Surminimisation of automata

Victor Marsault

Telecom-ParisTech, 46 rue Barrault 75013 Paris, France

Abstract. We introduce the notion of *surminimisation* of a finite deterministic automaton; it consists in performing a transition relabelling before executing the minimisation and it produces an automaton smaller than a sole minimisation would. While the classical minimisation process preserves the accepted language, the surminimisation process preserves its underlying ordered tree only. Surminimisation induces on languages and on Abstract Rational Numeration Systems (ARNS) a transformation that we call *label reduction.* We prove that all positional numeration systems are label-irreducible and that an ARNS and its label reduction are very close, in the sense that converting the integer representations from one system into the other is done by a simple Mealy machine.

1 Introduction

The classical notion of minimisation (*cf.* for instance [7]) is a transformation of deterministic finite automata and is associated with the automaton equivalence. Two automata are equivalent if they accept the same language L and minimising any automaton accepting L produces the automaton accepting L with the fewest amount of states. Hence, the invariant of minimisation is the accepted language.

In this article, we assume that the alphabet (of every automaton) is equipped with a total order. We then define another automaton transformation called *surminimisation* that produces an automaton with fewer states than the one resulting from a sole minimisation. The invariant of this new transformation is no longer the accepted language, but its underlying ordered tree.

For each state p of a given automaton, the order on the alphabet induces a (total) order on the outgoing transitions of p: a transition is smaller if it is labelled by a smaller letter. The *surminimisation* process consists in two steps. First, it relabels the outgoing transitions of each state p, such that their order is preserved: the smallest transition is relabelled by the letter 0, the second smallest is relabelled by 1, and so on. The second step simply consists in a minimisation.

Surminimisation induces on automata an equivalence relation that we call *T-equivalence*: two automata are T-equivalent if their surminimisations are isomorphic. Moreover, the surminimisation process is idempotent[1], hence each T-equivalence class features a canonical representative computed by surminimising any member of the class.

[1] Two successive surminimisations produce the same result as only one.

We then lift T-equivalence to regular languages (over ordered alphabets). We prove that if two trim automata are equivalent, then their surminimisations are equivalent as well. Hence, we say that two languages are T-equivalent if they are accepted by two T-equivalent (trim) automata and we call label reduction of a regular language L, the language accepted by the surminimisation of any trim automaton accepting L.

A regular language over an ordered alphabet is nothing else than an Abstract Regular Numeration System (ARNS, *cf.* [9]). It consists in ordering a language L by the *radix*, or *genealogical* order: a longer word is always genealogically greater than a shorter word, and the genealogical ordering of two words of equal length coincides with their lexicographical ordering. The representation of an integer n in the ARNS L is then defined as the $(n+1)$-th word of L according to the radix order. The two notions are so close that we use for ARNS's every notion defined for regular languages (such as T-equivalence, label reduction, etc.).

Two T-equivalent ARNS's are very close, in the sense that the function converting one system into the other is realised by a Mealy machine, as stated below.

Theorem 1 *The function that maps the representation of an integer n in an ARNS into the representation of n in an T-equivalent ARNS is realised by a Mealy machine.*

The converse to Theorem 1 is false in the general case. Indeed there exist ARNS's that are not T-equivalent but such that the conversion from one to another is realised by a Mealy machine. We call *locally increasing* a Mealy machine that is locally preserving the order of letters and prove the following statement, a weak converse to Theorem 1.

Theorem 2 *If the function that maps the representation of an integer n in an ARNS into the representation of n in another ARNS is realised by a locally-increasing Mealy machine, then the ARNS's are T-equivalent.*

ARNS's form the most general class of numeration systems. In particular, all (reasonable) *positional numeration systems* (or U-systems, *cf.* [5]) and all *Substitution Numeration Systems* (SNS, *cf.* [3]) are ARNS's.

We first prove that 0^*L is label-irreducible if L denotes the representation language of any positional numeration system. It has quite a significance when comparing the class of ARNS's to the class of label-irreducible ARNS's: the former contains the latter, but 1) brings no supplementary expressive power and 2) contains no additional concrete examples.

We also prove that every prefix-closed ARNS is T-equivalent to some SNS, by using classical transformations from substitutions into automata (*cf.* [11] or even [2]). It is known that every SNS is a prefix-closed ARNS (*cf.* [1]), and the previous results induce a weak converse to this statement: every prefix-closed ARNS is very close to some SNS (in the sense of Theorem 1).

The paper is organised as follows. In Section 2, we define in details the notions of surminimisation, label reduction, etc. The following Section 3 is dedicated to the proof of Theorems 1 and 2. Finally, Section 4 consists in a discussion of label reduction within numeration system theory.

2 Label Reduction and Surminimisation

For every integer k of $\mathbb{N}$, we write $[\![k]\!]$ for the set of the k smallest non-negative integers: $[\![k]\!] = \{0, 1, \ldots, k-1\}$. An *alphabet* is a set of *letters* and in the following **we consider ordered alphabets only**, that is, alphabets (implicitly) equipped with a total order, denoted by $<$. The set $[\![k]\!]$ will be considered both as an integer interval and as a digit alphabet naturally ordered by $0 < 1 < \cdots < (k-1)$.

Automata are directed labelled graphs and in the following **we consider deterministic automata only**, written as a 5-tuple $\mathcal{A} = \langle\, Q, A, \delta, i, F \,\rangle$ where Q is a finite set of *states*; A is a finite (ordered) alphabet; δ is the *transition function*, a **partial** function $Q \times A \to Q$; $i \in Q$ is called the *initial state*; and $F \subseteq Q$ is the set of *final states*. As usual, δ is extended to $Q \times A^*$ by $\delta(p, u\,a) = \delta(\delta(p, u), a)$ and we write $p \xrightarrow[\mathcal{A}]{u} p'$ if $\delta(p, u) = p'$.

The automaton $\mathcal{A}$ is said to be *trim* if each state of $\mathcal{A}$ may reach a final state and is reachable from the initial state. The language accepted by $\mathcal{A}$, denoted by $L(\mathcal{A})$, is the set of the words u such that $\delta(i, u)$ is a final state. Two automata are said *equivalent* if they accept the same language.

We also denote by $\mathsf{Out}_{\mathcal{A}}(p)$ the set of the transitions going out from p. We write $\mathsf{od}(p)$, or more often k_p, for $|\mathsf{Out}_{\mathcal{A}}(p)|$ the out-degree of the state p, and $\mathsf{od}(\mathcal{A}) = \max\{\mathsf{od}(p) \mid p \in Q\}$. For every state p of Q, the order on A induces an order on $\mathsf{Out}_{\mathcal{A}}(p)$; we enumerate $\mathsf{Out}_{\mathcal{A}}(p)$ w.r.t. this order as follows:

$$\forall i \in [\![k_p]\!] \qquad p \xrightarrow{a_i} p_i \quad \text{with } a_0 < a_1 < \cdots < a_{(k_p-1)}\,.$$

We call $(i+1)$*-th transition of* $\mathsf{Out}_{\mathcal{A}}(p)$ the transition $p \xrightarrow{a_i} p_i$, as defined above.

We first define the *label reduction* of an automaton. It consists in relabelling, for each state p, the transitions of $\mathsf{Out}_{\mathcal{A}}(p)$ using the alphabet $[\![k_p]\!]$ and such that the order of $\mathsf{Out}_{\mathcal{A}}(p)$ is preserved. More precisely.

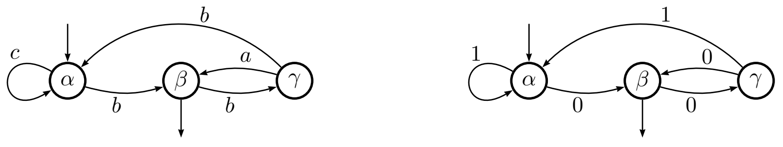
Definition 1. *Let $\mathcal{A} = \langle\, Q, A, \delta, i, F \,\rangle$ be a (deterministic) automaton. We call* label reduction of $\mathcal{A}$*, denoted by* $\mathsf{lred}(\mathcal{A})$ *the automaton:*

$$\mathsf{lred}(\mathcal{A}) \;=\; \langle\; Q,\, [\![\mathit{od}(\mathcal{A})]\!],\, \delta',\, i,\, F \;\rangle \;\;,$$

where δ' is such that, for every state p of Q, if $p \xrightarrow[\mathcal{A}]{a_i} p_i$ is the $(i+1)$-th transition of $\mathsf{Out}_{\mathcal{A}}(p)$ then $p \xrightarrow[\mathsf{lred}(\mathcal{A})]{i} p_i$ is a transition of $\mathsf{lred}(\mathcal{A})$.

Figure 1 shows an automaton $\mathcal{A}_1$ and its label reduction. The label-reduction process commutes with quotient (*cf.* Definition 2, below), as stated at Lemma 1.

Definition 2. *Let $\mathcal{A} = \langle\, Q_{\mathcal{A}}, A, \delta_{\mathcal{A}}, i_{\mathcal{A}}, F_{\mathcal{A}} \,\rangle$ and $\mathcal{M} = \langle\, Q_{\mathcal{M}}, A, \delta_{\mathcal{M}}, i_{\mathcal{M}}, F_{\mathcal{M}} \,\rangle$ be two automata. An* automaton morphism *$\phi : \mathcal{A} \to \mathcal{M}$ is a* surjective *function $Q_{\mathcal{A}} \to Q_{\mathcal{M}}$ meeting the three following conditions.*
1. *$\phi(i_{\mathcal{A}}) = i_{\mathcal{M}}$;*
2. *$p \xrightarrow[\mathcal{A}]{a} p'$ is a transition of $\mathcal{A}$ $\iff$ $\phi(p) \xrightarrow[\mathcal{M}]{a} \phi(p')$ is a transition of $\mathcal{M}$;*
3. *$F_{\mathcal{A}} = \phi^{-1}(F_{\mathcal{M}})$.*



(a) An automaton L_1 (b) $\mathsf{lred}(L_1)$

Fig. 1: Label reduction of an automaton L_1

In this case, $\mathcal{M}$ is called a quotient *of $\mathcal{A}$. If in addition, $\mathcal{M}$ is a quotient of another automaton $\mathcal{B}$, then $\mathcal{A}$ and $\mathcal{B}$ are said* bisimilar. *Every regular language L is canonically associated with a* minimal trim automaton $\mathcal{M}_L$*; it is a quotient of every trim automaton accepting L.*

Lemma 1. *Let $\mathcal{A}$ and $\mathcal{M}$ be two automata. If $\mathcal{M}$ is a quotient of $\mathcal{A}$, then $\mathsf{lred}(\mathcal{M})$ is a quotient of $\mathsf{lred}(\mathcal{A})$.*

Proof. We denote by $\phi : \mathcal{A} \to \mathcal{M}$ the automaton morphism associated with the quotient. Note that the state set of $\mathcal{A}$ and of $\mathsf{lred}(\mathcal{A})$ are identical (and similarly for $\mathcal{M}$ and $\mathsf{lred}(\mathcal{M})$), hence ϕ also maps states of $\mathsf{lred}(\mathcal{A})$ to states of $\mathsf{lred}(\mathcal{M})$; let us prove that ϕ is an automaton morphism from $\mathsf{lred}(\mathcal{A})$ to $\mathsf{lred}(\mathcal{M})$.

Let p be a state of $\mathsf{lred}(\mathcal{A})$. We enumerate the outgoing transitions of p in $\mathcal{A}$ as follows: $\forall i \in [\![k_p]\!]$, $p \xrightarrow{a_i} p_i$ with $a_0 < a_1 < \cdots < a_{(k_p-1)}$, where $k_p = \mathsf{od}(p)$. It follows that the enumeration of the outgoing transitions of $\phi(p)$ in $\mathcal{M}$ are $\forall i \in [\![k_p]\!]$, $\phi(p) \xrightarrow{a_i} \phi(p_i)$ with $a_0 < a_1 < \cdots < a_{(k_p-1)}$.

Hence, from Definition 1, $\mathsf{Out}_{\mathsf{lred}(\mathcal{A})}(p)$ consists of $p \xrightarrow{i} p_i$, $\forall i \in [\![k_p]\!]$. Similarly, $\mathsf{Out}_{\mathsf{lred}(\mathcal{M})}(\phi(p))$ consists of the transitions $\phi(p) \xrightarrow{i} \phi(p_i)$, $\forall i \in [\![k_p]\!]$.

The next proposition, follows almost immediately.

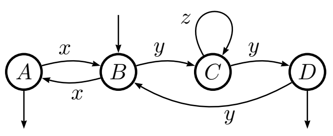
Proposition 1. *Let $\mathcal{A}$ and $\mathcal{B}$ be two trim automata. If $\mathcal{A}$ and $\mathcal{B}$ are equivalent then so are $\mathsf{lred}(\mathcal{A})$ and $\mathsf{lred}(\mathcal{B})$.*

The hypothesis *trim* in Proposition 1 is crucial. Indeed the complete automaton accepting 1^* is equivalent to the trim automaton accepting 1^* whereas their label reductions are not.

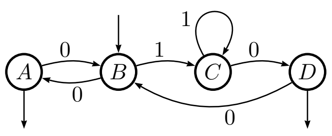
In the following, we consider trim automata only. Hence, Proposition 1 allows to lift *label reduction* to regular languages: the label reduction[2] $\mathsf{lred}(L)$ of a regular language L is the language $L(\mathsf{lred}(\mathcal{A}))$ where $\mathcal{A}$ is any *trim* automaton accepting L. For instance, the label reduction of $((a + b^*)c)^*$ is $(00 + 10^*1 + 2)^*$.

Definition 3 (T-equivalent automata). *Two automata $\mathcal{A}$ and $\mathcal{B}$ are said* tree-equivalent *(or for short* T-equivalent*), denoted by $\mathcal{A} \overset{T}{\sim} \mathcal{B}$, if their label reductions are equivalent: $L(\mathsf{lred}(\mathcal{A})) = L(\mathsf{lred}(\mathcal{B}))$. Similarly, two regular languages L and K are said* T-equivalent *if their label reductions are equal.*

[2] Label reduction may be defined directly on language; *cf.* Remark 2, page 10.



(a) An automaton $\mathcal{A}_2$



(b) $\mathsf{lred}(\mathcal{A}_2)$

Fig. 2: Label reduction of another automaton $\mathcal{A}_2$

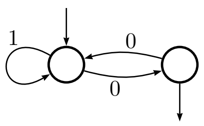


Fig. 3: The surminimisation either of $\mathcal{A}_1$ or of $\mathcal{A}_2$

Figure 2 shows the automaton $\mathcal{A}_2$ and its label reduction. This automaton is T-equivalent to the automaton $\mathcal{A}_1$ (previously shown at Figure 1a). Indeed, their respective label-reductions $\mathsf{lred}(\mathcal{A}_1)$ and $\mathsf{lred}(\mathcal{A}_2)$ are equivalent: they have the same minimisation, shown at Figure 3. This method is a good way to decide whether two automata are T-equivalent, as formalised below.

Definition 4. *We call* surminimisation *of an automaton $\mathcal{A}$, the minimisation of the label reduction of $\mathcal{A}$:* $\mathsf{surmin}(\mathcal{A}) = \mathsf{minim}(\mathsf{lred}(\mathcal{A}))$.

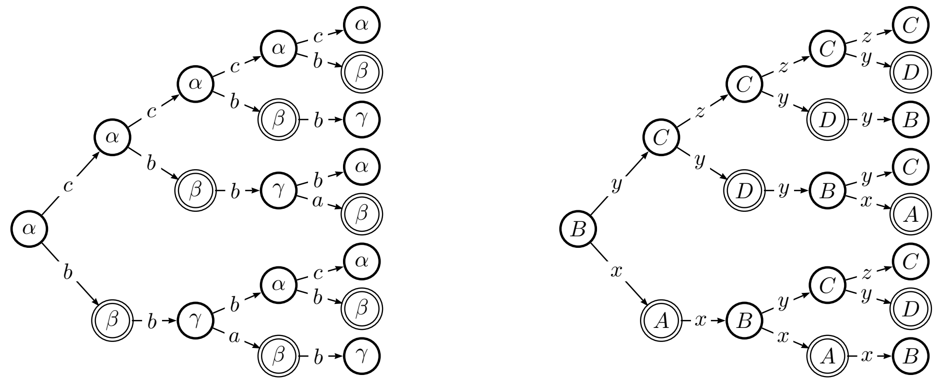
Figure 3 shows the surminimisation either of $\mathcal{A}_1$ or of $\mathcal{A}_2$. The next proposition follows directly from the definitions; it gives both a characterisation and an efficient decision algorithm for T-equivalence.

Proposition 2. *Two automata are T-equivalent if and only if their respective surminimisations are isomorphic.*

Remark 1. Surminimisation (or label reduction) removes the *meaning* of the letters (if there is any) and retains their order only. For instance the language 0^*1^* may be described as 0*'s followed by* 1*'s* while its label reduction is $0^* + 0^*10^*$ that may be described as *words with at most one* 1. In particular, surminimisation also removes the complexity due to an arbitrary choice of letters: for instance the label reduction of the language $L_3 = arbi(trary)^*$ is $0^4\left(0^5\right)^*$ and the label reduction of[3] $\mathsf{Pre}\,(L_3)$ is 0^*; this example highlights that the question of the succinctness of surminimisation is meaningless.

The classical notions of equivalence and minimisation feature a natural invariant: they preserve the accepted language. We have already seen that T-equivalence and surminimisation do not preserve the language; however they feature another invariant: the underlying (ordered) tree.

[3] $\mathsf{Pre}\,(L)$ is the set of prefixes of words of L: $\mathsf{Pre}\,(L) = \{\, u \mid u\,v \in L \text{ for some word } v \,\}$.



(a) The labelled tree T_{L_1} (b) The labelled tree T_{L_2}

Fig. 4: The unfolding of two T-equivalent automata

Definition 5. *A language L over an alphabet A may be represented as an infinite labelled tree (or* infinite *acyclic automaton) as follows: $T_L = (V, A, E, F)$. The vertex set is $V = \mathsf{Pre}(L)$; the edge labels are taken in the alphabet A; the edge set is $E = \{(u, a, ua) \mid ua \in V\} \subseteq V \times A \times V$; the set of final vertices $F = L$.*

If L is a regular language, an isomorphic tree may be obtained by unfolding a (trim) automaton accepting L.

Figure 4 shows the tree representations of $L_1 = L(\mathcal{A}_1)$ and $L_2 = L(\mathcal{A}_2)$; in the figure, a vertex is labelled by the corresponding state of the automaton, and is drawn with a double line if it is final. These two trees, T_{L_1} and T_{L_2}, coincide up to labelling; it is a consequence of the fact that L_1 and L_2 are T-equivalent, as stated in the next Proposition 3. It is a direct consequence of Lemma 2.

Proposition 3. *Let $\mathcal{A}$ and $\mathcal{B}$ be two automata. If $\mathcal{A} \overset{T}{\sim} \mathcal{B}$, then their respective unfoldings differ only by the labellings.*

Lemma 2. *Let $\mathcal{A}$ be an automaton and $\mathcal{M} = \mathsf{surmin}(\mathcal{A})$. Then the respective unfoldings of $\mathcal{A}$ and $\mathcal{M}$ differ only by the labellings.*

Proof (Sketch). Let ϕ be the automaton morphism $\mathsf{lred}(\mathcal{A}) \to \mathcal{M}$. It is also a function from $Q_{\mathcal{A}}$ to $Q_{\mathcal{M}}$ which is **not** an automaton morphism $\mathcal{A} \to \mathcal{M}$: it does not meet the condition 2 of Definition 2. However, ϕ satisfies the next condition.

$$p \xrightarrow[\mathcal{A}]{a} q \text{ is the } (i+1)\text{-th transition of } \mathsf{Out}_{\mathcal{A}}(p) \quad \Longleftrightarrow \quad \phi(p) \xrightarrow[\mathcal{M}]{i} \phi(q)$$

The function ϕ may be extended to a bijection that maps vertices of $T_{L(\mathcal{A})}$ to vertices of $T_{L(\mathcal{M})}$, and satisfies an analogous condition.

3 T-equivalent Languages Define the Same ARNS

An ordered alphabet A induces two orders on words of A^*, the classical *lexicographic order* $<_{\text{lex}}$ and the *radix order* $<_{\text{rad}}$ defined as follows: $u <_{\text{rad}} v$ either if $|u| < |v|$ or if $|u| = |v|$ and $u <_{\text{lex}} v$. Ordering a language L with the radix order defines the *Abstract Numeration System* (ANS, *cf.* [9]) associated with L: every integer n is represented by the $(n+1)$-th word of L in the radix order which is denoted by $\langle n \rangle_L$. If L is a regular language, it defines an Abstract **Regular** Numeration System (ARNS). Let K be another ANS; we call *conversion function from L into K* the function that maps $\langle n \rangle_L$ to $\langle n \rangle_K$, for every integer n.

A Mealy machine is a graph labelled with pair of letters (*cf.* [6]); it is written as a 6-tuple $\mathcal{T} = \langle\, Q, A, B, \tau, i, F \,\rangle$, where Q, A, i and F are defined as in an automaton, B is the *output alphabet* and τ is a function $Q \times A \to B \times Q$, extended as usual to $Q \times A^* \to B^* \times Q$. We write $p \xrightarrow[\mathcal{T}]{u\,|\,v} q$ if $\tau(p,u) = (v,q)$ and the pair $u\,|\,v$ is said to be *accepted by $\mathcal{T}$* if in addition $p = i$ and $q \in F$. The *function realised by $\mathcal{T}$* maps u to v for all pairs $u\,|\,v$ accepted by $\mathcal{T}$. [4]

Let $\mathcal{A}$ and $\mathcal{B}$ be two (trim) automata. We now define a Mealy machine $\mathcal{A} \boxtimes \mathcal{B}$; it is a variant of the well-known automaton product (used for regular-language intersection). The underlying graphs of $\mathcal{A} \boxtimes \mathcal{B}$ and $\mathsf{lred}(\mathcal{A}) \times \mathsf{lred}(\mathcal{B})$ coincide, but the transitions of $\mathcal{A} \boxtimes \mathcal{B}$ are labelled using the labels of $\mathcal{A}$ and $\mathcal{B}$. [5]

Definition 6. *Let $\mathcal{A} = \langle\, Q_{\mathcal{A}}, A, \delta_{\mathcal{A}}, i_{\mathcal{A}}, F_{\mathcal{A}} \,\rangle$ and $\mathcal{B} = \langle\, Q_{\mathcal{B}}, B, \delta_{\mathcal{B}}, i_{\mathcal{B}}, F_{\mathcal{B}} \,\rangle$ be two automata. We denote by $\mathcal{A} \boxtimes \mathcal{B}$ the Mealy machine*

$$\mathcal{A} \boxtimes \mathcal{B} \;=\; \langle\, Q_{\mathcal{A}} \times Q_{\mathcal{B}},\, A,\, B,\, \tau,\, (i_{\mathcal{A}}, i_{\mathcal{B}}),\, F_{\mathcal{A}} \times F_{\mathcal{B}} \,\rangle \;,$$

where the transition function τ is defined as follows. If $p \xrightarrow[\mathcal{A}]{a_i} p_i$ and $q \xrightarrow[\mathcal{B}]{b_i} q_i$ are respectively the $(i+1)$-th transitions of $\mathsf{Out}_{\mathcal{A}}(p)$ and of $\mathsf{Out}_{\mathcal{B}}(q)$, then $\mathcal{A} \boxtimes \mathcal{B}$ features the transition $(p,q) \xrightarrow[\mathcal{A}\boxtimes\mathcal{B}]{a_i\,|\,b_i} (p_i, q_i)$.

We say that a state (p,q) is inconsistent *if either 1) p or q is final but the other is not; or 2) the out-degrees of p and q are not equal: $\mathsf{od}_{\mathcal{A}}(p) \neq \mathsf{od}_{\mathcal{B}}(q)$.*

Figure 5 shows the automaton $\mathcal{A}_1 \boxtimes \mathcal{A}_2$; in the figure, inconsistent states are drawn in dotted lines and their outgoing transitions are omitted and inaccessible but consistent states are drawn in dashed lines. We can now state Theorem 1 under the more precise following form.

Theorem 1. *Let $\mathcal{A}$ and $\mathcal{B}$ be two trim automata. If $\mathcal{A}$ and $\mathcal{B}$ are associated with two T-equivalent ARNS's L and K, then $\mathcal{A} \boxtimes \mathcal{B}$ realises the conversion function from L into K.*

[4] According to transducer terminology, a Mealy machine is a pure sequential and letter-to-letter transducer *cf.* [5, 12]. Mealy machines have the same expressive power as Moore Machines, also called deterministic finite automata with output (DFAO).

[5] In the classical automaton product, transitions are matched using transition labels, hence $\mathsf{lred}(\mathcal{A}) \times \mathsf{lred}(\mathcal{B})$ and $\mathcal{A} \times \mathcal{B}$ have different underlying graphs.

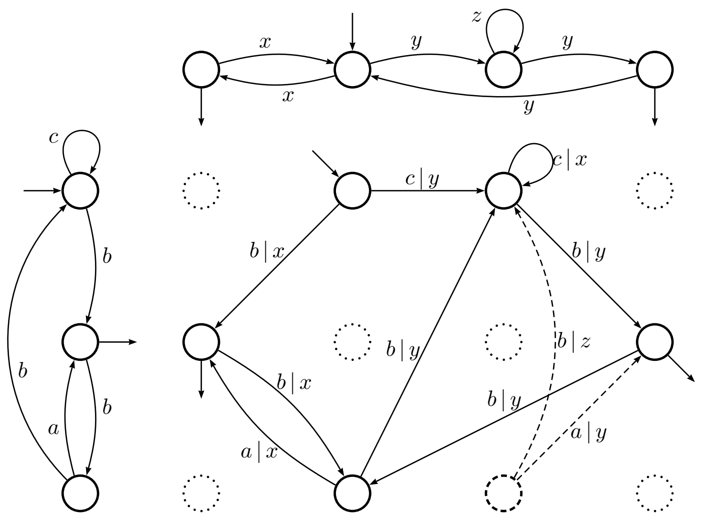


Fig. 5: The Mealy machine $\mathcal{A}_1 \boxtimes \mathcal{A}_2$

The proof of Theorem 1 breaks down into Lemmas 3 and 4. First, we give a few properties following directly from Definition 6.

Property 1. Let $\mathcal{A}$, $\mathcal{B}$ be two automata and $u \,|\, v$, $u' \,|\, v'$ be accepted by $\mathcal{A} \boxtimes \mathcal{B}$.

a. $u = u' \Leftrightarrow v = v'$.
b. $u <_{\text{rad}} u' \Leftrightarrow v <_{\text{rad}} v'$.
c. Let $(p, q) \xrightarrow{a \,|\, b} (p', q')$ be a transition of $\mathcal{A} \boxtimes \mathcal{B}$. Then $p \xrightarrow{a} p'$ is a transition of $\mathcal{A}$ and $q \xrightarrow{b} q'$ is a transition of $\mathcal{B}$.
d. Let $p \xrightarrow{a} p'$ be a transition of $\mathcal{A}$ and q be a state of $\mathcal{B}$. If (p, q) is consistent, then $(p, q) \xrightarrow{a \,|\, b} (p', q')$ is a transition of $\mathcal{A} \boxtimes \mathcal{B}$, for some q' and b. [6]

Lemma 3. *Let $\mathcal{A}$ and $\mathcal{B}$ be two automata. If $\mathcal{A} \boxtimes \mathcal{B}$ has no inconsistent accessible states, then it realises the conversion function from $L(\mathcal{A})$ into $L(\mathcal{B})$*

Proof. Let us denote by $\mathcal{A}'$ the trim of the input automaton of $\mathcal{A} \boxtimes \mathcal{B}$. If $\mathcal{A} \boxtimes \mathcal{B}$ has no inconsistent accessible states then it follows from Properties 1.c and 1.d that $\mathcal{A}$ is a quotient of $\mathcal{A}'$ (through the projection $(p, q) \mapsto p$). It follows that the input language of $\mathcal{A} \boxtimes \mathcal{B}$ is $L(\mathcal{A})$; symmetrically, the output language of $\mathcal{A} \boxtimes \mathcal{B}$ is $L(\mathcal{B})$. Since $\mathcal{A} \boxtimes \mathcal{B}$ realises a bijection (from Property 1.a) and preserves the order (from Property 1.b), it follows that $\mathcal{A} \boxtimes \mathcal{B}$ maps $\langle n \rangle_{\mathcal{A}}$ to $\langle n \rangle_{\mathcal{B}}$.

Lemma 4. *Let $\mathcal{A}$ and $\mathcal{B}$ be two automata. If $\mathcal{A} \overset{T}{\rightsquigarrow} \mathcal{B}$, then every inconsistent state of $\mathcal{A} \boxtimes \mathcal{B}$ is not accessible.*

Proof. Since they are T-equivalent, $\mathcal{A}$ and $\mathcal{B}$ have the same surminimisation, denoted by $\mathcal{M}$; it is a quotient both of $\mathsf{Ired}(\mathcal{A})$ and of $\mathsf{Ired}(\mathcal{B})$ realised by the

[6] For concision, we omitted the symmetrical statement.

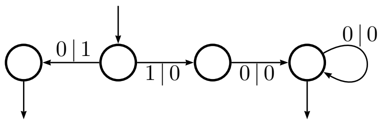


Fig. 6: A Mealy machine which is not locally increasing

automaton morphisms $\phi : \mathsf{lred}(\mathcal{A}) \to \mathcal{M}$ and $\psi : \mathsf{lred}(\mathcal{B}) \to \mathcal{M}$. The proof of the next claim consists in an induction over a traversal of $\mathcal{A} \boxtimes \mathcal{B}$ and is omitted here.

Claim. *Every accessible state* (p, q) *of* $\mathcal{A} \boxtimes \mathcal{B}$ *meets the condition* $\phi(p) = \psi(q)$.

Let (p, q) be an accessible state of $\mathcal{A} \boxtimes \mathcal{B}$. It follows that $\phi(p) = \psi(q)$ (from the claim), hence that p and q are both final or both non-final and that p and q have the same amount of outgoing transitions. Hence (p, q) is not an inconsistent state.

Theorem 1 follows directly from Lemmas 3 and 4. However, its converse is false in the general case: the Mealy machine shown at Figure 6 realises the conversion from $0+10^+$ into $1+00^+$, two distinct and label-irreducible languages hence not T-equivalent.

We say that a Mealy machine $\mathcal{T}$ is *locally increasing* if it locally preserves the order of labels or, more formally, if it satisfies the following condition.

$$\text{For every pair of transitions of } \mathcal{T} \quad \begin{array}{c} p \xrightarrow[\mathcal{T}]{a\,|\,b} q \\ p \xrightarrow[\mathcal{T}]{c\,|\,d} q' \end{array} \qquad a < c \;\Leftrightarrow\; b < d \tag{1}$$

For instance, the Mealy machine $\mathcal{A} \boxtimes \mathcal{B}$ is always locally increasing whereas the one shown at Figure 6 is not: the two outgoing transitions of the initial state reverse the order of the letters.

Theorem 2. *Let* $\mathcal{T}$ *be a locally increasing Mealy machine,* $\mathcal{A}$ *and* $\mathcal{B}$ *its respective input and output automata. Then* $\mathcal{A}$ *and* $\mathcal{B}$ *are T-equivalent.*

Proof. Note that $\mathcal{A}$, $\mathcal{B}$, $\mathcal{T}$ have the same state set and that since $\mathcal{T}$ is locally increasing, then $\mathcal{B}$ is deterministic. Let p be a state of $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{T}$. We enumerate the outgoing transitions of p in $\mathcal{T}$: $\forall i \in [\![k_p]\!] \quad p \xrightarrow[\mathcal{T}]{a_i\,|\,b_i} p_i$ where $k_p = \mathsf{od}_{\mathcal{T}}(p)$ and $a_0 < a_1 < \cdots < a_{k_p-1}$; since $\mathcal{T}$ is locally increasing, then $b_0 < b_1 < \cdots < b_{k_p-1}$.

We fix i in $[\![k_p]\!]$. The transitions $p \xrightarrow[\mathcal{A}]{a_i} p_i$ and $p \xrightarrow[\mathcal{B}]{b_i} p_i$ are respectively the $(i+1)$-th transitions of $\mathsf{Out}_{\mathcal{A}}(p)$ and $\mathsf{Out}_{\mathcal{B}}(p)$. It follows that both transitions are relabelled by the same digit i in $\mathsf{lred}(\mathcal{A})$ and $\mathsf{lred}(\mathcal{B})$, hence coincide. Since $\mathcal{A}$ and $\mathcal{B}$ differ only by their transition labels, $\mathsf{lred}(\mathcal{A})$ and $\mathsf{lred}(\mathcal{B})$ are isomorphic.

4 Label Reduction Within Numeration System Theory

We briefly recall here basic definitions and notations for positional numeration system; see for instance Section 2.3.3 of [5] for more details. A *basis* is a strictly increasing sequence of integers $(U_i)_{i\in\mathbb{N}}$ with $U_0 = 1$ defining the positional numeration system U. The U-evaluation function π_U maps a finite word $d_k\, d_{k-1} \cdots d_0$ over a digit alphabet to the integer $\pi_U(d_k\, d_{k-1} \cdots d_0) = \sum_{i=0}^{k} d_i\, U_i$. The Rényi greedy algorithm (*cf.* for instance [10, Chapter 7]) computes a word whose evaluation is n; it is called *the U-representation of n* and is denoted by $\langle n\rangle_U$. We also denote by $L(U)$ the language $L(U) = \{\langle n\rangle_U \mid n \in \mathbb{N}\}$.

In the following, we always assume that the ratio U_{n+1}/U_n is bounded by an integer constant $M = \sup\{\, \lceil U_{n+1}/U_n \rceil \mid n \in \mathbb{N}\,\}$, in which case the digits of each U-representation belong to the alphabet $A_U = [\![M]\!]$. The next two classical propositions follow.

Proposition 4 ([5, Proposition 2.3.44]). *Let u be a word of ${A_U}^*$. If u does not start with the letter 0 then $u \leq_{rad} \langle n\rangle_U$, where $n = \pi_U(u)$.*

Proposition 5 ([5, Proposition 2.3.45]). *Let n and m be two positive integers. Then $n < m$ if and only if $\langle n\rangle_U <_{rad} \langle m\rangle_U$.*

The next proposition (together with the remark following it) is the main result from this section.

Proposition 6. *Let U be a positional numeration system. If $L(U)$ is a regular language, then $0^*L(U)$ is label-irreducible.* [7]

Proof. For concision, we write $\langle\ \rangle$ and $\pi(\ )$ instead of $\langle\ \rangle_U$ and $\pi_U(\ )$ in this proof. The whole statement follows from the next claim.

Claim. *Let m be an integer u, v be two words of A_U^* and $(d+1)$ be a positive digit. If $\langle m\rangle = u(d+1)v$ and $(u,d) \neq (\epsilon, 0)$, then there exists an integer n such that $\langle n\rangle = udv'$, for some v'.*

Proof of the Claim. Without loss of generality, we may assume that v is the smallest word in the radix order such that $u(d+1)v$ is the representation of an integer (Assumption $(*)$). Let $k = |v|$, we denote by n the value of $u\,d\,v$: that is $n = \pi(u\,d\,v) = (m - U_k)$. Note that (since it is a prefix of $\langle m\rangle$,) u may not start with the letter zero; since $(u,d) \neq (\epsilon, 0)$, it follows that udv does not start with the letter 0 either and applying Propositions 4 and 5 yields the two following inequations: $u\,d\,v \ \leq_{\text{rad}}\ \langle n\rangle\ <_{\text{rad}}\ u\,(d+1)\,v$.
It follows that $\langle n\rangle$ is of one of the three following forms.

- $\langle n\rangle = u\,d\,w$ with $v <_{\text{rad}} w$ and $\pi(v) = \pi(w)$. It follows that $u\,(d+1)\,v <_{\text{rad}} u\,(d+1)\,w$ and that $\pi(u\,(d+1)\,w) = m$, hence the representation of m cannot be equal to $u\,(d+1)\,v$, a contradiction to Proposition 4.
- $\langle n\rangle = u(d+1)w$ for some $w <_{\text{rad}} v$, a contradiction to Assumption $(*)$.
- $\langle n\rangle = u\,d\,v$ yielding the proof of the claim.

Remark 2. Proposition 6 could be made substantially stronger if label reduction were defined directly on languages (ie. independently of automata). The label reduction of a language would be defined as $\mathsf{lred}(L) = \{f_L(u) \mid u \in L\}$ with

$$f_L(\epsilon) = \epsilon$$
$$f_L(u\,b) = f_L(u)\, g_L(u,b) \qquad \text{where } g_L(u,b) = |\{ua \mid a < b \text{ and } ua \in \mathsf{Pre}\,(L)\}| \ .$$

[7] A language is said *label-irreducible* if it is equal to its label reduction.

A language L is *label-irreducible* if $\forall ub,\ \ ub \in \mathsf{Pre}\,(L) \ \Rightarrow\ \forall a < b,\ ua \in \mathsf{Pre}\,(L)$. The statement *every positional numeration system is label-irreducible* is then an immediate consequence of the Claim of the previous proof.

Remark 3. Let us compare the classes of ARNS's and label-irreducible ARNS's.
- Every ARNS is T-equivalent to some label-irreducible ARNS, hence from Theorem 1, one may be converted into the other by means of a Mealy machine. It follows that both systems will share the same properties
- Within an T-equivalence class, the unique label-irreducible ARNS is associated with an automaton with the smallest amount of states.
- All concrete numeration systems seem to be label-irreducible. [8]

In the remainder of this section, we prove that every T-equivalence class contains a *Substitution Numeration System* (SNS, *cf.* [3]). It is known that an SNS is a particular ARNS (*cf.* [1]) and we use this result to give here a very brief description of SNS's as ARNS's.

Let X be an alphabet and let $\sigma : X^* \to X^*$ be a monoid morphism. We assume that σ is *prolongable on* $x \in X$, that is, 1) $\sigma(x)$ starts with an x and 2) $\lim_{n\to\infty}(|\sigma^n(x)|) = \infty$. In the following, we manipulate the alphabet B_σ (defined below) whose *letters* are *words* over the alphabet X. If u or $x_0x_1\cdots x_k$ denotes a word of X^*, the corresponding letter of B_σ is denoted by $[u]$ or $[x_0x_1\cdots x_k]$. $B_\sigma = \{\, [u] \mid u \text{ is a strict prefix of } \sigma(y) \text{ for some } y \in A \,\}$.

The *prefix automaton* $\mathcal{A}_\sigma = \langle\, X, B_\sigma, \delta, x, X \,\rangle$ is defined as follows. Its state set is X (*ie.* the alphabet of σ), its initial state is x and all states are accepting. The transition function δ is defined such that $\mathcal{A}_\sigma$ features the transition $y \xrightarrow[\mathcal{A}_\sigma]{[u]} z$ if and only if uz is a prefix of $\sigma(y)$. The SNS σ is then the ARNS L, where L consists of the words of $L(\mathcal{A}_\sigma)$ that do not start with the letter $[\epsilon]$.

If every ARNS is not necessarily an SNS, it is pretty close to be the case, as stated by the Corollary 1 of the next proposition. The proof of this statement is omitted here, and consists in a thorough examination of classical transformations from automaton to substitution (*cf.* [11, 8]).

Proposition 7. *Every prefix-closed ARNS is T-equivalent to an SNS.*

Corollary 1. *Every prefix-closed ARNS can be converted into an SNS through a Mealy machine.*

5 Conclusion

We introduced the notion of surminimisation of automata, a transformation producing an automata smaller than the one resulting from the classical minimisation. While minimisation preserves the language (that is, a labelled tree), surminimisation preserves the underlying unlabelled tree only.

[8] We call here *concrete* the numeration systems that may be defined by an evaluation function, by opposition to those defined by their representation language.

To the best of our knowledge, the only label-reducible concrete numeration systems are the *rational base numeration systems* (*cf.* Section 2.5 of [5]) and, even in this case, it happens to exist a variant which is label-irreducible (*cf.* [4]).

Surminimisation induces on Abstract Regular Numeration Systems (ARNS) a transformation, called label reduction, and an equivalence relation; each equivalence class features a canonical representative: the label reduction of any element of the class. We proved that members of the same equivalence class are essentially the same (*ie.* may be converted from one into another by a Mealy machine) and, conversely, that if the conversion from one ARNS into another is realised by a locally increasing Mealy machine, then the ARNS's are T-equivalent.

Moreover, a simple verification yields that all positional numeration systems are label-irreducible. In summary, label reduction allows to simplify ARNS's without excluding any concrete cases.

Acknowledgements The author thanks Michel Rigo and Émilie Charlier for their invitation to Liège and the discussions he had with them about this work.

References

1. Valérie Berthé and Michel Rigo. Odometers on regular languages. *Theory Comput. Syst.*, 40(1):1–31, 2007.
2. Alan Cobham. Uniform tag sequences. *Math. Systems Theory*, 6:164–192, 1972.
3. Jean-Marie Dumont and Alain Thomas. Digital sum problems and substitutions on a finite alphabet. *Journal of Number Theory*, 39(3):351–366, 1991.
4. Christiane Frougny and Karel Klouda. Rational base number systems for p-adic numbers. *RAIRO - Theor. Inf. and Applic.*, 46(1):87–106, 2012.
5. Christiane Frougny and Jacques Sakarovitch. Number representation and finite automata. in *Combinatorics, Automata and Number Theory*, V. Berthé, M. Rigo (Eds), Encyclopedia of Mathematics and its Applications 135, Cambridge Univ. Press (2010) 34–107.
6. Georges H. Mealy. A method for synthesizing sequential circuits. *Bell Syst. Tech. J.*, 34:1045–1079, 1955.
7. John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley, 1979.
8. Pierre Lecomte and Michel Rigo. Abstract numeration systems. in *Combinatorics, Automata and Number Theory*, V. Berthé, M. Rigo (Eds), Encyclopedia of Mathematics and its Applications 135, Cambridge Univ. Press (2010) 108–162.
9. Pierre Lecomte and Michel Rigo. Numeration systems on a regular language. *Theory Comput. Syst.*, 34:27–44, 2001.
10. M. Lothaire. *Algebraic Combinatorics on Words.* Cambridge University Press, 2002.
11. Michel Rigo and Arnaud Maes. More on generalized automatic sequences. *Journal of Automata, Languages and Combinatorics*, 7(3):351–376, 2002.
12. Jacques Sakarovitch. *Eléments de théorie des automates.* Vuibert, 2003. Corrected English translation: *Elements of Automata Theory*, Cambridge University Press, 2009.