the following decision problem:

INSTANCE: Two dpda $\mathcal{A}, \mathcal{B}$, over the same terminal alphabet X.
QUESTION: $\mathrm{L}(\mathcal{A}) = \mathrm{L}(\mathcal{B})$?

Theorem 1. *The* equivalence problem *for dpda is decidable.*

The result is exposed in [Sén97c,Sén97a] and proved in [Sén97b,Sén01], see in [Sti99] some simplifications.

3 Programming Languages

3.1 Semantics

The equivalence problem for program schemes
Let us say that two programs P, Q are *equivalent* iff, on every given input, either they both diverge or they both converge and compute the same result. It would be highly desirable to find an algorithm deciding this equivalence between programs since, if we consider that P is really a program and Q is a specification, this algorithm would be a "universal program-prover". Unfortunately one can easily see that, as soon as the programs P, Q compute on a sufficiently rich structure (for example the ring of integers), this notion of equivalence is undecidable. Nevertheless, this seemingly hopeless dream lead many authors to analyze the *reason why* this problem is undecidable and the suitable *restrictions* (either on the shape of programs or on the meaning of the basic operations they can perform) which might make this equivalence decidable. Informally, one can define an *interpretation* as an "universe of objects together with a certain definite meaning for each program primitive as a function on this universe" and a *program scheme* as a "program without interpretation"([Mil70, p.205,lines 5-13]). Several precise mathematical notions of "interpretation" and "program schemes" were given and studied ([Ian60],[Rut64],[Pat67],[Kap69],[Mil70],[LPP70],[GL73],[Niv75],[Ros75],[Fri77], [Cou78a],[Cou78b],[Gue81], see [Cou90b] for a survey). Many methods for either transforming programs or for proving properties of programs were established but, concerning the equivalence problem, the results turned out to be mostly negative: for example, in [LPP70, p.221, lines 24-26], the authors report that "for almost any reasonable notion of equivalence between computer programs, the two questions of equivalence and nonequivalence of pairs of schemas are *not* partially decidable". Nevertheless, two kinds of program schemes survived all these studies:

– the *monadic recursion schemes* where a special ternary function `if-then-else` has the fixed usual interpretation: in [GL73] the equivalence-problem for such schemes is reduced to the equivalence problem for dpda and in [Fri77] a reduction in the opposite direction is constructed,

- the *recursive polyadic program schemes*: in [Cou78a,Cou78b], following a representation principle introduced in [Ros75], the equivalence-problem for such schemes is reduced to the equivalence problem for dpda and conversely.

Corollary 2. *The equivalence problem for monadic recursion schemes (with interpreted* `if-then-else`*), is decidable.*

This follows from theorem 1 and the reduction given in [GL73].

Corollary 3. *The equivalence problem for recursive polyadic program schemes (with completely uninterpreted function symbols) is decidable.*

This follows from theorem 1 and the reduction given in [Cou78a, theorem 3.25] or the reduction given in [Gal81, corollary 4.4].

Interpretation by trees Let us precise now what a "recursive polyadic" program scheme is. Let (F, ρ) be some *ranked alphabet* i.e. a set F and a map $\rho : F \to \mathbb{N}$. A *tree* over the ranked alphabet (F, ρ) is a partial mapping $t : (\mathbb{N} - \{0\})^* \to F$ satisfying the following conditions:
C1: $\mathrm{dom}(t) \subseteq (\mathbb{N} - \{0\})^*$ is prefix-closed, i.e. if $\alpha \cdot \beta \in \mathrm{dom}(t)$, then $\alpha \in \mathrm{dom}(t)$.
C2: if $\rho(t(\alpha)) = k$, then, for every $i \in \mathbb{N}$, $\alpha i \in \mathrm{dom}(t) \Leftrightarrow 1 \leq i \leq k$.

We denote by $\mathrm{M}(F)$ (resp. $\mathrm{M}^\infty(F)$) the set of finite trees (resp. the set of all trees, finite or infinite) over F. With every symbol f of arity $k = \rho(f)$ is associated a k-ary operation $\hat{f} : \mathrm{M}(F)^k \to \mathrm{M}(F)$ defined by $\hat{f}(t_1, \ldots, t_k)$ is the unique tree which has a root labelled by f and which has k subtrees at depth 1, the ith subtree being t_i.

A *system of algebraic equations* Σ over F is defined as follows:
let $\Phi = \{\varphi_1, \ldots, \varphi_n\}$ be a ranked alphabet of "unknowns" (we denote by k_i the arity of φ_i) and let $X = \{x_1, \ldots, x_m\}$ be an alphabet of "variables", which are 0-ary symbols. Σ is a set of n equations of the form

$$\varphi_i(x_1, x_2, \ldots, x_{k_i}) = T_i \tag{7}$$

where every T_i is some element of $\mathrm{M}(F \cup \Phi, X)$. A *solution* of Σ is a tuple $(t_1, \ldots, t_n) \in (\mathrm{M}^\infty(F \cup \Phi, X))^n$ such that, for every $1 \leq i \leq n$:

$$\hat{t_i}(x_1, x_2, \ldots, x_m) = \hat{T_i}(x_1, x_2, \ldots, x_m) \tag{8}$$

where, $\hat{t_i}$ is the operation obtained by interpreting every symbol f by the corresponding operation $\hat{f}$ and $\hat{T_i}$ is the operation obtained by interpreting every symbol φ_i (resp. f) by the corresponding operation $\hat{t_i}$ (resp. $\hat{f}$). Σ will be said *in normal form* iff every tree T_i has its root labelled by an element of F. In such a case Σ has a *unique* solution. A program scheme is a pair (Σ, T) where Σ is an algebraic system of equations over F and T is a particular finite tree $T \in \mathrm{M}(F \cup \Phi, X)$. The *tree computed* by the scheme is then

$$t = \hat{T}(x_1, x_2, \ldots, x_m).$$

What corollary 3 precisely means is that the following problem is decidable:

INSTANCE: Two program schemes $(\Sigma_1, T_1), (\Sigma_2, T_2)$ (assumed in normal form)
QUESTION: Do these two schemes compute the same algebraic tree ?

Let us describe now some other F-magmas which extend $(\mathrm{M}^\infty(F), (\hat{f})_{f \in F})$. It turns out that corollary 3 is true, as well, in these structures.

Interpretation by formal power series
The idea of this link with formal power series is due to [Mat95]. Let F_2 be the field with 2 elements (i.e. $(\mathbb{Z}/2\mathbb{Z}, +, \cdot)$). Let $D = F_2[[X_1, X_2, \ldots, X_m, Y]]$ be the set of formal power series with $m+1$ commutative undeterminates and coefficients in F_2. We consider the two binary operations $\bar{f}, \bar{g}$ over D:

$$\bar{f}(S,T) = Y \cdot S^2 + Y^2 \cdot T^2; \;\; \bar{g}(S,T) = 1 + Y \cdot S^2 + Y^2 \cdot T^2.$$

Given an element $S \in D$ let us define the associated operation $\bar{S} : D^m \to D$ by

$$\bar{S}(S_1, S_2, \ldots, S_m) = S \circ (S_1, S_2, \ldots, S_m, Y)$$

i.e. the series obtained by substituting S_i to the undeterminate X_i and leaving Y unchanged.
Let Σ be a system of equations (7) over the ranked alphabet $F = \{f, g\}$, with $\rho(f) = \rho(g) = 2$. A solution in D of (7) is a n-tuple $(S_1, S_2, \ldots, S_n)$ such that for every $1 \leq i \leq n$:

$$\bar{S}_i(X_1, X_2, \ldots, X_m) = \bar{T}_i(X_1, X_2, \ldots, X_m) \tag{9}$$

where, $\bar{S}_i$ is the operation defined above and $\bar{T}_i$ is the operation obtained by interpreting every symbol φ_i (resp. f, g) by the corresponding operation $\bar{S}_i$ (resp by $\bar{f}, \bar{g}$).

Let us define a map $\mathcal{S} : \mathrm{M}^\infty(F, \{x_1, x_2, \ldots, x_m\}) \to F_2[[X_1, X_2, \ldots, X_m, Y]]$ by:for every $t_1, t_2 \in \mathrm{M}^\infty(F, \{x_1, x_2, \ldots, x_m\})$

$$\mathcal{S}(x_i) = X_i; \;\; \mathcal{S}(f(t_1, t_2)) = \bar{f}(\mathcal{S}(t_1), \mathcal{S}(t_2)); \;\; \mathcal{S}(g(t_1, t_2)) = \bar{g}(\mathcal{S}(t_1), \mathcal{S}(t_2)).$$

One can check that $\mathcal{S}$ is an injective homomorphism from the magma $(\mathrm{M}^\infty(F, \{x_1, x_2, \ldots, x_m\}), \hat{f}, \hat{g})$ into the magma $(F_2[[X_1, X_2, \ldots, X_m, Y]], \bar{f}, \bar{g})$. Moreover the map $\mathcal{S}$ is *compatible with substitution*, in the following sense: $\forall t \in \mathrm{M}^\infty(F, \{x_1, x_2, \ldots, x_m\}, \forall (u_1, \ldots, u_m) \in (\mathrm{M}^\infty(F, \{x_1, x_2, \ldots, x_m\})^m$,

$$\mathcal{S}(\hat{t}(u_1, \ldots, u_m)) = \overline{\mathcal{S}(t)}(\mathcal{S}(u_1), \ldots, \mathcal{S}(u_m)).$$

(This last property follows from the fact that $\bar{f}, \bar{g}$ were chosen to be *polynomial* operators).
The fact that $\mathcal{S}$ is an homomorphism and that it is compatible with substitution show that $\mathcal{S}$ maps every solution of Σ in $\mathrm{M}^\infty(F, \{x_1, x_2, \ldots, x_m\})$ to a solution of Σ in $F_2[[X_1, X_2, \ldots, X_m, Y]]$. The hypothesis that Σ is normal implies that it has a unique solution in $F_2[[X_1, X_2, \ldots, X_m, Y]]$ too. In conclusion, two algebraic systems of equations Σ_1, Σ_2 have the same solution (resp. the same first component of solution) in $F_2[[X_1, X_2, \ldots, X_m, Y]]$ iff they have the same solution (resp. the same first component of solution) in $\mathrm{M}^\infty(F, X)$.

Corollary 4. *The equality problem for series in* $F_2[[X_1, X_2, \ldots, X_m, Y]]$ *defined by an algebraic system of equations (with polynomial operators* $\bar{f}, \bar{g}$ *and with substitution) is* decidable*.*

Let us notice that for any finite field F_q (with $q = p^k$, p prime), one can introduce the polynomial operators:

$$\bar{f}(S_1, S_2, \ldots, S_p) = f + YS_1^p + \ldots + Y^j S_j^p + \ldots + Y^p S_p^p \quad (\text{ for } f \in F_q).$$

By the same type of arguments one obtains

Corollary 5. *The equality problem for series in* $F_q[[X_1, X_2, \ldots, X_m, Y]]$ *defined by an algebraic system of equations (with polynomial operators* $(\bar{f})_{f \in F_q}$ *and with substitution) is* decidable*.*

3.2 Types

Some works on *recursively defined parametric types* have raised the question whether it is possible to decide if two such types are equivalent or not. It has been shown in [Sol78] that this problem is reducible to the equivalence problem for dpda. Therefore we can state

Corollary 6. *The equivalence problem for recursively defined types is decidable.*

This follows from theorem 1 and the reduction given in [Sol78].

4 Graphs

We describe here several problems arising in the study of infinite graphs. Such infinite graphs arise naturally in computer science either as representing all the possible computations of a program (the infinite expansion of a recursive program scheme), or all the possible behaviours of a process, etc... They also arise in computational group theory as a geometrical view of the group (the so-called *Cayley*-graph of the group).
When the chosen class of graphs is well-related with pushdown automata, one can draw from either theorem 1 or its proof method some clue for solving decision problems on these graphs.

4.1 Bisimulations-Homomorphisms

Let X be a finite alphabet. We call *graph over* X any pair $\Gamma = (V_\Gamma, E_\Gamma)$ where V_Γ is a set and E_Γ is a subset of $V_\Gamma \times X \times V_\Gamma$. For every integer $n \in \mathbb{N}$, we call an n-graph every $(n+2)$-tuple $\Gamma = (V_\Gamma, E_\Gamma, v_1, \ldots, v_n)$ where (V_Γ, E_Γ) is a graph and $(v_1, \ldots, v_n)$ is a sequence of distinguished vertices: they are called the *sources* of Γ.
A 1-graph (V, E, v_1) is said to be *rooted* iff v_1 is a root of (V, E). Let Γ, Γ' be two n-graphs over X.

Definition 7 *$\mathcal{R}$ is a* simulation *from Γ to Γ' iff*

1. $\mathrm{dom}(\mathcal{R}) = V_\Gamma$,
2. $\forall i \in [1,n], (v_i, v'_i) \in \mathcal{R}$,
3. $\forall v, w \in V_\Gamma, v' \in V_{\Gamma'}, x \in X,$ *such that* $(v,x,w) \in E_\Gamma$ *and* $v\mathcal{R}v'$,

$$\exists w' \in V_{\Gamma'} \text{ such that } (v',x,w') \in E_{\Gamma'} \text{ and } w\mathcal{R}w'.$$

$\mathcal{R}$ is a bisimulation *iff both $\mathcal{R}$ and $\mathcal{R}^{-1}$ are simulations.*

In the case where $n = 0$ this definition coincides with the classical one from [Par81], [Mil89].

Definition 8 *We call* homorphism *from Γ to Γ' any mapping: $h : V_\Gamma \to V_{\Gamma'}$ such that h is a bisimulation in the sense of definition 7.*
h is called an isomorphism *iff h is a bijective bisimulation.*

4.2 Some Classes of Infinite Graphs

Equational Graphs. The reader can find in [Cou89,Bau92] formal definitions of what an *equational* graph is. However, theorem 9 below gives a strong link with pda.
We call *transition-graph* of a pda $\mathcal{M}$, denoted $\mathcal{T}(\mathcal{M})$, the 0-graph:
$\mathcal{T}(\mathcal{M}) = (V_{\mathcal{T}(\mathcal{M})}, E_{\mathcal{T}(\mathcal{M})})$ where $V_{\mathcal{T}(\mathcal{M})} = \{q\omega \mid q \in Q, \omega \in Z^*, q\omega \text{ is } \epsilon - \text{free}\}$ and

$$E_{\mathcal{T}(\mathcal{A})} = \{(c,x,c') \in V_{\mathcal{T}(\mathcal{M})} \times V_{\mathcal{T}(\mathcal{M})} \mid c \xrightarrow{x}_{\mathcal{M}} c'\}. \tag{10}$$

We call *computation 1-graph* of the pda $\mathcal{M}$, denoted $(\mathcal{C}(\mathcal{M}), v_\mathcal{M})$, the subgraph of $\mathcal{T}(\mathcal{M})$ induced by the set of vertices which are accessible from the vertex q_0z_0, together with the source $v_\mathcal{M} = q_0z_0$.

Theorem 9. *Let $\Gamma = (\Gamma_0, v_0)$ be a rooted 1-graph over X. The following conditions are equivalent:*

1. *Γ is equational and has finite out-degree.*
2. *Γ is isomorphic to the computation 1-graph $(\mathcal{C}(\mathcal{M}), v_\mathcal{M})$ of some normalized pushdown automaton $\mathcal{M}$.*

A proof of this theorem is sketched in [Sén00b, Annex,p.94-96].

Algebraic Trees. Let (F,ρ) be some *ranked alphabet* i.e. a set F and a map $\rho : F \to \mathbb{N}$. The notion of a *tree* over the ranked alphabet (F,ρ) has been recalled in §3.1. With such a tree one associates a new alphabet

$$Y = \{[f,k] \mid f \in F, 1 \leq k \leq \rho(f)\}. \tag{11}$$

Let $\alpha \in \mathrm{dom}(t) : \alpha = i_1 i_2 \ldots i_n$. The word associated with α is : $brch(\alpha) = [f_0, i_1][f_1, i_2] \ldots [f_{n-1}, i_n]$ where α_j is the prefix of α of length j and $f_j = t(\alpha_j)$. The *branch-language* associated to t is then:

$$Brch(t) = \{brch(\alpha) \mid \alpha \in \mathrm{dom}(t)\}.$$

One can easily check that, if t, t' are trees over F, with no leaf, then t, t' are isomorphic iff $Brch(t) = Brch(t')$. The notion of *algebraic* tree over F can be defined in terms of algebraic equations in the magma of (infinite) trees over F: $t \in \mathrm{M}(F)$ is algebraic iff there exists some normal system of n equations Σ over F (see equation (7) in §3.1)such that, t is the first component of its unique solution. Anyway, the following characterization is sufficient for our purpose

Theorem 10. *Let t be a tree over the ranked alphabet (F, ρ), without any leaf. t is algebraic iff the language $Brch(t)$ is deterministic context-free.*

A more general version valid even for trees having some leaves is given in [Cou83, Theorem 5.5.1, point 2]. The trees considered above can be named *ordered* trees because the sons of every vertex α are given with a fixed order: $\alpha 1, \alpha 2, \dots, \alpha k$. We call *unordered* tree the oriented graph $un(t)$, with vertices labelled over F and unlabelled edges, obtained by forgetting this ordering of the sons in an ordered tree t. An *algebraic unordered* tree is then a tree of the form $un(t)$ for some algebraic ordered tree t.

Automatic Graphs. The general idea underlying the notion of an *automatic* graph is that of a graph whose set of vertices is fully described sy some *rational* set of words and whose set of edges is fully described by some *rational* set of pairs of words. Several precise technical definitions following this general scheme have been studied in [Sén92,KN95,Pel97,CS99,BG00,Mor00,Ly00b,Ly00a]. This idea appeared first in the context of group theory in [ECH$^+$92].

Definition 1. *A deterministic 2-tape finite automaton (abbreviated 2-d.f.a. in the sequel) is a 5-tuple:*

$$\mathcal{M} = < X, Q, \delta, q_0, F >$$

where

- *X is a finite alphabet, the input-alphabet*
- *Q is a finite set, the set of states*
- *q_0 is a distinguished state , the initial state*
- *$F \subseteq Q$ is a set of distinguished states , the final states*
- *δ, the transition function, is a partial map from $Q \times (X \cup \{\#, \epsilon\})^2$ to Q (where $\#$ is a new letter not in X) fulfilling the restrictions:*
  1. *$\forall q \in Q, \delta(q, \epsilon, \epsilon)$ is undefined,*
  2. *$\forall q \in Q, \forall \boldsymbol{u} \in (X \cup \{\#\})^2, \forall a \in X \cup \{\#\}$, if $\delta(q, \epsilon, a)$ is defined, then*
  $$\delta(q, \boldsymbol{u}) \text{ is defined } \Longrightarrow \boldsymbol{u} = (\epsilon, b), \text{ for some } b \in X \cup \{\#\}$$
  3. *$\forall q \in Q, \forall \boldsymbol{u} \in (X \cup \{\#\})^2, \forall a \in X \cup \{\#\}$, if $\delta(q, a, \epsilon)$ is defined, then*
  $$\delta(q, \boldsymbol{u}) \text{ is defined } \Longrightarrow \boldsymbol{u} = (b, \epsilon), \text{ for some } b \in X \cup \{\#\}.$$

The notation $q \xrightarrow{(u_1,u_2)}_{\mathcal{M}} q'$ means that there is some computation of the automaton $\mathcal{M}$ starting fron state q , reading the input $(u_1, u_2) \in (X \cup \{\#\})^* \times (X \cup \{\#\})^*$ and ending in state q'.

The language recognized by $\mathcal{M}$ is:

$$\mathrm{L}(\mathcal{M}) = \{(u_1, u_2) \in X^* \times X^*, \mid \exists q \in F, q_0 \xrightarrow{(u_1\#,u_2\#)}_{\mathcal{M}} q\}.$$

We are interested in some restrictions on 2-d.f.a.
Let us consider the four situations:
$\exists q \in Q, q' \in F, p_1, p_2, u_1 \neq \epsilon, s_1, s_2 \in X^*$ such that

$$q_0 \xrightarrow{(p_1,p_2)}_{\mathcal{M}} q \xrightarrow{(u_1,\epsilon)}_{\mathcal{M}} q \xrightarrow{(s_1\#,s_2\#)}_{\mathcal{M}} q' \tag{12}$$

$\exists q \in Q, q' \in F, p_1, p_2, u_2 \neq \epsilon, s_1, s_2 \in X^*$ such that

$$q_0 \xrightarrow{(p_1,p_2)}_{\mathcal{M}} q \xrightarrow{(\epsilon,u_2)}_{\mathcal{M}} q \xrightarrow{(s_1\#,s_2\#)}_{\mathcal{M}} q' \tag{13}$$

$\exists q_1, q_1' \in Q, q' \in F, w_1, w_2, p_1, u_1 \neq \epsilon, s_1 \in X^*$ such that

$$q_0 \xrightarrow{(w_1,w_2\#)}_{\mathcal{M}} q_1 \xrightarrow{(p_1,\epsilon)}_{\mathcal{M}} q_1' \xrightarrow{(u_1,\epsilon)}_{\mathcal{M}} q_1' \xrightarrow{(s_1\#,\epsilon)}_{\mathcal{M}} q' \tag{14}$$

$\exists q_2, q_2' \in Q, q' \in F, w_1, w_2, p_2, u_2 \neq \epsilon, s_2 \in X^*$ such that

$$q_0 \xrightarrow{(w_1\#,w_2)}_{\mathcal{M}} q_2 \xrightarrow{(\epsilon,p_2)}_{\mathcal{M}} q_2' \xrightarrow{(\epsilon,u_2)}_{\mathcal{M}} q_2' \xrightarrow{(\epsilon,s_2\#)}_{\mathcal{M}} q' \tag{15}$$

The 2-d.f.a. $\mathcal{M}$ will be said *strictly balanced* iff neither of situations (12,13,14,15) is possible.
The 2-d.f.a. $\mathcal{M}$ will be said *balanced* iff neither of situations (12,13,15) is possible.

Given a deterministic graph $\Gamma = (V, E)$ on an alphabet X we call it *complete* if, for every vertex v and word $u \in X^*$ there exists a path γ in Γ starting from v and labelled by the word u. We denote by $v \odot u$ the unique vertex which is the end of this path γ.

Definition 2. *Let $\Gamma = (V, E, v_1)$ be a 1-graph which is deterministic , complete and such that v_1 is a root. We call* rational structure *on Γ every ($|X|$+2)-tuple of finite automata $\mathcal{S} = (W, M_\epsilon, (M_x)_{x \in X})$ such that*

1. *W is a one-tape deterministic finite automaton on X^* such that* $\forall u \in X^*, \exists w \in \mathrm{L}(W), v_1 \odot u = v_1 \odot w$
2. *M_ϵ is a 2-d.f.a. which is strictly balanced , and* $\mathrm{L}(M_\epsilon) = \{(w_1, w_2) \in \mathrm{L}(W) \times \mathrm{L}(W) \mid v_1 \odot w_1 = v_1 \odot w_2\}$
3. *for every letter $x \in X$, M_x is a 2-d.f.a. which is balanced , and* $\mathrm{L}(M_x) = \{(w_1, w_2) \in \mathrm{L}(W) \times \mathrm{L}(W) \mid v_1 \odot w_1 x = v_1 \odot w_2\}$

Theorem 11. *$\Gamma = (V, E, v_1)$ be a 1-graph which is deterministic and complete. Then Γ has a rational structure $\mathcal{S}$.*
Given a normalized d.p.d.a. such that Γ is the computation 1-graph of $\mathcal{M}$, one can compute such a rational structure $\mathcal{S}$.

4.3 Decision Problems

Let us investigate several decision problems over infinite graphs.
The **isomorphism** problem for **equational** graphs:

INSTANCE: Two equational graphs Γ_1, Γ_2.
QUESTION: Is Γ_1 isomorphic with Γ_2?

This problem has been solved by "model-theoretic" methods in [Cou89,Cou90a] (hence by a method quite different, by nature, from the method used for solving the equivalence problem for dpda in [Sén97c,Sén01,Sti99]). Let us mention that the subproblem where Γ_1, Γ_2 are supposed rooted and deterministic, can be solved as a corollary of the dpda's equivalence problem: it reduces to the property that each graph Γ_i is a homomorphic image of the other and this last problem is solved below by a corollary of the dpda's equivalence problem (theorem 16).

The **isomorphism** problem for **algebraic ordered** trees:

INSTANCE: Two algebraic ordered trees T_1, T_2.
QUESTION: Is T_1 isomorphic with T_2?

Corollary 12. *The isomorphism problem for algebraic ordered trees is decidable.*

This follows from theorem 1 and the reduction given in [Cou83, theorem 5.5.3 p.158].

The **bisimulation** problem for **rooted equational** graphs of **finite out-degree**.

INSTANCE: Two rooted equational graphs of finite out-degree Γ_1, Γ_2.
QUESTION: Is Γ_1 bisimilar to Γ_2?

Theorem 13. *The bisimulation problem for rooted equational graphs of finite out-degree is decidable.*

This kind of problem has been studied for many classes of graphs (or *processes*), see for example [BBK87],[Cau90],[HS91], [CHM93], [GH94], [HJM94],[CHS95], [Cau95],[Sti96],[Jan97],[Sén98a].The case of *rooted equational graphs of finite out-degree* was raised in [Cau95] (see Problem 6.2 of this reference) and is a significant sub case of the problem raised in [Sti96] (as the bisimulation-problem for processes " of type -1"). It is solved in [Sén98a,Sén00a] by a method derived from the one used to solve the equivalence problem for dpda.

The **isomorphism** problem for **algebraic unordered** trees.

INSTANCE: Two algebraic unordered trees T_1, T_2.

QUESTION: Is T_1 isomorphic with T_2?

Theorem 14. *The isomorphism problem for algebraic unordered trees is decidable.*

This result can be proved by a variant of the solution of the bisimulation problem for rooted equational graphs of finite out-degree.
Key idea:
Let us consider two algebraic unordered trees T_1, T_2. Their vertices are labelled on a ranked alphabet (F, ρ). We suppose that these trees have no leaf (one can easily reduce trees in such a normal form, by a transformation which preserves algebraicity and in such a way that isomorphic trees have isomorphic normal forms). Let us consider the two trees T_1', T_2', which are deduced from T_1,T_2 just by removing the labels on the vertices but encoding then the label of a vertex by labels, (we use the alphabet Y described in (11)), on the edges getting out of this vertex. The set of all words labelling a prefix of some branch of T_i' $(1 \le i \le 2)$is just the language $Brch(T_i)$, which is recognized by some dpda $\mathcal{A}_i$. Hence T_i' is the infinite unfolding of the computation 1-graph Γ_i of $\mathcal{A}_i$. Note that Γ_i (for $1 \le i \le 2$), is a rooted deterministic 1-graph over the alphabet Y and it is equational, by theorem 9. An equivalence relation η on Y is defined by:

$$\forall f, f' \in F, i \in [1, \rho(f)], j \in [1, \rho(f')], ([f, i]\eta[f, j] \Leftrightarrow f = f').$$

Definition 15 *Let Γ Γ' be two deterministic n-graphs over the alphabet Y. $\mathcal{R}$ is a η-*permutative bisimulation *from Γ to Γ' iff*

1. $\mathrm{dom}(\mathcal{R}) = V_\Gamma$, $\mathrm{im}(\mathcal{R}) = V_{\Gamma'}$
2. $\forall i \in [1, n], (v_i, v_i') \in \mathcal{R}$,
3. *$\forall (v, v') \in \mathcal{R}$, there exists a bijection $h_{v,v'}$ from $E(v) = \{(v, y, w) \in E_\Gamma\}$ onto $E(v') = \{(v', y', w') \in E_{\Gamma'}\}$ such that, for every $y \in Y, w \in V_\Gamma$: $h_{v,v'}(v, y, w) = (v', y', w') \Rightarrow (y\eta y')$.*

This notion of η-permutative bisimulation is close to the notion of η-bisimulation considered in ([Sén98a, Definition 3.1], [Sén00b, Definition 23]), so that an adaptation of the techniques can be easily done.

The **quotient** problem for **rooted deterministic** equational 1-graphs.

INSTANCE: Two rooted deterministic equational 1-graphs Γ_1, Γ_2.
QUESTION: Does there exist some homomorphism from Γ_1 to Γ_2?

Theorem 16. *The quotient problem for rooted deterministic equational 1-graphs is decidable*

This result can be derived from theorem 11 above and a more general result

Theorem 17. *Let Γ_1, Γ_2 be two rooted deterministic 1-graphs. Let us suppose that Γ_1 is given by an automatic structure $\mathcal{S}$ and that Γ_2 is the computation 1-graph of a given dpda $\mathcal{M}$. One can then decide whether there exists a graph homomorphism $h : \Gamma_1 \to \Gamma_2$.*

Sketch of proof: Let $\mathcal{S} = (W, M_\epsilon, (M_x)_{x \in X})$. For every $a \in X \cup \{\epsilon\}$, let us call a-decomposition, any $2n$-tuple $(u_1, u_1', u_2, u_2', \ldots, u_n, u_n')$ of words such that there exists a computation

$$q_0 \xrightarrow{(u_1,\epsilon)} q_1 \xrightarrow{(\epsilon,u_1')} q_1' \xrightarrow{(u_2,\epsilon)} q_2 \xrightarrow{(\epsilon,u_2')} q_2' \ldots \xrightarrow{(u_n,\epsilon)} q_n \xrightarrow{(\epsilon,u_n')} q_n' \in F \tag{16}$$

and

$$u_1 \cdot u_2 \cdots u_n \in X^*\#; \;\; u_1' \cdot u_2' \cdots u_n' \in X^*\#.$$

Let $\mathcal{M} =< X, Z, Q, \delta_2, q_0, z_0, F >$. We then define, for every $a \in X \cup \{\epsilon\}$, the languages

$$\mathrm{L}_a = \{u_1 \diamond u_1' \diamond u_2 \diamond u_2' \diamond \ldots \diamond u_n \diamond u_n' \diamond \diamond q \;\; {}^t\omega \mid (u_1, u_1', u_2, u_2', \ldots, u_n, u_n') \text{ is some a-decomposition and } q_0 z_0 \xrightarrow{u_1 u_2 \cdots u_n \#^{-1}}_{\mathcal{M}} q\omega \text{ where } q \in Q, \omega \in Z^*\}. \tag{17}$$

$$\mathrm{L}_a' = \{u_1 \diamond u_1' \diamond u_2 \diamond u_2' \diamond \ldots \diamond u_n \diamond u_n' \diamond \diamond q \;\; {}^t\omega \mid (u_1, u_1', u_2, u_2', \ldots, u_n, u_n') \text{ is some a-decomposition and } q_0 z_0 \xrightarrow{u_1' u_2' \cdots u_n' \#^{-1}}_{\mathcal{M}} q\omega \text{ where } q \in Q, \omega \in Z^*\}. \tag{18}$$

(Here $\diamond$ is just a new letter not in $X \cup \{\#\}$).
One can check that there exists some homomorphism $h : \Gamma_1 \to \Gamma_2$ iff

1. $\forall u \in X^*, \exists w \in \mathrm{L}(W)$, such that $q_0 z_0 \odot_{\Gamma_2} u = q_0 z_0 \odot_{\Gamma_2} w$
2. for every $a \in X \cup \{\epsilon\}$, $\mathrm{L}_a = \mathrm{L}_a'$.

Point 1 reduces to test the equality between the two rational languages over the alphabet $Q \cup Z$:

$$\{q\omega \in QZ^* \mid \exists u \in X^*, q_0 z_0 \xrightarrow{u}_{\mathcal{M}} q\omega\}, \{q\omega \in QZ^* \mid \exists w \in \mathrm{L}(W), q_0 z_0 \xrightarrow{w}_{\mathcal{M}} q\omega\}$$

which can be effectively done.
Point 2 amounts to test a finite number of dpda equivalences, which can be effectively done, by theorem 1. □

5 Thue-Systems

5.1 Abstract Rewriting Systems

Let E be some set and $\longrightarrow$ some binary relation over E .We shall call $\longrightarrow$ the *direct reduction*.We shall use the notations $\xrightarrow{i}$ for every integer $(i \geq 0)$, and $\xrightarrow{*}$, $\xrightarrow{+}$ in the usual way (see [Hue80]). By $\longleftrightarrow$, we denote the relation

$\longrightarrow \cup \longleftarrow$. The three relations $\stackrel{*}{\longrightarrow}, \stackrel{*}{\longleftarrow}$ and $\stackrel{*}{\longleftrightarrow}$ are respectively the *reduction*, *derivation* and the *equivalence* generated by $\longrightarrow$.

We use the notions of *confluent* relation and *noetherian* relation in their usual meaning ([Hue80] or [DJ91, §4, p.266-269]).

An element $e \in E$ is said to be *irreducible*(resp. *reducible*) modulo $(\longrightarrow)$ iff there exists no (resp. some) $e' \in E$ such that $e \longrightarrow e'$. By $\mathrm{Irr}(\longrightarrow)$ we denote the set of all the elements of E which are irreducible modulo $(\longrightarrow)$.

Given some subset A of E, we use the following notation:

$$< A >_{\stackrel{*}{\longleftarrow}} = \{e \in E \mid \exists a \in A, a \stackrel{*}{\longleftarrow} e\}, \ \ [A]_{\stackrel{*}{\longleftrightarrow}} = \{e \in E \mid \exists a \in A, a \stackrel{*}{\longleftrightarrow} e\}$$

5.2 Semi-Thue Systems

We call a subset $S \subseteq X^* \times X^*$ a *semi-Thue system* over X. By $\longrightarrow_S$ we denote the binary relation defined by: $\forall f, g \in X^*$,

$$f \longrightarrow_S g \text{ iff there exists } (u, v) \in S, \alpha, \beta \in X^* \text{ such that } f = \alpha u \beta, g = \alpha v \beta.$$

$\longrightarrow_S$ is the one-step reduction generated by S. All the definitions and notation defined in the §"abstract rewriting systems" apply to the binary relation $\longrightarrow_S$. Let us give now additional notions, notation and results which are specific to semi-Thue systems.

We use now the notation $\mathrm{Irr}(S)$ for $\mathrm{Irr}(\longrightarrow_S)$. We define the *size* of S as $\|S\| = \Sigma_{(u,v)\in S}|u| + |v|$. A system S is said *strictly length-reducing* (strict, for short) iff, $\forall (u, v) \in S, |u| > |v|$.

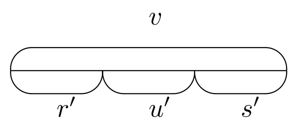
Definition 3. *Let us consider the following conditions on the rules of a semi-Thue system S:*

C1 : for every $(u, v), (u', v') \in S$ *and every* $r', s' \in X^*$
$v = r'u's' \Longrightarrow \mid s' \mid = \mid r' \mid = 0$
C2 : for every $(u, v), (u', v') \in S$ *and every* $r, s' \in X^*$
$rv = u's' \Longrightarrow \mid s' \mid = 0$ *or* $\mid s' \mid \geq \mid v \mid$
C3 : for every $(u, v), (u', v') \in S$ *and every* $r', s \in X^*$
$vs = r'u' \Longrightarrow \mid r' \mid = 0$ *or* $\mid r' \mid \geq \mid v \mid$
S *is said* special *iff* $\forall (u, v) \in S, \mid v \mid = 0$
S *is said* monadic *iff* $\forall (u, v) \in S, \mid v \mid \leq 1$
S *is said* basic *iff it fulfills C1, C2 and C3*
S *is said* left-basic *iff it fulfills C1 and C2.*

Each condition $C_i (i \in [1, 3])$ consists in the prohibition of some superposition configuration for two redexes $(r, u, v), (r', u', v')$ of S.

Condition C1 : C1 expresses the prohibition of the following configuration:
$v = r'u's'$ where $\mid u' \mid < \mid v \mid$

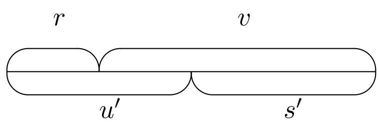
schema 1:



In other words, a righthand side of rule may not strictly embed any lefthand side of rule.

Condition C2 : C2 expresses the prohibition of the following configuration: $rv = u's'$ where $0 <| s' |<| v |$

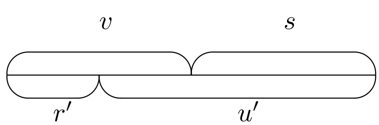
schema 2:



In other words, a righthand side of rule may not be "strictly overlapped on the left" by any lefthand side of rule.

Condition C3 : C3 expresses the prohibition of the following configuration: $vs = r'u'$ where $0 <| r' |<| v |$.

schema 3:



In other words, a right-hand side of rule may not be "strictly overlapped on the right" by any left-hand side of rule.
These definitions appeared in [Niv70,Coc71,But73,Sak79]. We refer the reader to [DJ91,BO93] for more information on rewriting systems and [Sén94] for links with formal language theory.

5.3 Decision Problems

It is well-known that the *confluence* property, for a semi-Thue system S, can be decided, provided that the relation $\longrightarrow_S$ is noetherian. Let us consider the following "weak confluence" property:

$$< f >_{\overset{*}{\longleftarrow}_S} = [f]_{\overset{*}{\longleftrightarrow}_S}. \tag{19}$$

A system S fulfilling (19) for some irreducible word f, is said *confluent over the class of* f. Such a property deserves some interest because,

- the language $< f >_{\overset{*}{\longleftarrow}_S}$ can be recognized in linear time, as soon as S is strict and finite (this motivation remains valid even for some graph rewriting systems, see [ACPS93]).
- in the case where $X^*/ \overset{*}{\longleftrightarrow}_S$ is a group, the confluence property over the class of the empty word, ε, is sufficient to ensure decidability of the word-problem.

Let us call **Class Confluence** Problem (CCP for short), the following decision-problem:

INSTANCE A finite alphabet X, a noetherian semi-Thue system S over X and a word $f \in X^*$, which is irreducible modulo S.
QUESTION Is S confluent on the class $[f]_{\overset{*}{\longleftrightarrow}_S}$?

Several works have been devoted to the decidability/complexity of problems similar to CCP ([Sén85], [ABS87], [Ott87], [Nar90], [Sén90], [OZ91], [Zha91], [MNO91], [Ott92a], [Ott92b], [Zha92], [GG93], [MNOZ93], [Eng94], [Sén98b]). In particular it is known that CCP becomes

- *undecidable* for strict, finite, semi-Thue systems ([Ott92b])
- decidable in *polynomial time* for strict, finite, basic semi-Thue systems ([Sén98b]).

Corollary 18. *The Class Confluence Problem is decidable for strict, finite, left-basic semi-Thue systems.*

This corollary follows from theorem 1 and the reduction given in [Sén90, theorem 5.17]. In the context of the two above complexities obtained for general semi-Thue systems and for basic semi-Thue systems, it is a natural challenge to determine the complexity of the CCP for strict, finite, left-basic semi-Thue systems. It is not even known, at the moment, whether this problem is *primitive recursive* or not (from this point of view, its status is the same as for the dpda's equivalence problem, since the reduction given in [Sén90] is in DEXP-time and a converse reduction, in DEXP-time too, is given).

Acknowledgement

The author thanks the organizers of MCU2001 for their invitation.

References

ABS87. J.M. Autebert, L. Boasson, and G. Sénizergues. Groups and NTS languages. *JCSS vol.35, no2*, pages 243–267, 1987.

ACPS93. S. Arnborg, B. Courcelle, A. Proskurowski, and D. Seese. An algebraic theory of graph reduction. *J. ACM 40*, pages 1134–1164, 1993.

Bau92. M. Bauderon. Infinite hypergraph II, systems of recursive equations. *TCS 103*, pages 165–190, 1992.

BBK87. J. Baeten, J. Bergstra, and J. Klop. Decidability of bisimulation equivalence for processes generating context-free languages. In *Proceedings of PARLE 87*, pages 94–111. LNCS 259, 1987.

BG00. A. Blumensath and E. Gräedel. Automatic structures. In *Proceedings LICS*, pages 105–118. IEEE Computer Society Press, 2000.

BO93. R.V. Book and F. Otto. *String Rewriting Systems.Texts and monographs in Computer Science*. Springer-Verlag, 1993.

But73. P. Butzbach. Une famille de congruences de Thue pour lesquelles l'équivalence est décidable. In *Proceedings 1rst ICALP*, pages 3–12. LNCS, 1973.

Cau90. D. Caucal. Graphes canoniques des graphes algébriques. *RAIRO, Informatique Théorique et Applications, 24(4)*, pages 339–352, 1990.

Cau95. D. Caucal. Bisimulation of context-free grammars and of pushdown automata. In *Modal Logic and process algebra*, pages 85–106. CSLI Lectures Notes, vol. 53, 1995.

CHM93. S. Christensen, Y. Hirshfeld, and F. Moller. Bisimulation is decidable for basic parallel processes. *LNCS 715, Springer*, pages 143–157, 1993.

CHS95. S. Christensen, H. Hüttel, and C. Stirling. Bisimulation equivalence is decidable for all context-free processes. *Information and Computation 121*, pages 143–148, 1995.

Coc71. Y. Cochet. Sur l'algébricité des classes de certaines congruences définies sur le monoide libre. *Thèse de l'université de Rennes*, 1971.

Cou78a. B. Courcelle. A representation of trees by languages,I. *Theoretical Computer Science 6*, pages 255–279, 1978.

Cou78b. B. Courcelle. A representation of trees by languages,II. *Theoretical Computer Science 7*, pages 25–55, 1978.

Cou83. B. Courcelle. Fundamental properties of infinite trees. *Theoretical Computer Science 25*, pages 95–169, 1983.

Cou89. B. Courcelle. The monadic second-order logic of graphs ii: infinite graphs of bounded width. *Math. Systems Theory 21*, pages 187–221, 1989.

Cou90a. B. Courcelle. The monadic second-order logic of graphs iv: definability properties of equational graphs. *Annals of Pure and Applied Logic 49*, pages 193–255, 1990.

Cou90b. B. Courcelle. Recursive applicative program schemes. In *Handbook of Theoretical Computer Science, edited by J. Van Leeuwen*, pages 461–490. Elsevier, 1990.

CS99. A. Carbone and S. Semmes. A graphic apology for symmetry and implicitness. *Preprint*, 1999.

DJ91. N. Dershowitz and J.P. Jouannaud. Rewrite systems. In *Handbook of theoretical computer science, vol.B, Chapter 2*, pages 243–320. Elsevier, 1991.

ECH+92. D.B.A. Epstein, J.W. Cannon, D.F. Holt, S.V.F. Levy, M.S. Paterson, and W.P. Thurston. *Word processing in groups*. Jones and Bartlett, 1992.

Eng94. J. Engelfriet. Deciding the NTS-property of context-free grammars. In *Important Results and Trends in Theoretical Computer Science, (Colloquium in honor of Aarto Salomaa)*, pages 124–130. Springer-Verlag,LNCS nr 812, 1994.

Fri77. E.P. Friedman. Equivalence problems for deterministic context-free languages and monadic recursion schemes. *Journal of Computer and System Sciences 14*, pages 344–359, 1977.

Gal81. J.H. Gallier. Dpda's in 'atomic normal form' and applications to equivalence problems. *Theoretical Computer Science 14*, pages 155–186, 1981.

GG66. S. Ginsburg and S. Greibach. Deterministic context-free languages. *Information and Control*, pages 620–648, 1966.

GG93. P. Grosset-Grange. Décidabilité de la confluence partielle d'un système semi-Thuéien rationnel. *Mémoire de DEA de l'université de Bordeaux 1*, pages 1–21, 1993.

GH94. J. Groote and H. Hüttel. Undecidable equivalences for basic process algebra. *Information and Computation 115*, pages 354–371, 1994.

GL73. S.J. Garland and D.C. Luckham. Program schemes, recursion schemes, and formal languages. *Journal of Computer and System Sciences 7*, pages 119–160, 1973.

Gue81. I. Guessarian. *Algebraic Semantics*. Lecture Notes in Computer Science, vol. 99, 1981.

HJM94. Y. Hirshfeld, M. Jerrum, and F. Moller. A polynomial algorithm for deciding equivalence of normed context-free processes. In *Proceedings of FOCS'94*, pages 623–631. IEEE, 1994.

HS91. H. Hüttel and C. Stirling. Actions speak louder than words: Proving bisimilarity for context-free processes. In *LICS'91*, pages 376–385. IEEE, 1991.

Hue80. G. Huet. Confluent reductions: abstract properties and applications to term rewriting systems. *JACM vol. 27 no 4*, pages 797–821, 1980.

Ian60. I.I. Ianov. The logical schemes of algorithms. *Problems of Cybernetics (USSR) 1*, pages 82–140, 1960.

Jan97. P. Jancar. Bisimulation is decidable for one-counter processes. In *Proceedings ICALP 97*, pages 549–559. Springer Verlag, 1997.

Kap69. D.M. Kaplan. Regular expressions and the equivalence of programs. *Journal of Computer and Systems Sciences 3*, pages 361–386, 1969.

KN95. B. Khoussainov and A. Nerode. Automatic presentations of structures. In *Logic and Computational Complexity, Indianapolis,94*, pages 367–392. L.N.C.S nr 960, 1995.

LPP70. D.C. Luckham, D.M. Park, and M.S. Paterson. On formalised computer programs. *Journal of Computer and Systems Sciences 4*, pages 220–249, 1970.

Ly00a. O. Ly. Automatic graphs and D0L-sequences of finite graphs. *Preprint, submitted to JCSS*, pages 1–33, 2000.

Ly00b. O. Ly. Automatic graphs and graph D0L-systems. In *Proceedings MFCS'2000*, pages 539–548. Springer, LNCS No 1893, 2000.

Mat95. Y.V. Matiyasevich. Personal communication. 1995.

Mil70. R. Milner. Equivalences on program schemes. *Journal of Computer and Systems Sciences 4*, pages 205–219, 1970.

Mil89. R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.

MNO91. K. Madlener, P. Narendran, and F. Otto. A specialized completion procedure for monadic string-rewriting systems presenting groups. In *Proceedings 18th ICALP*, pages 279–290. Springer, LNCS No 510, 1991.

MNOZ93. K. Madlener, P. Narendran, F. Otto, and L. Zhang. On weakly confluent monadic string-rewriting systems. *T.C.S. 113*, pages 119–165, 1993.

Mor00. C. Morvan. Rational graphs. In *Proceedings STACS'2000*. LNCS, 2000.

Nar90. P. Narendran. It is decidable whether a monadic Thue system is canonical over a regular set. *Math. Systems Theory 23*, pages 245–254, 1990.

Niv70. M. Nivat. On some families of languages related to the Dyck language. *2nd Annual Symposium on Theory of Computing*, 1970.

Niv75. M. Nivat. *On the interpretation of recursive polyadic program schemes.* Symposia Mathematica, vol. 15, Academic press, New-York, 1975.

Ott87. F. Otto. On deciding the confluence of a finite string-rewriting system on a given congruence class. *JCSS 35*, pages 285–310, 1987.

Ott92a. F. Otto. Completing a finite special string-rewriting system on the congruence class of the empty word. *Applicable Algebra in Engeneering Comm.Comput. 2*, pages 257–274, 1992.

Ott92b. F. Otto. The problem of deciding confluence on a given congruence class is tractable for finite special string-rewriting systems. *Math. Systems Theory 25*, pages 241–251, 1992.

OZ91. F. Otto and L. Zhang. Decision problems for finite special string-rewriting systems that are confluent on some congruence class. *Acta Informatica 28*, pages 477–510, 1991.

Par81. D. Park. Concurrency and automata on infinite sequences. *LNCS 104*, pages 167–183, 1981.

Pat67. M.S. Paterson. Equivalence problems in a model of computation. *Ph. D. Thesis, University of Cambridge, England*, 1967.

Pel97. L. Pelecq. *Isomorphismes et automorphismes des graphes context-free, équationnels et automatiques.* Thèse de doctorat, Université Bordeaux I, Juin 1997.

Ros75. B.K. Rosen. Program equivalence and context-free grammars. *Journal of Computer and System Sciences 11*, pages 358–374, 1975.

Rut64. J. Rutledge. On Ianov's program schemata. *Journal of the Association for Computing Machinery*, pages 1–9, 1964.

Sak79. J. Sakarovitch. Syntaxe des langages de Chomsky, essai sur le déterminisme. *Thèse de doctorat d'état de l'université Paris VII*, pages 1–175, 1979.

Sén85. G. Sénizergues. The equivalence and inclusion problems for NTS languages. *J. Comput. System Sci. 31(3)*, pages 303–331, 1985.

Sén90. G. Sénizergues. Some decision problems about controlled rewriting systems. *TCS 71*, pages 281–346, 1990.

Sén92. G. Sénizergues. Definability in weak monadic second-order logic of some infinite graphs. In *Dagstuhl seminar on Automata theory: Infinite computations. Wadern, Germany*, volume 28, pages 16–16, 1992.

Sén94. G. Sénizergues. Formal languages and word-rewriting. In *Term Rewriting, Advanced Course*, pages 75–94. Springer, LNCS 909, edited by H. Comon and J.P.Jouannaud, 1994.

Sén97a. G. Sénizergues. L(A) = L(B)? In *Proceedings INFINITY 97*, pages 1–26. Electronic Notes in Theoretical Computer Science 9, URL: http://www.elsevier.nl/locate/entcs/volume9.html, 1997.

Sén97b. G. Sénizergues. L(A) = L(B)? Technical report, LaBRI, Université Bordeaux I, report nr1161-97, 1997. Pages 1-71.

Sén97c. G. Sénizergues. The Equivalence Problem for Deterministic Pushdown Automata is Decidable. In *Proceedings ICALP 97*, pages 671–681. Springer, LNCS 1256, 1997.

Sén98a. G. Sénizergues. Decidability of bisimulation equivalence for equational graphs of finite out-degree. In Rajeev Motwani, editor, *Proceedings FOCS'98*, pages 120–129. IEEE Computer Society Press, 1998.

Sén98b. G. Sénizergues. A polynomial algorithm testing partial confluence of basic semi-Thue systems. *Theoretical Computer Science 192*, pages 55–75, 1998.

Sén00a. G. Sénizergues. Complete Formal Systems for Equivalence Problems. *Theoretical Computer Science*, 231:309–334, 2000.

Sén00b. G. Sénizergues. The bisimulation problem for equational graphs of finite out-degree. *submitted to SIAM Journal on Computing; can be accessed at URL:http://arXiv.org/abs/cs/0008018*, pages 1–98, 2000.

Sén01. G. Sénizergues. L(A) = L(B)? decidability results from complete formal systems. *Theoretical Computer Science*, 251:1–166, 2001.

Sol78. M. Solomon. Type definitions with parameters. In *Conference Record of the Fifth Annual ACM Symposium on Principles of Programming Languages*, pages 31–38, Tucson, Arizona, 1978. ACM SIGACT-SIGPLAN. Extended abstract.

Sti96. C. Stirling. Decidability of bisimulation equivalence for normed pushdown processes. In *Proceedings CONCUR 96*, pages 217–232. Springer-Verlag, LNCS 1119, 1996.

Sti99. C. Stirling. Decidability of dpda's equivalence. Technical report, Edinburgh ECS-LFCS-99-411, 1999. Pages 1-25, accepted for publication in TCS.

Zha91. L. Zhang. Weak confluence is tractable for finite string-rewriting systems. *preprint*, pages 1–10, 1991.

Zha92. L. Zhang. The pre-NTS property is undecidable for context-free grammars. *IPL 44*, pages 181–184, 1992.