Modelization of Deterministic Rational Relations

Serge Grigorieff
LIAFA, Université Paris 7
2, pl. Jussieu 75251 Paris Cedex 05 France
seg@liafa.jussieu.fr

November 23, 2001

Abstract

The definition of the class of deterministic rational relations is fundamentally based on the Read-only One-way Turing machine approach. The notion of deterministic automata developed up to now is too strong and asks for an unnatural detour via end-markers to give all deterministic rational relations (cf. §3.1). We stress that several conditions usually considered as related to determinism are mere normalizations of determinism and are not inherent to the notion (cf. §3.2). In this paper, we introduce pertinent notions of deterministic labelled graph automata (cf. §3.3) which avoid any use of end-markers: strong deterministic, n-deterministic automata for $n \in \mathbf{N}$. These notions form an increasing infinite hierarchy of classes of automata which all lead to the same usual class of deterministic rational relations. Moreover, the class corresponding to the natural extension to the case $n = \infty$ is exactly the class of unambiguous automata .

We also consider Nivat's characterization via multimorphisms applied to rational languages and introduce a hierarchy of deterministic versions of multimorphisms.

Properties of determinism and unambiguity are compared. The decision problems for ambiguity or determinism relative to automata and multimorphisms are settled. Roughly, all problems are undecidable in case of arity ≥ 2 with at least two non binary alphabets, else they are decidable, most being even polynomial time decidable.

Contents

1 Modelization of rational relations over words

1.1 The product monoid $\Sigma_1^* \times \ldots \times \Sigma_k^*$

As usual, Σ^* denotes the set of finite words in a finite alphabet Σ. The empty word is ϵ and $|u|$ is the length of u. We identify u with an application from $\{0, 1, \ldots, |u| - 1\}$ into Σ.
We shall denote words by u, v, w and tuples of words by $\overline{u}, \overline{v}, \overline{w}$.
Let Σ_1, ... ,Σ_k be finite alphabets, $k \geq 2$. The product monoid $\Sigma_1^* \times \ldots \times \Sigma_k^*$, also denoted $\prod \Sigma_i^*$, consists of k-tuples of words with componentwise concatenation product. Its neutral element is $\overline{\epsilon} = (\epsilon, \ldots, \epsilon)$.

Definition 1.1. 1) The length and depth of a multiword $\overline{u} = (u_1, \ldots, u_k)$ are

$$|\overline{u}| = |u_1| + \ldots + |u_m| \quad (1)$$
$$depth(\overline{u}) = \sup(|u_1|, \ldots, |u_m|) \quad (2)$$

2) The support of a k-tuple $\overline{u}$ is $Supp(\overline{u}) = \{i \in \{1, \ldots, k\} \mid u_i \neq \epsilon\}$.

Remark 1.2. The set of multiwords with length 1 (i.e. multiwords with all components equal to ϵ except one which is a letter) generates the monoid $\prod \Sigma_i^*$. In fact, it is the smallest (relative to set inclusion) set of generators.

The set-theoretical definition of rational relations included in $\prod \Sigma_i^*$ and the associated machine-like models (namely automata) are mere particular cases of notions defined in the general context of monoids. We recall these notions in the next subsections and refer to the litterature for proofs of Theorems (cf. Eilenberg, 1974, [4], Berstel, 1979, [1], Sakarovitch, 2001, [28] II.1.3 Thm 1.1).

1.2 Rational relations defined via set theoretical operations

Definition 1.3. 1) The product operation on elements of a monoid M (with neutral element 1_M) induces operations on subsets of M. If R and S are subsets of M then the concatenation product of R, S and the plus and star of R are the relations

$$RS = \{xy \mid x \in R \text{ and } y \in S\}$$
$$R^+ = \bigcup_{n \geq 1} R^n \qquad R^* = \{1_M\} \cup R^+$$

2) The family $Rat(M)$ of rational subsets of M is the closure of the family of finite relations by the operations of union, product and star.

1.3 Multitape automata

There are several machine counterparts for $Rat(M)$. Though the historical machine is the Read-only One-way multitape Turing Machine (cf.§1.7), the reference machine model is the labelled graph automaton introduced by John Myhill, 1957 [18].

Definition 1.4. 1) A finite automaton $\mathcal{A}$ over a monoid M (with neutral element 1_M) is a finite directed graph $\mathcal{A} = \langle Q, M, \delta, I, F \rangle$ labelled by elements of $M \setminus \{1_M\}$. Nodes and edges form the respective sets Q and $\delta \subseteq Q \times M \times Q$ and are also called states and transitions. I and F are distinguished subsets of Q called initial states and final states.

2) An $\mathcal{A}$-path c is a finite sequence of transitions

$$p_0 \xrightarrow{a_1} p_1 \xrightarrow{a_2} p_2 \ldots p_{n-1} \xrightarrow{a_n} p_n$$

The origin and end of the path c are p_0 and p_n. The label of c is the element $a_1 a_2 \ldots a_n$ of the monoid M. Case $n = 0$ corresponds to an empty path, with label 1_M and p_0 as both origin and end. Path c is successful if its origin is in I and its end is in F.

3) The behaviour of $\mathcal{A}$ is the set of labels of successful paths.

4) States which belong to some path with origin in I (resp. last state in F) are called accessible (resp. coaccessible). Automaton $\mathcal{A}$ is trim if all states are accessible and coaccessible.

Notation 1.5. 1) In case M is a free monoid Σ^* (with Σ a finite alphabet), the behaviour of $\mathcal{A}$ is also called the *language* associated to $\mathcal{A}$ and denoted $L(\mathcal{A})$.
2) In case M is a product of free monoids $\Sigma_1^* \times \ldots \times \Sigma_k^*$ (where all Σ_i's are finite alphabets), $\mathcal{A}$ is called a finite multitape (or k-tape) automaton and the behaviour of $\mathcal{A}$ is also called the *relation* associated to $\mathcal{A}$ and denoted $Rel(\mathcal{A})$.

The basic case $M = \Sigma^*$ of the following fundamental theorem is due to Kleene, 1956 [15]. The general statement for arbitrary monoids is due to Elgot & Mezei, 1965 [6], who detailed the case $\prod \Sigma_i^*$ in [6] Prop.3.5, and mentioned the general result in a footnote p.50.

Theorem 1.6 (Kleene's theorem). *A subset of M is rational if and only if it is the behaviour of some finite automaton (resp. trim automaton).*

1.4 Multiplicities

As in the case of languages, the set-theoretical operations defining $Rat(\prod \Sigma_i^*)$ can be augmented with a notion of multiplicity so as to allow deep algebraic considerations. This is done by replacing relations (which may be viewed as functions $R : \prod \Sigma_i^* \to \{0,1\}$) by $\mathbf{N}$-series over $\prod \Sigma_i^*$, i.e. *multiplicity functions* $s : \prod \Sigma_i^* \to \mathbf{N}$. The three set operations of union, product and star on relations respectively correspond to the following algebraic operations on series: *sum* $s + t$, *Cauchy product* st and *Cauchy star* s^*, where

$$st(\overline{u}) = \sum \{s(\overline{v})t(\overline{w}) \ : \ \overline{vw} = \overline{u}\} \tag{3}$$

$$s^*(\overline{u}) = \sum \{s^n(\overline{u}) \ : \ n \in \mathbf{N}\} \quad \text{defined if } s(\overline{\epsilon}) = 0 \tag{4}$$

with the convention that s^0 is the neutral element for Cauchy product, i.e. such that $s^0(\overline{\epsilon}) = 1$ and $s^0(\overline{u}) = 0$ for $\overline{u} \neq \overline{\epsilon}$.
That there are only finitely many non zero terms in the above sums is insured by easy considerations on the lengths of k-tuples of words.

Definition 1.7. The family of rational series over $\prod \Sigma_i^*$ is the closure of the family of finite relations by the operations of sum, Cauchy product and Cauchy star.

Definition 1.8. Let $\mathcal{A}$ be a finite automaton over $\prod \Sigma_i^*$. We allow several edges with the same label between any two states (in other words, edges have multiplicities). The $\mathcal{A}$-multiplicity series $\mathcal{A}$-mult : $\prod \Sigma_i^* \to \mathbf{N}$ associates to any multiword $\overline{u}$ the number of accepting paths with label $\overline{u}$ (this number is finite due to the fact that no edge is labelled $\overline{\epsilon}$).

The counterpart to Kleene's theorem 1.6 is due to Schützenberger, 1961 [30] for the basic case Σ^*. The extension to $\prod \Sigma_i^*$ is easy and folklore.

Theorem 1.9 (Kleene-Schützenberger's theorem). *Rational series are exactly the multiplicity series of finite automata (resp. trim and normal automata).*

Remark 1.10. In particular, a relation $R \subseteq \prod \Sigma_i^*$ is rational if and only if it is the support $supp(s)$ of some rational series s, where

$$supp(s) = \{\overline{u} \ : \ s(\overline{u}) \neq 0\}$$

1.5 Normalizing automata

For future reference (cf. proof of Prop.1.21, Remarks 3.18,3.19) we make some simple observations about normalization of automata over $\prod \Sigma_i^*$.

Definition 1.11. Let $\mathcal{A}$ be an automaton over $\prod \Sigma_i^*$.

1) $\mathcal{A}$ is quasi-normal if all labels of transitions have depth 1 (cf. Def.1.1) (i.e. if $\overline{u} = (u1, \ldots, u_k)$ is a label then $\sup\{|u_1|, \ldots, |u_k|\} = 1$, which means that each u_i is a letter or is ϵ and that some u_i is a letter).

2) $\mathcal{A}$ is normal if all labels of transitions have length 1 (cf. Def.1.1). (i.e. if $\overline{u} = (u1, \ldots, u_k)$ is a label then $|u_1| + \ldots + |u_k| = 1$, which means that among the u_i's exactly one is a letter and the other ones are ϵ).

The idea to normalize an automaton is to split a transition $p \xrightarrow{\overline{\alpha}} q$ into a sequence of transitions

$$p \xrightarrow{\overline{\beta^1}} (p, \overline{\beta^1}) \xrightarrow{\overline{\beta^2}} (p, \overline{\beta^1\beta^2}) \ldots \xrightarrow{\overline{\beta^{m-1}}} (p, \overline{\beta^1} \ldots \overline{\beta^{m-1}}) \xrightarrow{\overline{\beta^m}} q \qquad (5)$$

following a decomposition $\overline{\alpha} = \overline{\beta^1} \dots \overline{\beta^m}$.

Notation 1.12. If u is a word and $\overline{u} = (u_1, \dots, u_k)$ a multiword, we let

$$\begin{aligned} letter_t(u) &= \text{IF } 1 \leq j \leq |u| \text{ THEN the } t\text{-th letter of } u \text{ ELSE } \epsilon \\ letter_t(\overline{u}) &= (letter_t(u_1), \dots, letter_t(u_k)) \\ letter_t^i(\overline{u}) &= (\epsilon, \dots, \epsilon, letter_t(u_i), \epsilon, \dots, \epsilon) \end{aligned}$$

We consider two decompositions of $\overline{u}$, where $m = depth(\overline{u})$:

$$\overline{u} = letter_1(\overline{u}) \dots letter_m(\overline{u}) \quad (6)$$

$$\overline{u} = letter_1^1(u_1) \dots letter_{|u_1|}^1(u_1) \dots letter_1^k(u_k) \dots letter_{|u_k|}^k(u_k) \quad (7)$$

Definition 1.13. 1) We let $QuasiNormal(\mathcal{A})$ be the automaton obtained from $\mathcal{A}$ by splitting any $\mathcal{A}$-transition $p \xrightarrow{\overline{\alpha}} q$ as shown in equation (5) above following the decomposition of $\overline{\alpha}$ given by equation (6) above.
This introduces new states

$$(t, letter_1(\overline{\alpha}) \dots letter_t(\overline{\alpha}))$$

for $1 \leq t < depth(\overline{\alpha})$. Notice that some of these states may come from several transitions with origin p.

2) We let $Normal(\mathcal{A})$ be the automaton obtained from $\mathcal{A}$ by splitting any $\mathcal{A}$-transition as shown in equation (5) above following the decomposition of $\overline{\alpha}$ given by equation (7) above.
This introduces new states

$$(p, (\alpha_1, \dots, \alpha_j, \epsilon, \dots, \epsilon))$$

$$(p, (\alpha_1, \dots, \alpha_j, letter_1(\alpha_{j+1}) \dots letter_t(\alpha_{j+1}), \epsilon, \dots, \epsilon))$$

for $1 \leq j < k$ and $1 \leq t < |\alpha_{j+1}|$. Notice that some of these states may come from several transitions with origin p.

Proposition 1.14. *$Normal(\mathcal{A})$ (resp. $QuasiNormal(\mathcal{A})$) is a normal (resp. quasi-normal) automaton which has the same behaviour and multiplicity series (cf. §1.4) as $\mathcal{A}$.*

1.6 Multimorphisms and Nivat's theorem

Rational relations over $\prod \Sigma_i^*$ can be given still another characterization using usual rational languages and morphisms from some monoid Γ^* to the monoid $\prod \Sigma_i^*$. This is due to M. Nivat, 1968 [19]. Observe that a morphism $\Phi : \Gamma^* \to \prod \Sigma_i^*$ is a tuple of morphisms $(\varphi_i : \Gamma^* \to \Sigma_i^*)_{i=1,\dots,k}$. Whence the name multimorphism.

Definition 1.15. Let $\varphi : \Gamma^* \to \Sigma^*$ be a morphism and $\Phi = (\varphi_i : \Gamma^* \to \Sigma_i^*)_{i=1,\dots,k}$ be a multimorphism.

1) φ is alphabetical (resp. strict alphabetical) if $|\varphi(a)| \leq 1$ (resp. $|\varphi(a)| = 1$) for all $a \in \Gamma$. Φ is alphabetical (resp. strict alphabetical) if so are all φ_i's.

2) φ (resp. Φ) is proper if $\varphi(a) \neq \epsilon$ (resp. $\Phi(a) \neq \overline{\epsilon}$) for all $a \in \Gamma$.

Theorem 1.16 (Nivat's theorem [19]). *A relation $R \subseteq \prod \Sigma_i^*$ is rational if and only if $R = \Phi(L)$ where $L \subseteq \Gamma^*$ is rational and $\Phi : \Gamma^* \to \prod \Sigma_i^*$ is a multimorphism (resp. strict alphabetical multimorphism).*

1.7 Read-only One-way Turing Machines

The following material will be needed for the modelization of deterministic rational relations in §3.3.
As said in §1.3, the reference model for multitape automaton is the labelled graph model of Definition 1.4. However, as pointed by Fischer & Rosenberg, 1968 [7] p.90–91, the basic intuition for multitape automaton remains that of non deterministic *Read-only One-way multitape Turing machine* (*ROTM*) with exactly one head per tape.

Definition 1.17. An *ROTM* is a non deterministic Turing machine such that the symbols on the tapes are not modified and there is no backward move and every transition moves at least one head. Thus,

i) the machine starts in any initial state,

ii) a transition step depends (non deterministically) on the state of the machine and on the letters (or the blank symbol B lying on the right of the inputs) read by the k heads on the k tapes,

iii) a transition step changes state and moves *forward* some heads (possibly none),

iv) the machine stops when it enters a halting state,

v) halting states are of two types: accepting or rejecting,

vi) a computation is accepting if it enters an accepting halting state.

It is convenient to introduce a variant of the *ROTM* which we shall call *modified ROTM*'s.

Definition 1.18. A *modified ROTM* behaves according to *i*, *ii* of Def.1.17 and to

iiibis) a transition step changes state and moves *forward* at least one head (possibly several heads),

ivbis) a head which reads the blank symbol B does not move and the

machine stops when all heads read the blank symbol B,

vbis) states are of two types: final and non final,

vibis) a computation is accepting if its last state is final.

Formally, a modified $ROTM$ is a tuple

$$\mathcal{T} = \langle Q_{\mathcal{T}}, \Sigma_1, \ldots, \Sigma_k, \delta_{\mathcal{T}}, I_{\mathcal{T}}, F_{\mathcal{T}} \rangle$$

with

$$\delta_{\mathcal{T}} \subseteq Q \times \prod(\Sigma_i \cup \{B\}) \times Q \times \{0,1\}^k$$

A tuple $(q, \overline{a}, r, \overline{m})$ in $\delta_{\mathcal{T}}$ is interpreted as follows: q, r are the states before and after the transition, the k heads read $\overline{a}$ and move according to $\overline{m}$.

The following is folklore.

Proposition 1.19. *A relation is computable by an $ROTM$ (resp. deterministic, resp. unambiguous $ROTM$) if and only if it is computable by a modified $ROTM$ (resp. deterministic, resp. unambiguous modified $ROTM$).*

Proof. 1) *$ROTM \Rightarrow$ modified $ROTM$.*

Get rid of $\overline{\epsilon}$-transitions (i.e. transitions which move no head) in the standard way:

α) First, iteratively add new transitions as follows: if $(p, \overline{a}, q, \overline{0}), (q, \overline{a}, r, \overline{m})$ are transitions then add transition $(p, \overline{a}, r, \overline{m})$,

β) Then suppress all $\overline{\epsilon}$-transitions.

Now, force the machine to read entirely its input: if q is a halting state then add all transitions $(q, \overline{a}, q, \overline{m})$ where

$$m_i = \text{ IF } a_i = B \text{ THEN 0 ELSE 1}$$

Declare as final states all accepting halting states and all states from which one can access an accepting halting state by a sequence of transitions reading $(B, \ldots, B)$.

It is easy to check that these modifications transform an $ROTM$ into a modified $ROTM$ computing the same relation.

Moreover, this transformation preserves determinism and unambiguity.

But it can not preserve multiplicities:

- all multiplicities are necessarily finite with modified $ROTM$'s,

- infinite multiplicities are possible with $ROTM$'s since $\overline{\epsilon}$-transitions and transitions reading $(B, \ldots, B)$ can create loops.

2) *modified $ROTM \Rightarrow ROTM$.*

To every state p associate a new state p^{halt} and add transitions

$$(p, (B, \ldots, B), p^{halt}, \overline{0})$$

Declare as accepting (resp. rejecting) halting states those states p^{halt} such that p is final (resp. non final). □

It is also well-known that the $ROTM$ and modified $ROTM$ models are equivalent to the labelled graph automaton model (Fisher & Rosenberg, 1968 [7]).

Remark 1.20. For an $ROTM$ (as for any modified $ROTM$ or Turing machine), an occurrence of a symbol in an input on some tape is read again and again through successive transitions while there is no move of the head of that tape. *The labelled graph automaton model can thus be viewed as a very smooth normalization of the ROTM model in which each occurrence of a symbol of each input is read only once.*

Proposition 1.21. *A relation is rational if and only if it is computed by a modified ROTM. This is also true when multiplicities are considered.*

Proof. 1) *automaton* $\Rightarrow$ *modified ROTM*
Going from an automaton $\mathcal{A}$ to a modified $ROTM$ is quite easy and can be done as follows:
i) Quasi-normalize $\mathcal{A}$ as in Definition 1.13,
ii) A quasi-normal automaton can directly be interpreted as a modified $ROTM$.

2) *modified ROTM* $\Rightarrow$ *automaton*
The passage from a modified $ROTM$ to an automaton requires some care. As we shall need it for the proof of Thm.3.17, we detail the construction in Def.1.22. It is easy to check that this construction leads to a quasi-normal automaton $\mathcal{A}$ which computes the same relation as $\mathcal{T}$ and *also the same multiplicity series.*

Definition 1.22. Let $\mathcal{T} = \langle Q_\mathcal{T}, \Sigma_1, \ldots, \Sigma_k, \delta_\mathcal{T}, I_\mathcal{T}, F_\mathcal{T} \rangle$ be a modified $ROTM$. We define an automaton $\mathcal{A} = \langle Q_\mathcal{A}, \prod \Sigma_i^*, \delta_\mathcal{A}, I_\mathcal{A}, F_\mathcal{A} \rangle$ as follows:
i) $Q_\mathcal{A} = Q_\mathcal{T} \times \prod(\Sigma_i \cup \{B, new\})$
ii) $I_\mathcal{A} = I \times \prod\{new\}$, $F_\mathcal{A} = F_\mathcal{T} \times \prod\{B, new\}$
iii) $\delta_\mathcal{A} = \bigcup_{t \in \delta_\mathcal{T}} Trans(t)$
where if $t = (q, \overline{a}, r, \overline{m}) \in \delta_\mathcal{T}$ then $Trans(t)$ is the following family of transitions of $\mathcal{A}$:

$$\begin{aligned}\{((q,\overline{\xi}),\overline{\alpha},(r,\overline{\eta})) \quad : \quad & \text{for some } i = 1,\ldots,k \;\; \xi_i = new \\ & \text{and for all } i = 1,\ldots,k \\ & \xi_i \in \{a_i, new\} \text{ and } (a_i = B \Rightarrow \xi_i = B) \text{ and} \\ & \alpha_i = \text{ IF } (\xi_i = new \text{ and } a_i \neq B) \text{ THEN } a_i \text{ ELSE } \epsilon \\ & \text{and } \eta_i = \text{ IF } m_i = 1 \text{ THEN } new \text{ ELSE } a_i\}\end{aligned}$$

Thus, $Trans(t)$ contains $2^l - 1$ transitions where l is the number of i's such that $a_i \neq B$. Observe that this number is positive since $\mathcal{T}$ halts when it reads $(B, \ldots, B)$ (so that $l \geq 1$).
Intuition: A letter which has just been read in a $\mathcal{T}$-computation is retained in the state of the emulating $\mathcal{A}$-computation. This allows $\mathcal{A}$ to read each letter only once and not several times as $\mathcal{T}$ possibly does. Such an emulation of $\mathcal{T}$ by $\mathcal{A}$ cannot be static (state to state and transition to transition), it has to be somewhat dynamic. This is why to a single $\mathcal{T}$-transition t we associate a family $Trans(t)$ of $\mathcal{A}$-transitions which may emulate t. The i-th component of the label of an $\mathcal{A}$ transition is non empty if and only if this transition emulates a $\mathcal{T}$ transition in which the symbol on tape i is read for the first time and is not the blank end-marker (i.e., either this is the first transition or in the preceding transition, there has been a move on tape i). Thus, in a state $(q, \overline{\xi})$ of $\mathcal{A}$ we have $\xi_i = new$ if this state is initial or if it emulates a $\mathcal{T}$-state obtained after a $\mathcal{T}$-transition which makes a move on tape i. Else, ξ_i is the symbol which was read and will again be read by $\mathcal{T}$.

Remark 1.23. If the modified $ROTM$ $\mathcal{T}$ moves exactly one head per transition then all labels of $\mathcal{A}$-transitions have length 1 and $\mathcal{A}$ is a normal automaton.

1.8 In-between automata and $ROTM$'s: $ROTM$ transducers

The notion of (k,h)-$ROTM$ transducer is obtained by adding to the k input tapes of an $ROTM$ a family of h output tapes with alphabets $\Delta_1, \ldots, \Delta_h$:

$$\mathcal{T} = \langle Q_\mathcal{T}, \Sigma_1, \ldots, \Sigma_k, \delta_\mathcal{T}, I_\mathcal{T}, F_\mathcal{T}, \Delta_1, \ldots, \Delta_h, \lambda_\mathcal{T} \rangle$$

where $\lambda_\mathcal{T} \subseteq Q \times ((\Sigma_1 \cup \{B\}) \times \ldots \times (\Sigma_k \cup \{B\})) \times (\Delta_1 \times \ldots \times \Delta_h)$.
Now, k-tape $ROTM$'s can be viewed as $(k,0)$-$ROTM$ transducers while labelled graph k-tape automata can be viewed as $(0,k)$-$ROTM$ transducers. From this point of view, *automata appear as pure output machines with no input.*

1.9 Tally rational relations

Tally relations are relations over unary alphabets. Via an obvious bijective map, they correspond to relations over $\mathbf{N}$. A characterization as lattices in $\mathbf{N}^k$ was obtained by R.J. Parikh, 1961 [21], reprinted in Parikh, 1966 [22] (see also Goldstine, 1977 [10]). Another characterization via Presburger arithmetic was developed by Ginsburg & Spanier, 1966 [9].

Definition 1.24. A relation $R \subseteq \mathbf{N}^k$ is linear if there is a finite sequence $\overline{u_0}, \ldots, \overline{u_n}$ of elements of $\mathbf{N}^k$ such that

$$\begin{aligned} R &= \overline{u_0} + \overline{u_1}\mathbf{N} + \ldots + \overline{u_n}\mathbf{N} \\ &= \{\overline{u_0} + x_1\overline{u_1} + \ldots + x_n\overline{u_n} \ : \ x_1, \ldots, x_n \in \mathbf{N}\} \end{aligned} \tag{8}$$

A relation is semi-linear if it is the union of finitely many linear relations.

Theorem 1.25. *1)* [21] *Tally rational relations are exactly the semilinear relations and* [9] *are closed under boolean operations and projections.*

2) [9] *Let R be a tally relation. The following conditions are equivalent:*
i) R is rational
ii) R is definable in the structure $\langle \mathbf{N}, =, + \rangle$ (i.e. Presburger arithmetic)
iii) R is definable in the structure $\langle \mathbf{N}, =, + \rangle$ by a Σ_1^0 formula

Since Presburger arithmetic is decidable ([25]), this gives a tool to get decidability results about tally rational relations. For instance,

Corollary 1.26. *There is an algorithm to decide whether two tally rational relations are disjoint.*

A simple argument allows to extend this last result to the case of relations with at most one non unary component.

Corollary 1.27. *There is an algorithm to decide whether two rational relations with at most one non unary component are disjoint.*

Proof. Let R_1, R_2 be rational relations included in $\Sigma^* \times (\{0\}^*)^k$ and let

$$R = \{(\overline{v}, \overline{w}) \in (\{0\}^*)^{2k} \ : \ \exists u \in \Sigma^* \ ((u, \overline{v}) \in R_1 \ \wedge (u, \overline{w}) \in R_2)\}$$

be the composition of R_1, R_2 along their Σ^* component. Clearly, R_1, R_2 are disjoint if and only if R and the diagonal relation

$$Diag_k = \{(\overline{v}, \overline{v}) \in (\{0\}^*)^{2k} \ : \overline{v}) \in (\{0\}^*)^k\}$$

are disjoint. Now, R and $Diag_k$ are tally rational relations and automata to compute these relations can be polynomial time defined from automata for R_1, R_2. This gives a polynomial time reduction of the disjointness problem for rational $(k+1)$-ary relations with at most one non unary component to the disjointness problem for tally $2k$-ary rational relations. □

Remark 1.28. 1) The above result is also an easy application of the analog result (due to Ibarra, 1978, [13], Theorem 3.1 p. 124) about finite-turn multicounter machines, i.e. one-tape automata (over some non necessarily unary alphabet) with k counters making at most r alternations between push and pop modes for some fixed r (all counters being initially empty).

Cf. the proof of Corollary 1.31.
2) In case at least two components are non unary, the disjointness problem is undecidable (Rabin & Scott, 1959 [26], cf. also the proof of Theorem 2.4 below).

Using equation 8, the disjointness problem for k-ary tally rational relations can be related to *linear programming with non negative integers* involving systems of k linear equations. Such systems with a fixed number of equations are polynomial time solvable (H.W. Lenstra, 1983 [16], cf. also A. Schrijver's book [29], Cor.18.7 p.260). Unfortunately, the passage from the automaton to a semi-linear representation is a priori exponentially complex. Thus, it is not possible to directly apply Lenstra's theorem in order to get a polynomial time algorithm in Corollary 1.26. Nevertheless, as proved by Gurari & Ibarra, 1981, [11], this can be achieved via an argument which uses known sharp bounds to integral solutions of linear systems (Borosh & Treybig, 1976, [3], von zur Gathen & Sieveking, 1978, [31]).

As in Ibarra [13], the result proved by Gurari & Ibarra, 1981, [11] (Corollary 1 p. 224), deals with finite-turn multicounter machines (cf. Remark 1.28).

Theorem 1.29 (Gurari & Ibarra, 1981, [11]). *For fixed k, there is a polynomial time algorithm to decide whether two finite-turn multicounter machines compute disjoint languages.*

Remark 1.30. However, the degree of the polynomial bound increases with k.

A simple polynomial time reduction leads to the following improvement of Corollary 1.27

Corollary 1.31. *For fixed k, there is a polynomial time algorithm to decide whether two $(k+1)$-tapes automata over k unary alphabets and one possibly non unary alphabet compute disjoint rational relations.*

Proof. A $(k+1)$-tape automaton $\mathcal{A}$ over k unary alphabets and one possibly non unary alphabet can be emulated by a finite-turn $2k$-counter machine as follows:
i) Code the $k+1$ inputs $0^{n_1}, 0^{n_2}, \ldots, 0^{n_k}, u$ as a binary input $0^{n_1}10^{n_2}1 \ldots 10^{n_k}1u$.
ii) consider the 1-turn k-counter machine $\mathcal{M}$ which first pushes the first k blocks of zeros $0^{n_1}, 0^{n_2}, \ldots, 0^{n_k}$ of its input into the k counters of $\mathcal{M}$, then emulates $\mathcal{A}$ on inputs $0^{n_1}, 0^{n_2}, \ldots, 0^{n_k}, u$ so that a move on the i-th tape of $\mathcal{A}$ becomes a pop on the i-th counter of $\mathcal{M}$.

This emulation is clearly polynomial time computable and gives a polynomial time reduction of the associated disjointness problems. □.

2 Modelization of unambiguous rational relations

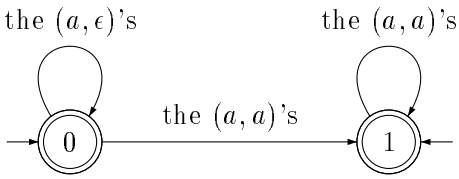
This section reviews known notions and facts and proves new polynomial time decidability results for diverse problems about ambiguity in case there is at most one non unary alphabet (cf. Cor.2.6). It also stresses analogies with the material we are going to develop for the modelization of determinism in §3.

2.1 Unambiguous automata and relations

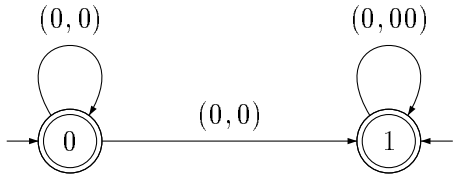
Definition 2.1. 1) An automaton is unambiguous if for every input there exists at most one accepting run (but there can be many non accepting runs), i.e. the associated multiplicity series (cf. Def.1.8) is $\{0,1\}$-valued.

2) A rational relation is unambiguous if it is accepted by some unambiguous automaton.

Example 2.2. Two unambiguous (but non deterministic) relations:
1) The suffix relation for words on a non unary alphabet Σ.



2) $R = \{(0^m, 0^n) \ : \ m \leq n \leq 2m\}$. To get an unambiguous automaton for R, observe that every pair in R can be written in a unique way in the form $(0^{p+q}, 0^{p+2q})$.



Rational relations which cannot be accepted by unambiguous automata are called inherently ambiguous. The standard example is analog to the standard inherently ambiguous context-free language:

$$\{(0^m 1^n, 0^p) \ : \ p = m \text{ or } p = n\}$$

However, Eilenberg and Schützenberger, 1969 [5] proved the following theorem.

Theorem 2.3 ([5]). *Every tally rational relation is unambiguous.*

2.2 Decidability of ambiguity for automata

Ambiguity does not correspond to a simple property of machines. In the general case, this is an undecidable property.

Theorem 2.4 (Rabin & Scott, 1959 [26]). *Let $k \geq 2$ and suppose that at least two among the k alphabets $\Sigma_1, \ldots, \Sigma_k$ are non unary. Then the class of unambiguous automata is not recursive. It is, in fact, Π_1^0-complete.*

Proof. Rabin & Scott [26] really proved that the disjointness problem is undecidable, but their proof applies with no change to the ambiguity problem. We briefly recall their argument. The *Post Correspondence Problem* for m sequences (PCP_m) is the class of pairs of homomorphisms $\varphi, \psi : \{1, \ldots, m\}^* \to \{0,1\}^*$ such that there exists $u \neq \epsilon$ satisfying $\varphi(u) = \psi(u)$. Now, let $\mathcal{A}_\varphi = \langle \{q_0, q_1\}, \delta_\varphi, \{q_0\}, \{q_1\} \rangle$ where

$$\delta = \{(q, i, \varphi(i), q_1) \ : \ q \in \{q_0, q_1\} \ , \ i \in \{1, \ldots, m\}$$

be the obvious 2-tape automaton which computes the relation

$$R_\varphi = \{(u, \varphi(u)) \ : \ u \in \{1, \ldots, m\}^* \ , \ u \neq \epsilon\}.$$

Observe that $\mathcal{A}_\varphi$ is unambiguous (even super-deterministic automata, cf. Def.3.1). It is clear that the union automaton $\mathcal{A}_\varphi \cup \mathcal{A}_\psi$ is ambiguous if and only if $(\varphi, \psi) \in PCP_m$. This gives a recursive reduction (in fact polynomial time reduction) of PCP_m into the class of ambiguous automata on alphabets $\{1, \ldots, m\}, \{0,1\}$.
It is known that PCP_m is not recursive and is Σ_1^0-complete for large m (Post, 1946 [24]), even for $m = 7$ (Matiyasevich & Senizergues, 1996 [17]). This proves the theorem for the case of alphabets $\{1, \ldots, m\}, \{0,1\}$. Standard coding transfers the result to any finite sequence of alphabets with at least two non unary alphabets. □

However, when at most one alphabet is non unary, unambiguous automata form a recursive class.
First, recall the notion of truth-table reducibility (cf. standard textbooks on recursion theory, [27] p.109–110, [20] p.268). A class $X \subseteq \Sigma^*$ is *truth-table reducible* to $Y \subseteq \Delta^*$ if there exists a recursive function $\Phi : \Sigma^* \to P_{fin}(\Delta^*)$ (where $P_{fin}(\Delta^*)$ is the set of finite subsets of Δ^*) such that

$$\forall u \ (u \in X \ \Leftrightarrow \Phi(u) \subseteq Y)$$

X is *polynomially truth-table reducible* to Y if the above Φ is polynomial time computable (in particular, the number of words in $\Phi(u)$ and their lengths are bounded by a polynomial in $|u|$).

Proposition 2.5. *Fix alphabets $\Sigma_1, \ldots, \Sigma_k$. The class of unambiguous automata is polynomially truth-table reducible to the class of pairs of automata computing disjoint relations.*

Proof. Suppose that $\mathcal{A}$ is a trim automaton (cf. Def.1.4, point 4). Then $\mathcal{A}$ is ambiguous if and only if there exists two distinct paths having the same label, starting in the same state but having different first transitions.
Let's denote by $\mathcal{A}_q$ the automaton $\mathcal{A}$ with q as (unique) initial state. If $(p, \overline{\alpha}, q)$ is a transition of $\mathcal{A}$, let's denote $\mathcal{A}_{(p,\overline{\alpha},q)}$ the automaton obtained from $\mathcal{A}$ by adding a new state p^{bis} as the (unique) initial state and a new transition $(p^{bis}, \overline{\alpha}, q)$. It is clear that $Rel(\mathcal{A}_{(p,\overline{\alpha},q)}) = \overline{\alpha}\ Rel(\mathcal{A}_q)$.
Using these notations, $\mathcal{A}$ is ambiguous if and only if there exists two distinct transitions $(p, \overline{\alpha}, q), (p, \overline{\beta}, r)$ with the same origin such that $\overline{\alpha}\ Rel(\mathcal{A}_q) \cap \overline{\beta}\ Rel(\mathcal{A}_r) \neq \emptyset$.
Thus, $\mathcal{A}$ is unambiguous if and only if $Rel(\mathcal{A}_{(p,\overline{\alpha},q)}) \cap Rel(\mathcal{A}_{(p,\overline{\beta},r)}) = \emptyset$ for all pairs $(p, \overline{\alpha}, q), (p, \overline{\beta}, r)$ of distinct transitions with the same origin.
This gives a polynomial time truth-table reduction of the class of unambiguous automata to the class of pairs of automata computing disjoint relations. □

From Prop.2.5 and Cor.1.31 we get

Corollary 2.6. *If there is at most one non unary alphabet then the class of unambiguous automata is polynomial time computable.*

2.3 Unambiguous multimorphisms

The analog of Nivat's theorem 1.16 relative to unambiguous relations holds.

Theorem 2.7. *A relation $R \subseteq \prod \Sigma_i^*$ is unambiguous rational if and only if $R = \Phi(L)$ where $L \subseteq \Gamma^*$ is rational and $\Phi : \Gamma^* \to \prod \Sigma_i^*$ is a multimorphism (resp. strict alphabetical multimorphism) which is* injective *on L.*

Proof. 1) Let $L \subseteq \Gamma^*$ be computed by $\mathcal{A}_L$ and $\Phi : \Gamma^* \to \prod \Sigma_i^*$ be a multimorphism. A simple non deterministic automaton $\mathcal{R}_{\mathcal{A},\Phi}$ which computes $\Phi(L)$ acts as follows : it guesses the successive letters of a word $w \in \Gamma^*$, simulates $\mathcal{A}_L$ to check that $w \in L$ and compares $\Phi(w)$ to the input.
If the restriction of Φ to L is injective and if $\mathcal{A}_L$ is unambiguous (which we may suppose without loss of generality) then the above automaton $\mathcal{R}_{\mathcal{A},\Phi}$ is

clearly unambiguous. This proves that $\Phi(L)$ is unambiguous.
2) Now, le R be a rational relation computed by a k-tape automaton $\mathcal{A}$. Let $\delta_{\mathcal{A}} \subseteq Q_{\mathcal{A}} \times \prod \Sigma_i^* \times Q_{\mathcal{A}}$ be the finite set of transitions of $\mathcal{A}$. Considering $\delta_{\mathcal{A}}$ as an alphabet Γ, automaton $\mathcal{A}$ can be viewed as a 1-tape automaton computing a language $L \subseteq \Gamma^*$. Also, the label function $(q, \overline{u}, r) \mapsto \overline{u}$ from Γ into $\prod \Sigma_i^*$ has a unique extension to a multimorphism $\Phi : \Gamma^* \to \prod \Sigma_i^*$. It is clear that $\Phi(L) = R$.
Clearly, $\mathcal{A}$ is unambiguous if and only if the restriction of Φ to L is injective. Lastly, observe that if $\mathcal{A}$ is normal (which can be supposed without loss of generality) then Φ is strictly alphabetical. □

As can be expected, the answer to the decision problem relative to the above characterization is much the same as in Cor.2.6. However, this answer also depends on the cardinality of the source alphabet Γ of L and Φ.

Proposition 2.8. *1) If at least two of the alphabets Σ_i's are non unary then*

i) There is a finite alphabet Γ and a rational language $L \subset \Gamma^*$ such that the family of multimorphisms $\Phi : \Gamma^* \to \prod \Sigma_i^*$ which are injective on L is Π_1^0-complete hence undecidable.

ii) There is a strict alphabetical multimorphism $\Phi : \{0,1,2,3\}^* \to \prod \Sigma_i^*$ such that the family of rational languages on which Φ injective on L is Π_1^0-complete hence undecidable.

2) If Γ has at most 3 letters then there is a polynomial time algorithm to decide whether an alphabetical *multimorphism $\Phi : \Gamma^* \to \prod \Sigma_i^*$ is injective on a rational language $L \subseteq \Gamma^*$.*

3) If there is at most one non unary alphabet then there is a polynomial time algorithm to decide whether a multimorphism $\Phi : \Gamma^* \to \prod \Sigma_i^*$ is injective on a rational language $L \subseteq \Gamma^*$.

Remark 2.9. 1) Point *1i)* cannot be improved with a restriction to alphabetical multimorphisms since for a fixed finite Γ there are finitely many alphabetical multimorphisms $\Phi : \Gamma^* \to \prod \Sigma_i^*$.
2) The proof of point *1i)* (together with the best known bound $m = 7$ for the undecidability of PCP_m, [17]) leads to an alphabet Γ with 14 symbols. We do not know what is the least possible cardinality of Γ.

Proof. 1) It is sufficient to consider the case $k = 2$ and $\Sigma_1 = \Sigma_2 = \{0,1\}$. We keep the notations of the proof of Thm.2.4.
i) Set $\Gamma = \{1, \ldots, 2m\}$ and $L = \{1, \ldots, m\}^* \cup \{m+1, \ldots, 2m\}^*$. Let

$\Phi_{\varphi,\psi} : \Gamma^* \to \{0,1\}^* \times \{0,1\}^*$ be the multimorphism such that

$$\Phi_{\varphi,\psi}(i) = \text{IF } 1 \leq i \leq m \text{ THEN } (0^i 1, \varphi(i)) \text{ ELSE } (0^{i-m}1, \psi(i))$$

It is clear that $\Phi_{\varphi,\psi}$ is injective on L if and only if $(\varphi,\psi) \notin PCP_m$. This last property is Π_1^0-complete for $m \geq 7$ ([17], cf. proof of Thm.thm:unamb-undec).

ii) Let $\Phi : \{0,1,2,3\}^* \to \prod \Sigma_i^*$ be the strict alphabetical multimorphism such that

$$\Phi(0) = (\epsilon, 0)\ ,\ \Phi(1) = (\epsilon, 1)\ ,\ \Phi(2) = (0, \epsilon)\ ,\ \Phi(3) = (1, \epsilon)$$

Set $L_{\varphi,\psi} = \{2^i 3\varphi(i) \ : \ i = 1, \ldots, m\} \cup \{2^i 3\psi(i) \ : \ i = 1, \ldots, m\}$. It is clear that $\Phi(2^i 3\varphi(i)) = (0^i 1, \varphi(i))$ and $\Phi(2^i 3\psi(i)) = (0^i 1, \psi(i))$. So that Φ is injective on $L_{\varphi,\psi}$ if and only if $(\varphi,\psi) \notin PCP_m$.

2) We first reduce the problem of injectivity of multimorphisms on rational languages to the ambiguity problem of multitape automata.

Let $\Phi : \Gamma^* \to \prod \Sigma_i^*$ be a multimorphism, $L \subseteq \Gamma^*$ be a rational language and $\mathcal{A}$ be a deterministic (one-tape) automaton computing L. Define $\mathcal{A}^\Phi$ as the non deterministic k-tape automaton computing $\Phi(L)$ as follows:

- $\mathcal{A}^\Phi$ (non deterministically) guesses a word $u \in \Gamma^*$,
- $\mathcal{A}^\Phi$ compares $\Phi(u)$ with its input $\overline{\sigma} \in \prod \Sigma_i^*$,
- $\mathcal{A}^\Phi$ emulates $\mathcal{A}$ to check if $u \in L$,
- $\mathcal{A}^\Phi$ accepts if $u \in L$ and $\Phi(u) = \overline{\sigma}$.

It is clear that the accepting runs of $\mathcal{A}^\Phi$ on an input $\overline{\sigma} \in \prod \Sigma_i^*$ are in a 1-1 correspondence with the words $u \in \Phi^{-1}(\overline{\sigma})$. Thus, Φ is injective on L if and only if $\mathcal{A}^\Phi$ is unambiguous.

A priori, the alphabets of the k tapes of $\mathcal{A}^\Phi$ are the Σ_i's. Of course, one can reduce the i-th alphabet Σ_i to the subalphabet X_i formed by the letters of the i-th components of the multiwords in $\Phi(\Gamma)$.

Now, suppose $\Phi : \Gamma^* \to \prod \Sigma_i^*$ is alphabetical and let

$$\Gamma_i = \{\gamma \in \Gamma \ : \ \Phi(\gamma) \text{ has a non empty i-th component}\}$$

Clearly X_i is the set of i-th components of $\Phi(\Gamma_i)$.

The Γ_i's form a partition of Γ and (denoting $\sharp(X)$ the cardinality of a set X) we have $\sharp(X_i) \leq \sharp(\Gamma_i)$.

Now, if Γ has at most 3 letters then there is at most one component i for which Γ_i has more than one element. A fortiori, there is at most one component i for which X_i has more than one element. But this means that $\mathcal{A}^\Phi$ has at most one non unary alphabet. So that Cor.1.31 allows to decide in polynomial time if $\mathcal{A}^\Phi$ is unambiguous.

3) Suppose now there is one non unary alphabet Σ and k unary alphabets. Let $L \subseteq \Gamma^*$ be a rational language and $\Phi = (\varphi, \Psi)$ a multimorphism with $\varphi : \Gamma^* \to \Sigma^*$, $\Psi : \Gamma^* \to (\{0\}^*)^k$. Set

$$T = \{(\varphi(\alpha), \Psi(\alpha), \Psi(\beta)) \ : \ \alpha, \beta \in L \ \wedge \ (\varphi(\alpha) = \varphi(\beta) \ \wedge \ \alpha \neq \beta)\}.$$

Clearly, Φ is injective on L if and only if the projection of T on $(\{0\}^*)^{2k}$ is disjoint from the diagonal of $(\{0\}^*)^k \times (\{0\}^*)^k$. To conclude via Corollary 1.31, it suffices to prove that T is rational and construct in polynomial time some automaton for T.

Now, the condition $\alpha \neq \beta$ can be expressed as the disjunction of conditions:

(i) α is a strict prefix of β

(ii) β is a strict prefix of α

(iii) $\alpha = \xi a \eta$ and $\beta = \xi b \zeta$ for some $\xi, \eta, \zeta \in \Gamma^*$ and $a, b \in \Gamma$ and $a \neq b$

so that, with obvious notations

$$T = T_< \ \cup \ T_> \ \cup \ \bigcup_{a,b \in \Sigma, a \neq b} T_{a,b}$$

Let $\mathcal{A}$ be a one-tape deterministic automaton computing L and denote $\mathcal{A}_p$ and L_p (resp. $\mathcal{A}^p$ and L^p) the automaton $\mathcal{A}$ with p as the unique initial (resp. final) state and the language it computes. We then have

$T_< = \bigcup_{p \in Q_{\mathcal{A}}} (\varphi, \Psi, \Psi)(L^p) \ (\{\epsilon\} \times \{\overline{\epsilon}\} \times \Psi(L_p \cap (\varphi^{-1}(\epsilon) \setminus \{\epsilon\})))$

$T_> = \bigcup_{p \in Q_{\mathcal{A}}} (\varphi, \Psi, \Psi)(L^p) \ (\{\epsilon\} \times \Psi(L_p \cap (\varphi^{-1}(\epsilon) \setminus \{\epsilon\})) \times \{\overline{\epsilon}\})$

$T_{a,b} = \bigcup_{p \in Q_{\mathcal{A}}} (\varphi, \Psi, \Psi)(L^p) E_{p,a,b}$

where

$$E_{p,a,b} = \{(\varphi(au), \Psi(au), \Psi(bv)) \ : \ au, bv \in L_p \wedge \varphi(au) = \varphi(bv)\}$$

All these sets are obviously rational, hence also T, and associated automata can easily be constructed in polynomial time. □

3 Modelization of deterministic rational relations

For a detailed study of deterministic rational relations, we refer to Pelletier & Sakarovitch, 1999 [23] and Sakarovitch, 2001 [28]. Here, we shall be concerned with the problem of modelization of determinism along the different approaches to rational relations described in §1.

3.1 End-markers and super-deterministic automata

The obvious notion of deterministic *ROTM* leads to the natural class of deterministic rational relations. Hence, we would like to define a reasonable notion of deterministic labelled graph automata leading to the same class.

However, from the very start of the theory (Rabin & Scott, 1959 [26], p.85-86, Elgot & Mezei, 1965 [6], p.48, Fischer & Rosenber, 1968[7], p.89), through its development (Bird, 1973 [2], Kinber, 1983 [14], Harju & Karhumäki, 1991, [12], ...), up to the most recent papers (Pelletier & Sakarovitch, 1999 [23] §2.2) , the notion of deterministic automata which has been considered

does not lead directly to the class of deterministic rational relations. A detour is made via auxiliary relations obtained by adding end-markers. We recall this classical definition (together with a variant in the vein of Def.3.8 below).

Definition 3.1. Let $\mathcal{A} = \langle Q, \prod \Sigma_i^*, E, I, F \rangle$ be a multitape automaton.

1) $\mathcal{A}$ is *super-deterministic* if it has a unique initial state and if the labels of two distinct edges with a common origin are prefix incompatible (i.e. have no common extension).

2) [Rabin & Scott, 1959 [26]] $\mathcal{A}$ is *normal super-deterministic* if if it has a unique initial state and if there is a partition $(Q_1, \ldots, Q_k)$ of Q such that
i) for all $i \in \{1, \ldots, k\}$ and for all $p \in Q_i$ the label of any edge with origin p has all components empty but the i-th which is a letter. In particular, the support of the label is $\{i\}$ (cf. Def.1.12).
ii) different edges with origin p have different labels.

These notions are effective.

Proposition 3.2. *One can decide in polynomial time whether an automaton is super-deterministic or normal super-deterministic.*

Remark 3.3. Super-deterministic automata are called deterministic in the literature. The reason why we depart from the standard terminology will be clear from Def.3.8 and Thm.3.17 below.

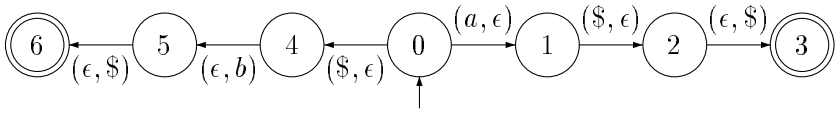
Remark 3.4. Exact polynomial complexity in Prop.3.2 depends on the parameters taken into account for $\mathcal{A}$ (number of edges, number of nodes, outer degree, i.e. maximum number of edges having a common origin) and the presentation of the automaton (as a list of labelled edges or as a list of pairs consisting of a node and the list of labelled edges coming out of that node). This will apply as well to Prop.3.20 below.

The folllowing simple fact insures that Definition 3.1 is sound.

Proposition 3.5. *Let $R \subseteq \Sigma_1^* \times \ldots \times \Sigma_k^*$ and let $R^{\$}$ be obtained by adding an end-marker \$ to each word in tuples of R.*
R is computed by a deterministic ROTM if and only if $R^{\$}$ is the behaviour of a super-deterministic (resp. normal super-deterministic) multitape automaton.

The reason for such a detour is due to a difficulty arising from commutation of multiwords having disjoint supports (cf. Def.1.12). This is illustrated by the following simple example. Let a, b be letters and consider the relation

$R = \{(a, \epsilon), (\epsilon, b)\}$. This relation is obviously $ROTM$ deterministic. However, it is easy to see that it is not the behaviour of any super-deterministic automaton. Whereas, $R^{\$} = \{(a\$, \$), (\$, b\$)\}$ is easy to compute by a normal super-deterministic automaton:



Thus, despite the fact that multitape automata are always identified with labelled graph automata, the modelization of determinism is still very much reminiscent of the $ROTM$ model with the blank symbol appearing at the right of each input.

3.2 What is a deterministic automaton?

Nevertheless, *there does exist* some reasonable notion of deterministic labelled graph automata which directly compute deterministic rational relations. As far as we know (and surprisingly as it may be), such notions seem to be original.

Before entering the drier stuff of formal definitions, let's illustrate the intuition on an example.

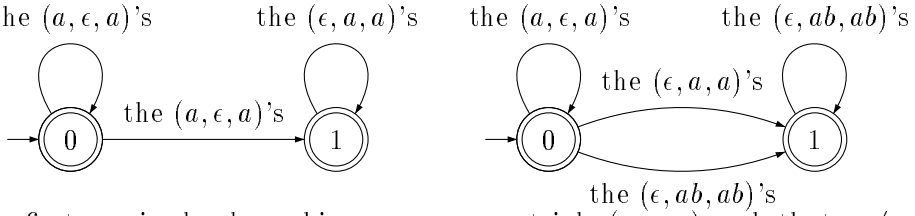
Example 3.6. The graph of the concatenation function,

$$\{(u, v, w) \ : \ uv = w\}$$

is obviously $ROTM$ deterministic: first read (u, ϵ, u) and then (ϵ, v, v). It is easy to see that it is not the behaviour of a super-deterministic automaton (argue as above with (a, ϵ, a) and (ϵ, a, a) instead of $R = \{(a, \epsilon), (\epsilon, b)\}$).

Let's look at diverse automata which compute it.

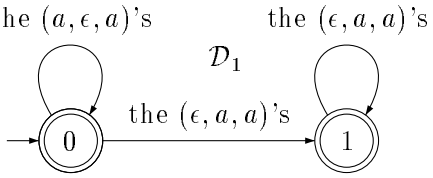
1) First, two automata which can in no way be deterministic.



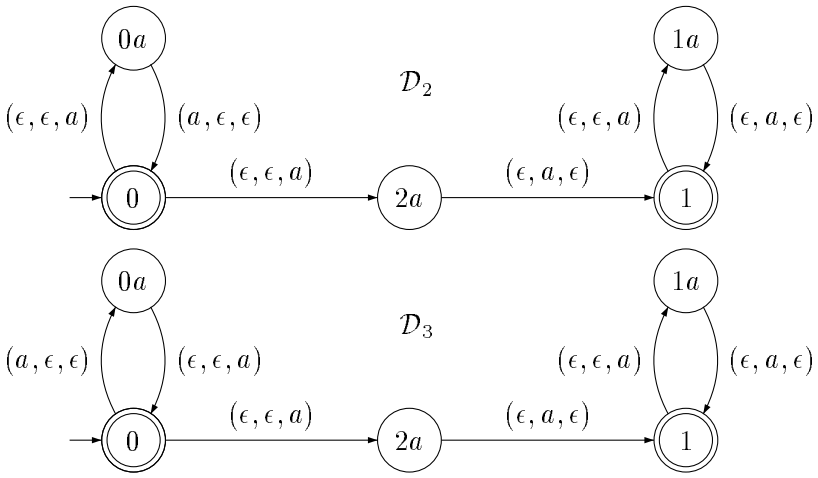
The first one is clearly ambiguous on every triple (u, ϵ, u) such that $u \neq \epsilon$, hence surely non deterministic.

The second one is unambiguous. However, to decide which transition is the right one to go from state 0 to state 1, one has to know whether $|v|$ is even or odd, and this is known only when v is completely read. So, there is no way for determinism.

2) Now, an automaton $\mathcal{D}_1$ which is not super-deterministic because there are transitions $0 \xrightarrow{(a,\epsilon,a)} 0$ and $0 \xrightarrow{(\epsilon,a,a)} 1$ with prefix compatible labels.



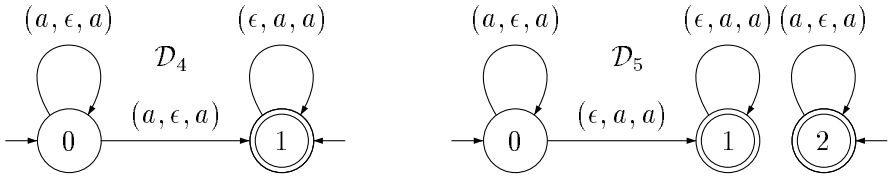
However, the first transition reads an a on tape 1 whereas the second does not and leads to a state from which it is no more possible to read anything on tape 1 (we shall call such a state an 1-end). Thus, there is no problem to decide which transition is the right one. So, we shall consider this automaton as deterministic. The same with the following normalized versions $\mathcal{D}_2, \mathcal{D}_3$ (which have $2 + 3|\Sigma|$ states):



3) Now, automata $\mathcal{D}_4, \mathcal{D}_5$ which are not super-deterministic for two reasons:

i) There are transitions with prefix compatible labels.

ii) There are *several* initial states.



Let's consider the first automaton.

As for *i)*, transitions $0 \xrightarrow{(a,\epsilon,a)} 0$ and $0 \xrightarrow{(a,\epsilon,a)} 1$ have the same label. However, the second one leads to a state (namely 1) from which it is no more possible

to read anything on tape 1. Whereas, from the first one, one is forced to read something on tape 1 in order to go to a final state. Thus, there is no problem to decide which transition is the right one.
As for *ii)*, a similar argument does work. If we start at state 0, we are forced to read something on tape 1 in order to go to a final state. Whereas, if we start at state 1, then nothing can be read on tape 1. Thus, there is no problem to decide which initial stateis the right one to start with.
Let's now consider the second automaton.
As for *i)* argue as with automaton of point 2 above. As for *ii)*, if we start from state 2 then nothing can be read on tape 2. Whereas, from state 0 we are forced to read tape 2 in order to go to a final state. So, again, there is no problem to decide which initial state to start with.

Thus, we shall also consider these automata as deterministic.

Conclusion. From the above examples, we see that *the prefix incompatibility of the labels of different transitions from a given state is not a condition inherent to determinism.* Also, *the unicity of the initial state is no more inherent to determinism, it's a mere normalization condition.*

Remark 3.7. There is still another reason which could be considered to get determinism. Let say that a state q is i-consistent if the label of every path from q to a final state has a non empty i-th component.
Suppose q is i-consistent and r is an i-end. If $p \xrightarrow{\overline{\alpha}} q$ and $p \xrightarrow{\overline{\alpha}} s$ are two transitions then there is no problem to choose between these two transitions: just look ahead at the i-th component of the input.
However, we shall not retain this type of deterministic character. The reason is that it does not carry to subautomata, contrarily to all above deterministic characters (cf. Prop.3.13 below).

3.3 Strongly deterministic and n-deterministic automata

Now, we come to the desired definitions and introduce two types of deterministic multitape automata.

Definition 3.8 (Strong determinism). 1) A state p is an *i-end* ($1 \leq i \leq k$) for a k-tape automaton $\mathcal{A}$ if any path from p to a state in $F_{\mathcal{A}}$ has an empty i-th component.

2) $\mathcal{A}$ is *strong deterministic* if it has a unique initial state and if for every pair of distinct transitions $(p, \overline{\xi}, q)$, $(p, \overline{\eta}, r)$ with the same origin, at least one of the following conditions hold:
i) $\overline{\xi}$ and $\overline{\eta}$ are prefix incompatible

ii) q is an i-end and $\overline{\xi}(i)$ is a strict prefix of $\overline{\eta}(i)$ for some $i \in \{1, \ldots, k\}$
iii) r is an i-end and $\overline{\eta}(i)$ is a strict prefix of $\overline{\xi}(i)$ for some $i \in \{1, \ldots, k\}$

3) $\mathcal{A}$ is *normal strong deterministic* if it is strong deterministic and normal (cf. Def.1.11).

Remark 3.9. A super-deterministic automaton (cf. Def.refdef:classicdet) is obviously strong deterministic.

Notation 3.10. 1) If $n \in \mathbf{N} \cup \{\infty\}$, u is a word, $\overline{u} = (u_1, \ldots, u_k)$ is a multi-word, we let

$$\max(n, u) = \max(n, |u|) \ , \ \max(n, \overline{u}) = (\max(n, |u_1|), \ldots, \max(n, |u_k|))$$

2) If $\overline{p} = (p_1, \ldots, p_k) \in (\mathbf{N} \cup \{\infty\})^k$ and R a relation on words, we let

$$\begin{aligned}
n\text{-Prefix}(u) &= \text{ the prefix of } u \text{ with length } \min(n, |u|) \\
n\text{-Prefix}(\overline{u}) &= (n\text{-Prefix}(u_1), \ldots, n\text{-Prefix}(u_k)) \\
n\text{-Prefix}(R) &= \{n\text{-Prefix}(\overline{u}) \ : \ \overline{u} \in R\} \\
\overline{p}\text{-Prefix}(\overline{u}) &= (p_1\text{-Prefix}(u_1), \ldots, p_k\text{-Prefix}(u_k)) \\
\overline{p}\text{-Prefix}(R) &= \{\overline{p}\text{-Prefix}(\overline{u}) \ : \ \overline{u} \in R\}
\end{aligned}$$

Definition 3.11 (n-determinism). 1) $\mathcal{A}$ is *n-deterministic* if the following two conditions are satisfied.

i) If p, q are distinct initial states then

$$n\text{-Prefix}(Rel(\mathcal{A}_p)) \ \cap \ n\text{-Prefix}(Rel(\mathcal{A}_q)) \ = \ \emptyset$$

ii) If $(p, \overline{\xi}, q)$, $(p, \overline{\eta}, r)$ are distinct transitions with the same origin then

$$\max(n, \overline{\xi}, \overline{\eta})\text{-Prefix}(\overline{\xi} Rel(\mathcal{A}_q)) \ \cap \ \max(n, \overline{\xi}, \overline{\eta})\text{-Prefix}(\overline{\eta} Rel(\mathcal{A}_r)) \ = \ \emptyset$$

In case all labels of transitions of $\mathcal{A}$ have length $\leq n$ (in particular if $n \geq 1$ and $\mathcal{A}$ is normal or quasi-normal, cf. Def.1.11) then condition ii) can be expressed in a simpler form:

iibis) $n\text{-Prefix}(\overline{\xi} Rel(\mathcal{A}_q)) \ \cap \ n\text{-Prefix}(\overline{\eta} Rel(\mathcal{A}_r)) = \emptyset$

2) $\mathcal{A}$ is *normal n-deterministic* if it is n-deterministic and normal.

Example 3.12. Let's review the deterministic character of the automata introduced in Example 3.6.
Automaton $\mathcal{D}_1$ is strong deterministic: $(\epsilon, a, a)_1 = \epsilon <_{prefix} a = (a, \epsilon, a)_1$ and state 1 is an 1-end.
Similarly, automaton $\mathcal{D}_3$ is normal strong deterministic: $(\epsilon, \epsilon, a)_1 = \epsilon <_{prefix} a = (a, \epsilon, \epsilon)_1$ and state $2a$ is an 1-end.

Automata $\mathcal{D}_2, \mathcal{D}_4, \mathcal{D}_5$ are not strong deterministic:

- there are two $\mathcal{D}_2$-transitions from state 0 with the same label (ϵ, ϵ, a),
- there are two $\mathcal{D}_4$-transitions from state 0 with the same label (a, ϵ, ϵ),
- $\mathcal{D}_5$ has two initial *states* 0 and 2.

However, $\mathcal{D}_2, \mathcal{D}_5 are 1-deterministic and \mathcal{D}_5$ is 4-deterministic. (since these automata are quasi-normal, condition *iibis* is to be checked):

$$\begin{aligned}(\epsilon,\epsilon,a)Rel((\mathcal{D}_2)_{0a})) &= \{(a^{1+m}, a^{1+n}, a^{2+m+n}) \ : \ m,n \in \mathbf{N}\}\\ (\epsilon,\epsilon,a)Rel((\mathcal{D}_2)_{2a})) &= \{(\epsilon, a^{1+n}, a^{1+n}) \ : \ n \in \mathbf{N}\}\end{aligned}$$

and the associated 1-Prefix relations are disjoint.

$$\begin{aligned}(a,\epsilon,a)Rel((\mathcal{D}_4)_0)) &= \{(a^{2+m}, a^{n}, a^{2+m+n}) \ : \ m,n \in \mathbf{N}\}\\ (a,\epsilon,a)Rel((\mathcal{D}_4)_1)) &= \{(a, a^{n}, a^{1+n}) \ : \ n \in \mathbf{N}\}\end{aligned}$$

and the associated 2-Prefix relations are disjoint.

$$\begin{aligned}Rel((\mathcal{D}_5)_0)) &= \{(a^{m}, a^{1+n}, a^{1+m+n}) \ : \ m,n \in \mathbf{N}\}\\ Rel((\mathcal{D}_5)_2)) &= \{(a,\epsilon,a)\}\end{aligned}$$

and the associated 1-Prefix relations are disjoint. Also,

$$\begin{aligned}(a,\epsilon,a)Rel((\mathcal{D}_5)_0)) &= \{(a^{1+m}, a^{1+n}, a^{2+m+n}) \ : \ m,n \in \mathbf{N}\}\\ (\epsilon,a,a)Rel((\mathcal{D}_5)_1)) &= \{(\epsilon, a^{1+n}, a^{1+n}) \ : \ n \in \mathbf{N}\}\end{aligned}$$

and the associated 1-Prefix relations are disjoint.
It is easy to check that $\mathcal{D}_4$ is not 1-deterministic.

As announced in Remark 3.7, we have the following property (which is obvious from the definitions).

Proposition 3.13. *If an automaton is n-deterministic (resp. strong deterministic) then so is the automaton obtained by suppressing any collection of nodes or edges.*

The following result is easy.

Proposition 3.14. *1) For all* $n \in \mathbf{N}$

$\mathcal{A}$ *is strong deterministic* $\Rightarrow$ $\mathcal{A}$ *is* 0*-deterministic*

$\mathcal{A}$ *is n-deterministic* $\Rightarrow$ $\mathcal{A}$ *is* $(n+1)$*-deterministic*

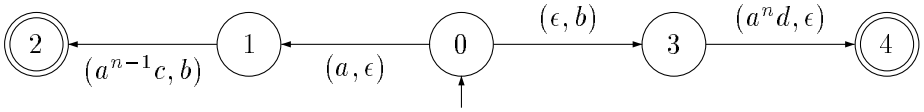
2) The above implications cannot be reversed.

Proof. Point 1 is straightforward. As for Point 2, the first implication cannot be reversed since a 0-deterministic automaton with two distinct initial states cannot be strong deterministic.
We now deal with the second implication. For $n = 0$, observe that the following automaton is 1-deterministic but not 0-deterministic: the relations computed from the two initial states are

$$Rel(\mathcal{A}_0) = \{(a, \epsilon)\} \ , \ Rel(\mathcal{A}_2) = \{(\epsilon, b)\}$$

so that their 1-Prefixes are distinct but their 0-Prefixes are equal.

For $n \geq 1$, consider the following automaton $\mathcal{A}$:



We have

$$(a, \epsilon)Rel(\mathcal{A}_1) = \{(a^n c, b)\} \ , \ (\epsilon, b)Rel(\mathcal{A}_3) = \{(a^n d, b)\}$$

so that if $c \neq d$ then $\mathcal{A}$ is $(n+1)$-deterministic but not n-deterministic.

3.4 Ambiguity and ∞-determinism

Though straightforward, the following result is worth noticing.

Proposition 3.15. *1) For all $n \in \mathbf{N}$, every n-deterministic automaton is ∞-deterministic.*

2) An automaton $\mathcal{A}$ is ∞-deterministic if and only if it is unambiguous.

Remark 3.16. However, there are unambiguous automata which are not n-deterministic for any n. For instance, the second automaton given in point 1 of Example 3.6.

3.5 Deterministic automata compute what is expected

Theorem 3.17. *Let R be a relation. The following conditions are equivalent:*

i) For some $n \in \mathbf{N}$ the relation R is the behaviour of some n-deterministic automaton

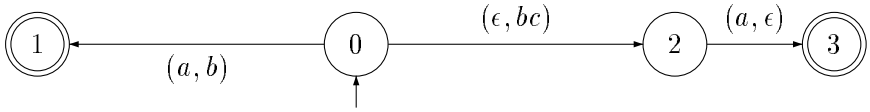
ii) For all $n \in \mathbf{N}$ the relation R is the behaviour of some normal n-deterministic automaton

iii) R is the behaviour of some strong deterministic automaton.

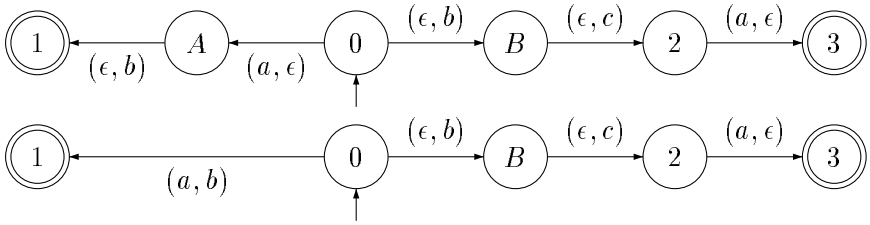
iv) R is the behaviour of some normal strong deterministic automaton.
v) R is computed by some deterministic $ROTM$
vi) R is computed by some modified deterministic $ROTM$

Before coming to the proof, we observe the following fact.

Remark 3.18. Normalization and quasi-normalization of automata (cf. Def. 1.13) *do not preserve* neither strong determinism nor n-determinism. A counterexample is obtained by considering the following strongly deterministic automaton $\mathcal{A}$ which computes $\{(a,b),(a,bc)\}$.



The associated normalized and quasi-normalized automata $\mathcal{A}^{nl}$ and $\mathcal{A}^{qnl}$ are as follows (where A,B stand for states $(0,(a,\epsilon)),(0,(\epsilon,b))$):



$\mathcal{A}^{nl}$ is not strongly deterministic: the pair of transitions starting at state 0 violates the condition for strong determinism since A is not a 2-end and B is not a 1-end. $\mathcal{A}^{nl}$ is not even 1-deterministic since

$$\text{1-Prefix}((a,\epsilon)Rel(\mathcal{A}_A^{normal})) = \text{1-Prefix}((\epsilon,b)Rel(\mathcal{A}_B^{normal})) = \{(a,b)\}$$

However, $\mathcal{A}^{nl}$ is 2-deterministic. All the same properties hold with $\mathcal{A}^{qnl}$.

Proof of Theorem 3.17. We shall prove implications $i \Rightarrow vi$, $vi \Rightarrow iii$, $vi \Rightarrow iv$. All other implications follow from these ones and Propositions 3.14, 1.19.

$i \Rightarrow vi$ Given an n-deterministic automaton $\mathcal{A}$, we describe a deterministic $ROTM$ $\mathcal{T}$ which has the same behaviour. Let m be the maximum width of labels of transitions of $\mathcal{A}$. Then $\mathcal{T}$ acts as follows:
α) Before emulating any $\mathcal{A}$-transition, $\mathcal{T}$ reads its tapes so as to memorize up to $\max(n,m)$ letters of each one of the k inputs (an information it retains in its state).
β) When $\mathcal{T}$ has completed this memorization, it emulates a transition of $\mathcal{A}$ (which is necessarily unique, since $\mathcal{A}$ is n-deterministic), changes state accordingly and forgets the portion of the memorized input corresponding to the label of the simulated transition.

$vi \Rightarrow iii$ Def.1.22 associates to a modified $ROTM$ $\mathcal{T}$ a quasi-normal automaton $\mathcal{A}$ which computes the same relation as $\mathcal{T}$ does (cf. Prop.1.21). We show that if $\mathcal{T}$ is a *deterministic* modified $ROTM$ then $\mathcal{A}$ is strong deterministic.

Since T has a unique initial state so does A. Thus, the first condition for strong determinism is satisfied.

(A) $\delta_{\mathcal{A}}$ is functional

i.e. two edges with the same origin $(q, \overline{\xi})$ and label $\overline{\alpha}$ are equal.
In fact, suppose $((q, \overline{\xi}), \overline{\alpha}, (r, \overline{\eta}))$ is an $\mathcal{A}$-transition. Observe that there is a unique $\mathcal{T}$-transition $t = (q, \overline{a}, r, \overline{m})$ such that $((q, \overline{\xi}), \overline{\alpha}, (r, \overline{\eta})) \in Trans(t)$. The reason is that $\overline{a}$ and $\overline{m}$ are determined as follows:

$$a_i = \text{IF } \xi_i \neq new \text{ THEN } \xi_i \text{ ELSE (IF } \alpha_i = \epsilon \text{ THEN } B \text{ ELSE } \alpha_i) \quad (9)$$

$$m_i = \text{IF } \eta_i = new \text{ THEN 1 ELSE 0} \quad (10)$$

Since $\mathcal{T}$ is deterministic, from q and $\overline{a}$ we get r and $\overline{m}$. Combined with equation 9 and the definition of $\overline{\eta}$ from $\overline{m}$, this proves that from $q, \overline{\xi}, \overline{\alpha}$ we get r and $\overline{\eta}$, i.e. $\delta_{\mathcal{A}}$ is functional.

(B) Any $\mathcal{A}$-state $(s, \overline{\theta})$ such that $\theta_i = B$ is clearly an i-end

Consider now two distinct edges $((q, \overline{\xi}), \overline{\alpha}, (r, \overline{\eta}))$ and $((q, \overline{\xi}), \overline{\beta}, (s, \overline{\theta}))$ out of some $\mathcal{A}$-state $(q, \overline{\xi})$.
Due to (A), they must have distinct labels: $\overline{\alpha} \neq \overline{\beta}$. Suppose these distinct labels are prefix compatible and let i be such that $|\alpha_i| > |\beta_i|$. Since $\mathcal{A}$ is quasi-normal, this implies that $\beta_i = \epsilon$ and that $\alpha_i \in \Sigma_i$ is a letter. Recall

$$\alpha_i = \text{ IF } (\xi_i = new \text{ and } a_i \neq B) \text{ THEN } a_i \text{ ELSE } \epsilon$$

so that, from $\alpha_i \in \Sigma_i$ we get $\xi_i = new$.
Let $t = (q, \overline{b}, s, \overline{m})$ be such that $((q, \overline{\xi}), \overline{\beta}, (s, \overline{\theta})) \in Trans(t)$. From $\xi_i = new$ and $\beta_i = \epsilon$ we get $b_i = B$. Therefore $\mathcal{T}$ makes no move on tape i and $m_i = 0$, whence $\theta_i = b_i = B$. Using (B) we see that $(s, \overline{\theta})$ is an i-end. Which proves the condition for strong deterlminism of $\mathcal{A}$.

$vi \Rightarrow iv$ The above automaton $\mathcal{A}$ is quasi-normal. To get a normal automaton, we argue as follows:
- Normalize the $ROTM$ so that it moves exactly one head per transition. Observe that the obvious way to do that *does preserve determinism.*
- Use Remark 1.23 to conclude that $\mathcal{A}$ is then normal. □

Remark 3.19. The direct way of normalizing a multitape automaton *does not* preserve neither n-determinism nor strong determinism (cf. Remark

3.18). We can use the above construction to get a (rather tortuous) method to normalize a deterministic automaton:
1) Go from $\mathcal{A}$ to a deterministic modified $ROTM\ TM$
2) Transform TM to $Atomic(TM)$ so as there is exactly one move per transition. This does preserve $ROTM$ determinism.
3) From $Atomic(TM)$ get $\mathcal{A}$ normal strong deterministic using the construction given in the proof of Thm.3.17.

3.6 Decidability of determinism for automata

Proposition 3.20. *1) The class of strong deterministic automata is polynomial time decidable (cf. Remark 3.4).*

2) Let expl-DET $= \{(n, \mathcal{A}) : \mathcal{A}$ *is n-deterministic*$\}$ *(where "expl" stands for "explicit"). Caution: n is to be considered as an object of length n, i.e. it is written in unary.*
i) If $k \geq 2$ *and there are at least two non unary alphabets then the class expl-DET is co-NP-complete.*
ii) If there is at most one non unary alphabet then the class expl-DET is polynomial time decidable.

Proof. 1) It is easy to devise a polynomial time algorithms to decide if a state is an i-end of $\mathcal{A}$.
2*i)* Deciding if the relations computed by two automata have a common n-Prefix is clearly in NP. Hence the disjointness conditions *i), ii)* of Def.3.11 for n-determinism lead to an obvious co-NP algorithm.
Conversely, a straightforward adaptation of the proof of Thm.2.4 leads to a polynomial time reduction of the bounded Post Correspondence Problem (which is PCP in which we want a solution u with length $\leq n$) to the non-disjointness problem of the n-Prefixes of relations computed by finite *super-deterministic* automata. This last problem reduces easily to the complement of the class $expl\text{-}DET$. We conclude using the well-known NP-completeness of the bounded-PCP (cf. [8] p.228).
2*ii)* An automaton $\mathcal{A}^{[n]}$ computing n-Prefix$(Rel(\mathcal{A}))$ is as follows:
- $Q_{\mathcal{A}^{[n]}} = Q_{\mathcal{A}} \times \{0, 1, \ldots, n\}^k$,
- $\mathcal{A}^{[n]}$ counts the number of letters read on each of the k components,
- $\mathcal{A}^{[n]}$ emulates $\mathcal{A}$ while all counts are $\leq n$. Such an automaton can be constructed in polynomial time. Construct (as above) automata computing
- relations n-Prefix$(Rel(\mathcal{A}_p))$ where p is an initial state of $\mathcal{A}$,
- relations $\max(n, \overline{\xi}, \overline{\eta})$-Prefix$(\overline{\xi} Rel(\mathcal{A}_p))$ where $\overline{\xi}, \overline{\eta}$ are labels of transitions starting at p,

Using these automata (which can be constructed in polynomial time) and the fact that at most one alphabet is non unary, we can apply Cor.1.31 to check in polynomial time the disjointness of relations occuring in condition *ii)* of Def.3.11 for n-determinism. □

However, for existentially quantified n the problem is in general undecidable.

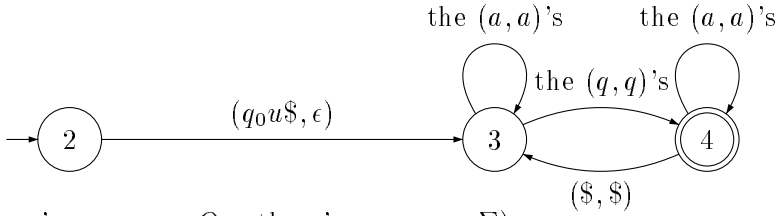
Proposition 3.21. *Let* $DET_{\Sigma_1,\dots,\Sigma_k} = \{\mathcal{A} \ : \ \exists n \ (\mathcal{A} \text{ is } n\text{-deterministic})\}$.
1) Suppose $k \geq 2$ *and there are at least two non binary alphabets. Then* $DET_{\Sigma_1,\dots,\Sigma_k}$ *is* Σ_1^0*-complete hence undecidable.*

2) If there is at most one non unary alphabet then $DET_{\Sigma_1,\dots,\Sigma_k}$ *is decidable.*

Remark 3.22. We do not know the exact complexity of the class $DET_{\Sigma_1,\dots,\Sigma_k}$ in case there is at most one non unary alphabet. It is bounded by that of Presburger arithmetic.

Proof. of Prop. 3.21. 1) Let $\mathcal{M}$ be a deterministic Turing machine with input alphabet Σ, set of states Q and initial state q_0. Sequences of instantaneous descriptions (i.d.) of $\mathcal{M}$ can be coded as words in alphabet $\Delta = \Sigma \cup Q \cup \{\$\}$, where \$ serves as a flag separating successive i.d.'s. If I is an i.d. with non final state, we denote I^+ the i.d. obtained from I with one $\mathcal{M}$-transition.
For each $u \in \Sigma^*$ we define 2-tape automata $\mathcal{A}^u, \mathcal{B}^u$ as follows:
i) $\mathcal{A}^u$ computes the relation

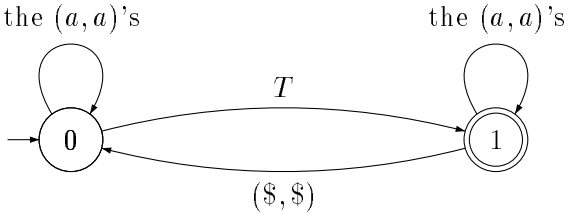
$$\{(q_0 u\$I_1\$I_2\$\dots\$I_t \ , I_1\$I_2\$\dots\$I_t) \ : \text{ the } \ I_i\text{'s are i.d.'s}\}$$



(The q's vary over $Q_{\mathcal{A}}$, the a's vary over Σ).

ii) $\mathcal{B}^u$ computes the relation

$$\{(I_0\$I_1\$\dots\$I_t \ , I_0^+\$I_1^+\$\dots\$I_t^+) \ : \text{ the } \ I_i\text{'s are non final i.d.'s}\}$$



Labels in T are $(bqa, rbc), (qa, rc), (qa, cr)$ according to the emulated $\mathcal{M}$-transition

$$\delta(q,a) = (r,c,-1) \text{ or } (r,c,0) \text{ or } (r,c,1)$$

(where $-1, 0, 1$ mean "left move", "no move", "right move" and c is what $\mathcal{M}$ writes in place of a).

It is clear that $\mathcal{A}^u$ is super-deterministic. Also, $\mathcal{B}^u$ is 0-deterministic:

- transitions starting at state 1 have incompatible labels,
- transitions in T starting at state 0 have incompatible labels,
- for transitions $(0,(a,a),0)$ and $(0,\xi,1) \in T$, the value $\max(0,(a,a),\xi)$ is 2 or 3 and

$\max(0,(a,a),\xi)$-Prefix$((a,a)Rel(\mathcal{B}^u_0) \cap \max(0,(a,a),\xi)$-Prefix$(\xi Rel(\mathcal{B}^u_1) = \emptyset$

Since $Rel(\mathcal{A}^u) \cup Rel(\mathcal{B}^u) = \emptyset$ the union automaton $\mathcal{A}^u \cup \mathcal{B}^u$ is always unambiguous. We now look under which condition $\mathcal{A}^u \cup \mathcal{B}^u$ is n-deterministic for some n.

First, observe that condition *ii)* in Def.3.11 is automatically satisfied with $n = 0$.

Condition *i)* in Def.3.11 is satisfied for some n if and only there is a bound to the depths of common prefixes to a pair in $Rel(\mathcal{A}^u)$ and a pair in $Rel(\mathcal{B}^u)$. Now, such common prefixes are exactly the prefixes of the pairs

$$(q_0 u \$ I_1 \$ I_2 \$ \ldots \$ I_t \ , I_1 \$ I_2 \$ \ldots \$ I_t)$$

where the I_i's are the successive instantaneous configurations of the computation of $\mathcal{M}$ on input u.

In particular, there is a bound to their depths if and only if $\mathcal{M}$ halts on input u. Thus, automaton $\mathcal{A}^u \cup \mathcal{B}^u$ is n-deterministic for some n if and only if $\mathcal{M}$ halts on input u (and the smallest such n is then the sum of the lengths of the successive i.d.'s of the finite $\mathcal{M}$-computation on input u).

Considering a universal Turing machine $\mathcal{M}$, we get a recursive reduction of the halting problem for $\mathcal{M}$ to the problem of determinism for automata. Hence the wanted undecidability and also the Σ^0_1-completeness.

2) In case there is one non unary alphabet Σ and k unary alphabets, we reduce to Presburger arithmetic as in the proof of Cor.1.27.

Let's add an n-component to the sets considered in conditions *i), ii)* of Def.3.11. For initial states q, r this leads to define

$$\begin{aligned} I^{q,r} = \{(w, \overline{s}, n) \quad : \quad & w \in \Sigma^* \ \wedge \ \overline{s} \in (\mathbf{N})^k \ \wedge \ n \in \mathbf{N} \\ & \wedge \ (w, \overline{s}) \in n\text{-Prefix}(Rel(\mathcal{A}_q)) \\ & \wedge \ (w, \overline{s}) \in n\text{-Prefix}(Rel(\mathcal{A}_q))\} \end{aligned}$$

For transitions $(p,(u,\overline{\xi}),q), (p,(v,\overline{\eta}),r)$ the sole interesting case is when p, q are prefix comparable (otherwise condition *ii)* is trivial). Thus, we restrict

to the case u is a prefix of v, i.e. v is of the form $v = uu'$. This leads to define

$$\begin{aligned} S^{q,r}_{u,u',\overline{\xi},\overline{\eta}} = \{(w,\overline{s},n) \;\; : \;\; & w \in \Sigma^* \;\wedge\; \overline{s} \in (\mathbf{N})^k \;\wedge\; n \in \mathbf{N} \\ & \wedge\; (w,\overline{s}) \in n\text{-Prefix}((u,\overline{\xi})Rel(\mathcal{A}_q)) \\ & \wedge\; (w,\overline{s}) \in n\text{-Prefix}((uu',\overline{\eta})Rel(\mathcal{A}_r))\} \end{aligned}$$

However, these relations $I^{q,r}, S^{q,r}_{u,u',\overline{\xi},\overline{\eta}}$ *are not* rational. So we introduce variants $J^{q,r}, T^{q,r}_{u,u',\overline{\xi},\overline{\eta}}$ which are rational:

$$\begin{aligned} J^{q,r} = \{(w,\overline{s},\overline{t},m,n) \;\; : \;\; & w \in \Sigma^* \;\wedge\; \overline{s},\overline{t} \in (\mathbf{N})^k \;\wedge\; m,n \in \mathbf{N} \\ & \wedge\; \exists \alpha,\beta \in \Sigma^* \; \exists \overline{\lambda},\overline{\mu} \in \mathbf{N}^k \\ & \wedge\; (\alpha,\overline{\lambda}) \in Rel(\mathcal{A}_q)) \\ & \wedge\; (\beta,\overline{\mu}) \in Rel(\mathcal{A}_r) \\ & \wedge\; (w,\overline{s}) = m\text{-Prefix}(\alpha,\overline{\lambda}) \\ & \wedge\; (w,\overline{t}) = n\text{-Prefix}(\beta,\overline{\mu})\} \end{aligned}$$

$$\begin{aligned} T^{q,r}_{u,u',\overline{\xi},\overline{\eta}} = \{(w,\overline{s},\overline{t},m,n) \;\; : \;\; & w \in \Sigma^* \;\wedge\; \overline{s},\overline{t} \in (\mathbf{N})^k \;\wedge\; m,n \in \mathbf{N} \\ & \wedge\; \exists \alpha,\beta \in \Sigma^* \; \exists \overline{\lambda},\overline{\mu} \in \mathbf{N}^k \\ & \wedge\; (\alpha,\overline{\lambda}) \in Rel(\mathcal{A}_q)) \\ & \wedge\; (\beta,\overline{\mu}) \in Rel(\mathcal{A}_r) \\ & \wedge\; (w,\overline{s}) = m\text{-Prefix}(u\alpha,\overline{\xi\lambda}) \\ & \wedge\; (w,\overline{t}) = n\text{-Prefix}(uu'\beta,\overline{\eta\mu})\} \end{aligned}$$

Clearly,

$$\begin{aligned} I^{q,r} &= \{(w,\overline{s},n) \;:\; (w,\overline{s},\overline{s},n,n) \in J^{q,r}\} \\ S^{q,r}_{u,u',\overline{\xi},\overline{\eta}} &= \{(w,\overline{s},n) \;:\; (w,\overline{s},\overline{s},n,n) \in T^{q,r}_{u,u',\overline{\xi},\overline{\eta}}\} \end{aligned}$$

Let $K^{q,r}, U^{q,r}_{u,u',\overline{\xi},\overline{\eta}}$ be the projections of $J^{q,r}, T^{q,r}_{u,u',\overline{\xi},\overline{\eta}}$ parallel to the Σ^* component.
Now, $\mathcal{A}$ is deterministic if and only if there exists n such that for all initial states p,q and all transitions $(p,(u,\overline{\xi}),q)$, $(p,(uu',\overline{\eta}),r)$

$$\{(w,\overline{s}) : (w,\overline{s},n) \in I^{q,r}\} = \emptyset \;,\; \{(w,\overline{s}) : (w,\overline{s},n) \in S^{q,r}_{u,u',\overline{\xi},\overline{\eta}}\} = \emptyset$$

i.e.

$\{(w,\overline{s}) : (w,\overline{s},\overline{s},n,n) \in J^{q,r}\} = \emptyset$, $\{(w,\overline{s}) : (w,\overline{s},\overline{s},n,n) \in T^{q,r}_{u,u',\overline{\xi},\overline{\eta}}\} = \emptyset$
This amounts to say

$$\exists n\ \forall \overline{s}\ \ (\overline{s},\overline{s},n,n) \notin K^{q,r}\ \ ,\ \ \exists n\ \forall \overline{s}\ \ (\overline{s},\overline{s},n,n) \notin U^{q,r}_{u,u',\overline{\xi},\overline{\eta}} \tag{11}$$

Since relations $K^{q,r}, U^{q,r}_{u,u',\overline{\xi},\overline{\eta}}$ are rational, they can be expressed in Presburger arithmetic (cf. Thm.1.25). Hence assertions (11) can be expressed in Presburger arithmetic and we get the wanted decidability. □

3.7 Deterministic multimorphisms

The tight relation between k-tape automata and multimorphisms (cf. point 3 of the proof of Thm.2.7) leads to a natural notion of n-determinism and strong determinism.

Definition 3.23. Let $\Phi : \Gamma^* \to \prod \Sigma_i^*$ be a multimorphism and $L \subseteq \Gamma^*$ be a rational language.

1) Φ is an i-end for a language $X \subseteq \Gamma^*$ if $\Phi(a)_i = \epsilon$ for all letters a occurring in some word in X (where $\Phi(x)_i$ denotes the i-th component of $\Phi(x)$).

2) Φ is n-deterministic on L if for all words $u \in \Gamma^*$, for all distinct letters $a,b \in \Gamma$ if ua, ub are prefixes of words in L then
$$\text{n-Prefix}(\{\Phi(ax)\ :\ uax \in L\}) \cap \text{n-Prefix}(\{\Phi(by)\ :\ uby \in L\}) = \emptyset$$
3) Φ is strong deterministic on L if for all words $u \in \Gamma^*$, for all distinct letters $a,b \in \Gamma$, if ua, ub are prefixes of words in L then at least one of the following conditions holds:
i) $\Phi(a), \Phi(b)$ are prefix-incompatible in $\prod \Sigma_i^*$
ii) $\Phi(a)_i$ is a strict prefix of $\Phi(b)_i$ and Φ is an i-end for $(ua)^{-1}L$.
iii) $\Phi(b)_i$ is a strict prefix of $\Phi(a)_i$ and Φ is an i-end for $(ub)^{-1}L$.

We can now state the deterministic version of Nivat's Multimorphism Theorem, the proof of which is straightforward from the definitions.

Theorem 3.24. *A rational relation $R \subseteq \prod \Sigma_i^*$ is n-deterministic (resp. strong deterministic, resp. normal strong deterministic) if and only if $R = \Phi(L)$ where $L \subseteq \Gamma^*$ is a rational language and $\Phi : \Gamma^* \to \prod \Sigma_i^*$ is a proper multimorphism which is n-deterministic on L (resp. strong deterministic on L, resp. strong deterministic on L and alphabetical multimorphism).*

As for the effectiveness of the notion, results are analog to those of Prop.3.21, 3.20 (with similar proofs).

Proposition 3.25. *1) Consider the class of triples $(n, \mathcal{A}, \Phi)$ such that $\mathcal{A}$ is a one-tape automaton on alphabet Γ and $\Phi : \Gamma^* \to \prod \Sigma_i^*$ is a multimorphism which is n-deterministic on $L(\mathcal{A})$. (Caution: n is to be considered as an object of length n, i.e. it is written in unary).*
i) If $k \geq 2$ and there are at least two non unary alphabets then this class is co-NP-complete.
ii) If there is at most one non unary alphabet then this class is polynomial time decidable.

2) Consider the class of pairs $(\mathcal{A}, \Phi$ such that $\mathcal{A}$ is a one-tape automaton on alphabet Γ and $\Phi : \Gamma^* \to \prod \Sigma_i^*$ is a multimorphism which is n-deterministic on $L(\mathcal{A})$ for some n.
i) If $k \geq 2$ and there are at least two non unary alphabets then this class is Σ_1^0-complete hence not recursive.
ii) If there is at most one non unary alphabet then this class is recursive

Remark 3.26. As for the notion of deterministic automaton (cf. Rk.3.22), we do not know the exact complexity in Point *2ii*. It is bounded by that of Presburger arithmetic.

4 Acknowledgements

Many thanks to Christian Choffrut and to an anonymous referee for comments and corrections.

References

[1] J. Berstel. *Transductions and context-free languages.* B. G. Teubner, 1979.

[2] M.R. Bird. The equivalence problem for deterministic two-tape automata. *J. Computer System Sci.*, 7:218–236, 1973.

[3] I. Borosh and L. Treybig. Bounds on positive integral solutions of linear diophantine equations. *Proc. Amer. Math. Soc.*, 55(2):299–304, 1976.

[4] S. Eilenberg. *Automata, languages and machines*, volume A. Academic Press, 1974.

[5] S. Eilenberg and M.P. Schützenberger. Rational sets in commutative monoids. *J. Algebra*, 13:173–191, 1969.

[6] C.C. Elgot and J.E. Mezei. On relations defined by finite automata. *IBM J. Res. and Develop.*, 9:47–68, 1965.

[7] P.C. Fischer and A. Rosenberg. Multitape one-way nonwriting automata. *J. Computer System Sci.*, 2:88–101, 1968.

[8] M; Garey and D. Johnson. *Computers and intractability. A guide to the theory of NP-compppleteness.* Freeman, 1979.

[9] S. Ginsburg and E.H. Spanier. Semigroups, presburger formulas, and languages. *Pacific J. Math.*, 16:285–296, 1966.

[10] J. Goldstine. A simplified proof of parikh's theorem. *Discrete Math.*, 19:235–240, 1977.

[11] E. Gurari and O. Ibarra. The complexity of dcision problems for finite-turn multicounter machines. *J. Computer System Sci.*, 22:220–229, 1981.

[12] T. Harju and J. Karhumäki. The equivalence problem of multitape finite automata. *Theoret. Comput. Sci.*, 78:347–355, 1991.

[13] O. Ibarra. Reversal-bounded multicounter machines and their decision probems. *J. Assoc. Comput. Mach.*, 25(1):116–133, 1978.

[14] E.B. Kinber. The inclusion problem for some classes of deterministic automata. *Theoret. Comput. Sci.*, 26:1–24, 1983.

[15] S.C. Kleene. Representations of events in nerve nets and finite automata. In *Automata Studies*, pages 3–41, 1956.

[16] H.W. Lenstra. Integer programming with a fixed number of variables. *Mathematics of Operations Research.*, 8(4):538–548, 1983.

[17] Y. Matiyasevich and G. Senizergues. Decision problems for semi-Thue systems with a few rules. In *11th Symposium on Logic in Computer Science*, pages 523–531, 1996.

[18] J. Myhill. Finite automata and the representation of events. *WADD Technical Report, Wright-Paterson Air Force Base*, 57-624, November, 1957.

[19] M. Nivat. Transductions des langages de Chomsky. *Ann. Inst. Fourier*, 108:339–456, 1968.

[20] P. Odifreddi. *Classical recursion theory*, volume 125 of *Studies in logic and the foundations of mathematics*. North-Holland, 1989.

[21] R.J. Parikh. Language-generating devices. *Quarterly Progress Report*, 60:199–212, 1961.

[22] R.J. Parikh. On context-free languages. *J. Assoc. Comput. Mach.*, 13:570–581, 1966.

[23] M. Pelletier and J. Sakarovitch. On the representation of finite deterministic 2-tape automata. *Theoret. Comput. Sci.*, 225:1–63, 1999.

[24] E. Post. A variant of a recursively unsolvable problem. *Bull. Amer. Math. Soc.*, 52:264–268, 1946.

[25] M. Presburger. Ueber die Vollständigkeit eines gewissen Systems ganzer Zahlen, in welchen die Addition als einzige Operation hervortritt. In *1er Congrès des Math. des Pays Slaves, Varsovie*, pages 92–101, 1930. English translation : On the completeness of a certain system of arithmetic of whole numbers in which addition occurs as the only operation, *History and Philosophy of Logic*, 12, 1991, 225-233.

[26] M.O. Rabin and D. Scott. Finite automata and their decision problems. *IBM J. Res. and Develop.*, 3:63–90, 1959. Reprinted in Sequential Machines, Moore editor, 1964, Addison-Wesley.

[27] H. Rogers. *Theory of recursive functions and effective computability.* McGraw-Hill, 1967.

[28] J. Sakarovitch. *Eléments de théorie des automates.* 2001. To appear.

[29] A. Schrijver. *Theory of linear and integer programming.* Wiley, 1987.

[30] M.P. Schützenberger. On the definition of a family of automata. *Inform. and Control*, 4:245–270, 1961.

[31] J. Von zur Gathen and M. Sieveking. A bound on solutions of linear integer equalities and inequalities. *Proc. Amer. Math. Soc.*, 72(1):155–158, 1978.