Synchronized Product of Linear Bounded Machines

Teodor KNAPIK and Étienne PAYET

IREMIA, Université de La Réunion, BP 7151, 97715 Saint Denis Messag. Cedex 9, France
{knapik,epayet}@univ-reunion.fr

Abstract. This paper introduces a class of graphs associated to linear bounded machines. It is shown that this class is closed, up to observational equivalence, under synchronized product. The first–order theory of these graphs is investegated and shown to be undecidable. The latter result extends to any logic in which the existence of sinks may be stated.

1 Introduction

Finite transition systems together with their synchronized product define a simple and elegant theoretical framework for specification and verification of systems of communicating processes. This framework is known as the Arnold–Nivat approach [2, 16]. A number of equivalent approaches as *e.g.* CCS [14] or Meije [3] and decision procedures for various logics (see *e.g.* [8] and [11]) have provided grounds for the model checking (see *e.g.* [12] or [13]). In spite of encouraging time–complexity results in this area, the approaches based on finite transition systems encounter space–complexity problems. To face up these problems, many compression–like techniques, as *e.g.* binary decision diagrams, have been developed [4].

The problem of storage space for a representation of a process may be overcome using (possibly) infinite transition systems. Among these, the best known are the pushdown transition systems *viz* the transition graphs of pushdown machines. Since the result of [15] about the decidability of the monadic second–order logic of these graphs, more general families of graphs that enjoy this decidability property have been discovered in terms of several descriptions (see [7] and [5]).

Although the latter approaches provide an increased expressive power, they did not give rise to an important development of the "infinite model checking" theory and practice. In the authors' opinion, this is due to the fact that interacting processes cannot be described within these approaches, because the classes of graphs of [5], [7] and [15] are not closed under the synchronized product.

The classes of finite (resp. pushdown) transition systems are naturally related to rational (resp. context–free) languages. Both classes have been already investegated, the former more deeply than the latter. But almost nothing is known about graphs related to the next level of the Chomsky hierarchy, namely the context–sensitive languages. The present paper goes into this direction with the emphasis on the synchronized product.

We consider a multi–tape linear bounded machine model with a single, read–only input tape and we define the transition graphs of such devices. We study two transformations on these machines. The first–one is similar to the usual simulation of a multi–tape Turing machine by a single–tape one. The second–one consists in a construction of a multi–tape machine that behaves like several communicating single–tape machines. In both cases, we show that the transformations preserve observational equivalence of associated graphs (this is the main difference with the usual treatment where isomorphisms are considered.) The composition of both transformations allows to establish that the class of graphs of linear bounded machines is closed, up to observational equivalence, under synchronized product.

Unfortunately, as established in the paper, the first–order theory of the graphs of linear bounded machines is not decidable.

2 Preliminaries

Throughout this paper, the empty word is written ε and, if $n \in \mathbb{N}$, $[n]$ stands for the set $\{1, \ldots, n\}$ (with $[0] = \varnothing$).

2.1 Rooted graphs, Their synchronized Product and Their ε-Equivalence

A simple directed edge–labelled graph $\mathcal{G}$ (or more simply a *graph*) over C is a set of *edges*, *i.e.* a subset of $D \times C \times D$ where D is an arbitrary set, the elements of which are called the *vertices* of $\mathcal{G}$ and C is an alphabet possibly extended with the empty word. Given d and d' in D, an edge from d to d' labelled by $c \in C$ is written $d \xrightarrow{c} d'$. Thus, $\xrightarrow{c}$ is a binary relation on D for each $c \in C$. A (finite) *path* in $\mathcal{G}$ from d to d' is a sequence of edges of the form $d_0 \xrightarrow{c_1} d_1, \ldots, d_{n-1} \xrightarrow{c_n} d_n$ such that $d_0 = d$ and $d_n = d'$. The word $w = c_1 \ldots c_n$ is then the *label* of the path. In this case, we may write $d \xrightarrow{w} d'$. We shall constantly consider graphs, the vertices of which are all accessible from some distinguished vertex. Thus, a graph $\mathcal{G}$ is said to be *rooted on a vertex* d if there exists a path from d to each vertex of $\mathcal{G}$. The maximal subgraph of $\mathcal{G}$ that is rooted on a vertex e is written $\mathcal{G}[e]$.

Synchronized Product of Rooted Graphs. The synchronized product of graphs has been introduced by Arnold and Nivat [2, 16]. It is an essential part of the semantic of interacting processes. For more material, the reader may refer to [1]. We introduce here a definition that is a variant of Arnold and Nivat's one. Indeed, in the scope of this paper, we need a product that takes as entry some rooted graphs and returns a rooted graph as well. Given n alphabets $C_1, \ldots, C_n$ possibly extended with ε, a *synchronization constraint* $\mathcal{C}$ over $C_1, \ldots, C_n$ is a subset of $\prod_{i\in[n]} C_i$. Let $\mathcal{G}_1[d_1], \ldots, \mathcal{G}_n[d_n]$ be some rooted graphs over $C_1, \ldots, C_n$. The *synchronized product* of $\mathcal{G}_1[d_1], \ldots, \mathcal{G}_n[d_n]$ with respect to $\mathcal{C}$, written $\prod^{\mathcal{C}}_{i\in[n]} \mathcal{G}_i[d_i]$, is the graph $\mathcal{G}[(d_1, \ldots, d_n)]$ where

$$\mathcal{G} = \left\{ e \xrightarrow{c} e' \;\middle|\; c \in \mathcal{C} \text{ and } \forall i \in [n],\ e_i \xrightarrow{c_i} e'_i \in \mathcal{G}_i[d_i] \right\} .$$

In this definition, e_i (resp. c_i, resp. e'_i) stands for the i^{th} coordinate of tuple e (resp. c, resp. e').

ε–Equivalence of Rooted Graphs. In the following definition, $\overset{\varepsilon}{\dashrightarrow}$ stands for the reflexive–transitive closure of $\xrightarrow{\varepsilon}$ and $\overset{a}{\dashrightarrow} = \overset{\varepsilon}{\dashrightarrow} \circ \xrightarrow{a} \circ \overset{\varepsilon}{\dashrightarrow}$ for any a in alphabet Σ. Let $\mathcal{G}_1[e_1]$ and $\mathcal{G}_2[e_2]$ be two rooted graphs over $\Sigma \cup \{\varepsilon\}$ with set of vertices D_1 and D_2 respectively. These graphs are said to be *ε–equivalent* if there exists a relation $\leftrightsquigarrow \subseteq D_1 \times D_2$ such that:

1. $\mathrm{Dom}(\leftrightsquigarrow) = D_1$ and $\mathrm{Ran}(\leftrightsquigarrow) = D_2$;
2. $e_1 \leftrightsquigarrow e_2$;
3. for each a in Σ, each path $d_1 \overset{a}{\dashrightarrow} d'_1$ in $\mathcal{G}_1[e_1]$ and each vertex d_2 of $\mathcal{G}_2[e_2]$ such that $d_1 \leftrightsquigarrow d_2$, there exists a vertex d'_2 in $\mathcal{G}_2[e_2]$ such that $d'_1 \leftrightsquigarrow d'_2$ and $d_2 \overset{a}{\dashrightarrow} d'_2$ is a path of $\mathcal{G}_2[e_2]$;
4. for each a in Σ, each path $d_2 \overset{a}{\dashrightarrow} d'_2$ in $\mathcal{G}_2[e_2]$ and each vertex d_1 of $\mathcal{G}_1[e_1]$ such that $d_1 \leftrightsquigarrow d_2$, there exists a vertex d'_1 in $\mathcal{G}_1[e_1]$ such that $d'_1 \leftrightsquigarrow d'_2$ and $d_1 \overset{a}{\dashrightarrow} d'_1$ is a path of $\mathcal{G}_1[e_1]$.

It should be noted that ε–equivalence is an observational equivalence (also called weak bisimulation) as defined by Milner in [14] if one considers ε to be an invisible event.

2.2 Linear Bounded Machines and Their Graphs

We use a definition of a Linear Bounded Machine[1] (*LBM* for short) that is slightly different from standard ones and has a flavour of a *Chaitin computer* [6]. Our motivation does not rely on any languages theory aspect. Actually, we are interested in LBM's as an approach for modelling process behaviour that is defined as a graph associated to an LBM. For that matter, we need LBM's such that the motion of the input tape head is one–way, from the left to the right. When this head moves from a cell c to the right neighboor of c, the machine reads the content of c. Moreover, the work tapes are infinite to the left and to the right and, in addition to the usual moves (left and right), each work tape head may stay at its place.

Formally, a k work tapes LBM $\mathcal{L}$ is a tuple $(Q, \Sigma, \Gamma_1, \ldots, \Gamma_k, \delta, q_0)$ where Q is the finite set of *states*, Σ is the *input alphabet*, $\Gamma_1, \ldots, \Gamma_k$ are the *work tapes alphabets*, q_0 is the *initial state*, and δ is the set of *transitions*:

$$\delta \subseteq Q \times \Sigma \cup \{\varepsilon\} \times \Gamma_1 \times \cdots \times \Gamma_k \times \Gamma_1 \times \{\blacktriangleleft, \blacktriangleright, \blacksquare\} \times \cdots \times \Gamma_k \times \{\blacktriangleleft, \blacktriangleright, \blacksquare\} \times Q$$

where $\blacktriangleleft$ (resp. $\blacktriangleright$ and $\blacksquare$) symbolizes a move to the left (resp. a move to the right and no move) and an input ε represents the special case where the input tape head of $\mathcal{L}$ does not move and does not read any character. We assume that the blank character, written $\square$, belongs to each Γ_i.

[1] A linear bounded machine is a linear bounded automaton with no final state.

An *internal configuration* $(\mu_1 q\nu_1, \ldots, \mu_k q\nu_k)$ of $\mathcal{L}$ is an element of the set ${\Gamma_1}^*.Q.{\Gamma_1}^* \times \cdots \times {\Gamma_k}^*.Q.{\Gamma_k}^*$. This encodes the description of $\mathcal{L}$ at a time as follows: q is the current state, for all i in $[k]$, $\mu_i\nu_i$ is the content of the i^{th} work tape from the leftmost nonblank character to the rightmost nonblank character and for all i in $[k]$, the head of the i^{th} work tape is reading ν_i's first character or $\square$ if ν_i is empty. Notice that for all i in $[k]$, both μ_i or ν_i may contain $\square$. An internal configuration, every coordinate of which equals q_0, is called *initial configuration* of $\mathcal{L}$.

To every LBM $\mathcal{L} = (Q, \Sigma, \Gamma_1, \ldots, \Gamma_k, \delta, q_0)$ we associate the graph $\mathcal{G}_{\mathcal{L}}[\iota]$ where ι is the initial configuration of $\mathcal{L}$ and $\mathcal{G}_{\mathcal{L}}$ is defined as follows. The vertices of $\mathcal{G}_{\mathcal{L}}$ are all the internal configurations of $\mathcal{L}$ and the labels of $\mathcal{G}_{\mathcal{L}}$ belong to the set $\Sigma \cup \{\varepsilon\}$. Moreover, $(\mu_1 q\nu_1, \ldots, \mu_k q\nu_k) \xrightarrow{c} (\mu'_1 q'\nu'_1, \ldots, \mu'_k q'\nu'_k)$ is an edge of $\mathcal{G}_{\mathcal{L}}$ if and only if for each $i \in [k]$, there exist $X_i, Y_i \in \Gamma_i$ and $\blacklozenge_i \in \{\blacktriangleleft, \blacktriangleright, \blacksquare\}$ such that $(q, c, X_1, \ldots, X_k, Y_1, \blacklozenge_1, \ldots, Y_k, \blacklozenge_k, q') \in \delta$ and either $\blacklozenge_i = \blacktriangleleft$ and one of the following holds:

- $\exists \alpha_i, \beta_i \in {\Gamma_i}^*$, $\exists Z_i \in \Gamma_i$, $\mu_i = \alpha_i Z_i$, $\nu_i = X_i\beta_i$, $\mu'_i = \alpha_i$ and, if $Z_iY_i\beta_i \notin \{\square\}^*$, then $\nu'_i = Z_iY_i\beta_i$, else $\nu'_i = \varepsilon$;
- $\mu_i = \varepsilon$, $\exists \alpha_i \in {\Gamma_i}^*$, $\nu_i = X_i\alpha_i$, $\mu'_i = \varepsilon$ and, if $Y_i\alpha_i \notin \{\square\}^*$, then $\nu'_i = \square Y_i\alpha_i$, else $\nu'_i = \varepsilon$;
- $X_i = \square$, $\mu_i = \varepsilon$, $\nu_i = \varepsilon$, $\mu'_i = \varepsilon$ and, if $Y_i \neq \square$, then $\nu'_i = \square Y_i$, else $\nu'_i = \varepsilon$;

or $\blacklozenge_i = \blacktriangleright$ and one of the following holds:

- $\exists \alpha_i \in {\Gamma_i}^*$, $\nu_i = X_i\alpha_i$, $\nu'_i = \alpha_i$ and, if $\mu_iY_i \notin \{\square\}^*$, then $\mu'_i = \mu_iY_i$ else $\mu'_i = \varepsilon$;
- $X_i = \square$, $\nu_i = \varepsilon$, $\nu'_i = \varepsilon$, and if $\mu_iY_i \notin \{\square\}^*$, then $\mu'_i = \mu_iY_i$, else $\mu'_i = \varepsilon$;

or $\blacklozenge_i = \blacksquare$ and one of the following holds:

- $\exists \alpha_i \in {\Gamma_i}^*$, $\nu_i = X_i\alpha_i$, $\mu'_i = \mu_i$ and, if $Y_i\alpha_i \notin \{\square\}^*$, then $\nu'_i = Y_i\alpha_i$, else $\nu'_i = \varepsilon$;
- $X_i = \square$, $\nu_i = \varepsilon$, $\mu'_i = \mu_i$ and, if $Y_i \neq \square$, then $\nu'_i = Y_i$, else $\nu'_i = \varepsilon$.

We say that two LBM's are ε*–equivalent* if the associated graphs are so.

3 Multi–Work Tape LBM's

In this section, it is established that every LBM with k work tapes is ε–equivalent to an LBM with one work tape only. This is done by providing a construction of the one work tape LBM from the k work tapes one.

It is important to note that this result is not necessarily a consequence of the fact that the languages recognized by the linear bounded automata with n work tapes are the same as those recognized by the linear bounded automata with m work tapes, for all n and m in $\mathbb{N}$ (a linear bounded automaton is an LBM provided with a set of final states). When two kinds of devices accept the same family of languages then the classes of graphs generated by both devices need not to be the same up to observational equivalence. For instance, the graphs of pushdown automata are not, in general, ε–equivalent to the graphs of realtime pushdown automata whereas both kinds of automata accept exactly the family of context–free languages.

Let $\mathcal{L} = (Q, \Sigma, \Gamma_1, \ldots, \Gamma_k, \delta, q_0)$ be a k work tapes LBM. The one work tape LBM $\mathcal{L}'$, that is ε–equivalent to $\mathcal{L}$, is constructed in the following way. On one hand, the work tape inscription of $\mathcal{L}'$ consists of the concatenation of the inscriptions of all the work tapes of $\mathcal{L}$ separated by *delimiters*. On the other hand, the motion of the k work tape heads of $\mathcal{L}$ is simulated by the single work tape head of $\mathcal{L}'$ by means of *head marks*.

More precisely, we introduce the following new characters: for each i in $[k+1]$, a delimiter, written $\%_i$, and, for each work tape character X, a corresponding head mark, written $\dot{X}$. In the sequel, for all i in $[k]$, H_i denotes the set $\{\dot{X} \mid X \in \Gamma_i\}$. An internal configuration $(\mu_1 q \nu_1, \ldots, \mu_k q \nu_k)$ of $\mathcal{L}$ is simulated by an inscription $\%_1\ \alpha_1 \dot{X}_1 \beta_1\ \%_2\ \ldots\ \%_k\ \alpha_k \dot{X}_k \beta_k\ \%_{k+1}$ on the work tape of $\mathcal{L}'$. This inscription is such that, for each i in $[k]$, $\nu_i \neq \varepsilon \Rightarrow (\exists \gamma_i \in \{\square\}^*,\ \nu_i \gamma_i = X_i \beta_i)$, $\nu_i = \varepsilon \Rightarrow (X_i = \square \text{ and } \beta_i \in \{\square\}^*)$ and $\exists \gamma_i \in \{\square\}^*,\ \gamma_i \mu_i = \alpha_i$.

The LBM $\mathcal{L}'$ simulates $\mathcal{L}$ in the following way. First, in order to simulate the initial configuration of $\mathcal{L}$, $\mathcal{L}'$ copies the word $\%_1\ \dot{\square}\ \%_2\ \ldots\ \%_k\ \dot{\square}\ \%_{k+1}$ on its work tape. Let δ_{copy} denote the set of transitions of $\mathcal{L}'$ performing this copy, Q_{copy} denote the set of states involved in δ_{copy} and suppose that after these operations, $\mathcal{L}'$ switches to state q_0. A computation for $\mathcal{L}$ consists in overprinting a character Y_i on a character X_i on each work tape i and possibly moving the head of i to the left or to the right. This is simulated by $\mathcal{L}'$ in the following way. If $\mathcal{L}$ does not move the head of i, then $\mathcal{L}'$ overprints $\dot{Y}_i$ on $\dot{X}_i$. If $\mathcal{L}$ moves the head of i to the left (resp. right), then $\mathcal{L}'$ overprints Y_i on $\dot{X}_i$ and, if Z is the character written to the left (resp. right) of Y_i, it overprints $\dot{Z}$ on Z. Overprintings on each i^{th} portion of the work tape of $\mathcal{L}'$ (*i.e.* the part of this tape corresponding to the i^{th} work tape of $\mathcal{L}$) that are due to a transition $t \in \delta$ are performed by means of state $q_{t,i}$. Moreover, in order to perform each overprinting due to t, the work tape head of $\mathcal{L}'$ must be able to move from the i^{th} portion of the work tape to the next one to the right (if it exists); this is done by means of the state $m_{t,i}$; when the last overprinting is done on the k^{th} portion, the work tape head comes back to the first portion of the tape by means of the state $m_{t,k}$.

Whenever $\mathcal{L}$ is placing the head of i before the beginning (resp. after the end) of the inscription, $\mathcal{L}'$ has to insert $\dot{\square}$ to the right of $\%_i$ (resp. to the left of $\%_{i+1}$). For that matter, $\mathcal{L}'$ shifts to the left (resp. to the right) the portion of its inscription from $\%_1$ to $\%_i$ (resp. from $\%_{i+1}$ to $\%_{k+1}$) and then writes $\dot{\square}$ in the right cell. These operations are performed by a set of transitions denoted by δ_{shift} and they start from $l_{t,i} \in Q_{shift}$ (resp. $r_{t,i+1} \in Q_{shift}$) where Q_{shift} is the set of states involved in δ_{shift}. We suppose that after shifting the portion of its tape and writting $\dot{\square}$, $\mathcal{L}'$ switches to $m_{t,i}$.

Finally, take it that $\mathcal{L}' = (Q', \Sigma, \Gamma', \delta_{copy} \cup \delta_{shift} \cup \delta', q_0{}')$ where Q' is the set $Q \cup \{q_{t,i} \mid t \in \delta \text{ and } i \in [k]\} \cup \{m_{t,i} \mid t \in \delta \text{ and } i \in [k]\} \cup Q_{copy} \cup Q_{shift}$, Γ' is the set $\Gamma_1 \cup \cdots \cup \Gamma_k \cup \{\%_1, \ldots, \%_{k+1}\} \cup \mathrm{H}_1 \cup \cdots \cup \mathrm{H}_k$ and δ' is constructed this way: $t = (q, c, X_1, \ldots, X_k, Y_1, \blacklozenge_1, Y_2, \blacklozenge_2, \ldots, Y_k, \blacklozenge_k, q') \in \delta$ if and only if the following set is part of δ':

$$\{(q,c,\dot{X}_1,Y_1,\blacklozenge_1,q_{t,1})\} \ \bigcup \ \{(q_{t,i},\varepsilon,X,\dot{X},\blacktriangleright,m_{t,i}) \mid i\in[k-1],\ X\in\Gamma_i\} \ \bigcup$$

$$\{(q_{t,i},\varepsilon,\%_i,\%_i,\blacksquare,l_{t,i}) \mid i\in[k]\} \ \bigcup \ \{(q_{t,i},\varepsilon,\%_{i+i},\%_{i+1},\blacksquare,r_{t,i+1}) \mid i\in[k]\} \ \bigcup$$

$$\{(q_{t,k},\varepsilon,X,\dot{X},\blacksquare,m_{t,k}) \mid X\in\Gamma_k\} \ \bigcup$$

$$\{(m_{t,i},\varepsilon,X,X,\blacktriangleright,m_{t,i}) \mid i\in[k-1],\ X\in\Gamma_i\cup\{\%_{i+1}\}\cup\Gamma_{i+1}\} \ \bigcup$$

$$\{(m_{t,i},\varepsilon,\dot{X}_{i+1},Y_{i+1},\blacklozenge_{i+1},q_{t,i+1}) \mid i\in[k-1]\} \ \bigcup$$

$$\{(m_{t,k},\varepsilon,X,X,\blacktriangleleft,m_{t,k}) \mid X\in\textstyle\bigcup_{i\in[k]}\Gamma_i\cup\{\%_1,\ldots,\%_k\}\cup\bigcup_{i\in[k]\setminus\{1\}}\mathrm{H}_i\} \ \bigcup$$

$$\{(m_{t,k},\varepsilon,\dot{X},\dot{X},\blacksquare,q') \mid X\in\Gamma_1\}\ .$$

Consequently, the internal configurations of $\mathcal{L}'$ are the elements of

$$\%_1.{\Gamma_1}^*.Q'.{\Gamma_1}^*.\mathrm{H}_1.{\Gamma_1}^*.\%_2.{\Gamma_2}^*.\mathrm{H}_2.{\Gamma_2}^*.\%_3\ \ldots\%_k{\Gamma_k}^*.\mathrm{H}_k.{\Gamma_k}^*.\%_{k+1}\ \cup\ \ldots$$

(the work tape head may be located anywhere, so the dots mean "the same with Q' anywhere else").

Obviously, $\mathcal{L}'$ is an LBM because the number of cells that are used on its work tape is a linear function of the size of the input word; this number is $(k+1)+\sum_{i\in[k]}S_i(n)$ where $k+1$ stands for the number of cells containing a delimiter, n is the size of the input word and the S_i are the k linear functions in the size of the input word of $\mathcal{L}$. Notice that the number of states of $\mathcal{L}'$ is in $O\big(k.|\delta|\big)$ and the number of transitions of $\mathcal{L}'$ is in $O\big(|\delta|.\sum_{i\in[k]}|\Gamma_i|\big)$.

In view of the definition of $\mathcal{L}'$, the following proposition is straighforward.

Proposition 3.1. *Let ι and ι' denote the initial configuration of $\mathcal{L}$ and $\mathcal{L}'$ respectively. Then, $\mathcal{G}_{\mathcal{L}}[\iota]$ and $\mathcal{G}_{\mathcal{L}'}[\iota']$ are ε–equivalent.*

4 Synchronized Product of LBM's

We consider n LBM $\mathcal{L}_i$, $i\in[n]$. Suppose that for all i in $[n]$, $\mathcal{L}_i$ has k_i work tapes, ι_i denotes the initial configuration of $\mathcal{L}_i$ and $\mathcal{L}_i=(Q_i,\Sigma_i,\Gamma_{i,1},\ldots,\Gamma_{i,k_i},\delta_i,{q_0}_i)$.

Let $\mathcal{C}\subseteq\prod_{i\in[n]}(\Sigma_i\cup\{\varepsilon\})$ be a synchronization constraint. The *composed LBM* $\mathcal{L}$ is the $\sum_{i\in[n]}k_i$ work tapes LBM defined as follows: the set of states is $\prod_{i\in[n]}Q_i$, the input alphabet is $\mathcal{C}$, the work tape alphabets are all the $\Gamma_{i,j}$ for $i\in[n]$ and $j\in[k_i]$, the initial state is $({q_0}_1,\ldots,{q_0}_n)$, and the set of transitions is composed of the tuples

$$\Big(q,a,(X_{1,j})_{j\in[k_1]}\ ,\ldots,(X_{n,j})_{j\in[k_n]}\ ,(Y_{1,j},\blacklozenge_{1,j})_{j\in[k_1]}\ ,\ldots,(Y_{n,j},\blacklozenge_{n,j})_{j\in[k_n]}\ ,q'\Big)$$

such that $a\in\mathcal{C}$ and, for each $i\in[n]$, $(q_i,a_i,(X_{i,j})_{j\in[k_i]}\ ,(Y_{i,j},\blacklozenge_{i,j})_{j\in[k_i]}\ ,q'_i)\in\delta_i$. Here, for each $i\in[n]$, $(X_{i,j})_{j\in[k_i]}$ stands for the sequence $X_{i,1}\ ,\ldots,X_{i,k_i}$ and

$(Y_{i,j}, \blacklozenge_{i,j})_{j\in[k_i]}$ stands for the sequence $Y_{i,1}\ , \blacklozenge_{i,1}\ , \ldots, Y_{i,k_i}\ , \blacklozenge_{i,k_i}$. Moreover, q_i (resp. a_i, resp. q'_i) stands for the i^{th} coordinate of tuple q (resp. a, resp. q'). Notice that the number of states of $\mathcal{L}$ is equal to $\prod_{i\in[n]} |Q_i|$ and that the number of transitions of $\mathcal{L}$ is at most equal to $\prod_{i\in[n]} |\delta_i|$.

Proposition 4.1. *Let ι denote the initial configuration of $\mathcal{L}$. Then, graphs $\mathcal{G}_{\mathcal{L}}[\iota]$ and $\prod^{\mathcal{C}}_{i\in[n]} \mathcal{G}_{\mathcal{L}_i}[\iota_i]$ are isomorphic.*

Proof. Let ϕ be the one–to–one correspondence

$$\prod_{i\in[n]} \big(\prod_{j\in[k_i]} \mathit{\Gamma}_{i,j}{}^*.Q_i.\mathit{\Gamma}_{i,j}{}^*\big) \longrightarrow \prod_{i\in[n],j\in[k_i]} \mathit{\Gamma}_{i,j}{}^*.\big(\prod_{l\in[n]} Q_l\big).\mathit{\Gamma}_{i,j}{}^* \quad \text{such that}$$

$$\phi\Big((\mu_{1,1}q_1\nu_{1,1}\ , \ldots, \mu_{1,k_1}q_1\nu_{1,k_1})\ , \ldots, (\mu_{n,1}q_n\nu_{n,1}\ , \ldots, \mu_{n,k_n}q_n\nu_{n,k_n})\Big)$$
$$= \big(\mu_{1,1}(q_1, \ldots, q_n)\nu_{1,1}\ , \ldots, \mu_{n,k_n}(q_1, \ldots, q_n)\nu_{n,k_n}\big)\ .$$

The roots $\iota_1, \ldots, \iota_n$ and ι are such that $\phi\big(\iota_1, \ldots, \iota_n\big) = \iota$. Moreover, in view of the definition of the set of transitions of $\mathcal{L}$, it is clear that $d \xrightarrow{a} d'$ is an edge of $\prod^{\mathcal{C}}_{i\in[n]} \mathcal{G}_{\mathcal{L}_i}[\iota_i]$ if and only if $\phi(d) \xrightarrow{a} \phi(d')$ is an edge of $\mathcal{G}_{\mathcal{L}}[\iota]$. □

In view of the above, the following corollary is obvious.

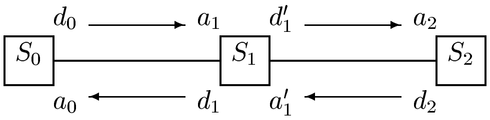
Corollary 4.2. *The class of graphs of LBM's is closed, up to isomorphism, under synchronized product.*

The next corollary is obtained as a composition of Proposition 3.1 and Proposition 4.1.

Corollary 4.3. *The class of graphs of single work tape LBM's is closed, up to ε-equivalence, under synchronized product.*

5 An Example

We consider a small portion of a railway network that is composed of three stations S_0, S_1 and S_2 linked together by a single track:



Train crossing is allowed only at station S_1 which is composed of two platforms. Notice that a train arriving to S_1 from S_0 (resp. S_2) may go back to S_0 (resp. S_2). We model the behaviour of this railway portion by means of a synchronized product of LBM's.

This portion of the railway network can be seen as a *composed process* in the sense that its *behaviour* is the result of the *parallel* working of station S_1 and portions S_0S_1 and S_1S_2 of the track. Note that this composed process, unlike its composing parts, is not a pushdown one *i.e.* its behaviour cannot be modelled by a graph which is the one of a pushdown process. Indeed, suppose that left–to–right motions are distinguished from right–to–left motions. In the left–to–right direction, the departure of a train from S_0 is represented by d_0, the arrival of a train at S_1 is represented by a_1, the departure of a train from S_1 is represented by d_1' and the arrival of a train at S_2 is represented by a_2. Then, if $\mathcal{G}$ is a rooted graph modelling the behaviour of the whole portion, the set $\mathrm{Lang}(\mathcal{G})$ of the labels of the paths from the root to any other vertex looks like

$$\Big\{ w \in \{a_0, d_0, a_1, d_1, a_1', d_1', a_2, d_2\}^* \;\Big|\; \forall u, v \in \{a_0, d_0, a_1, d_1, a_1', d_1', a_2, d_2\}^*,\\ uv = w \Rightarrow (|u|_{a_1} \leq |u|_{d_0},\ |u|_{a_2} \leq |u|_{d_1'},\ |u|_{a_1'} \leq |u|_{d_2},\ |u|_{a_0} \leq |u|_{d_1},\\ |u|_{d_1} \leq |u|_{a_1} + |u|_{a_1'} \text{ and } |u|_{d_1'} \leq |u|_{a_1} + |u|_{a_1'}) \Big\}\,.$$

Obviously, $\mathrm{Lang}(\mathcal{G})$ is context–sensitive and one can establish, using Ogden's Lemma, that $\mathrm{Lang}(\mathcal{G})$ is not context–free. But, if $\mathcal{G}$ was the graph of a pushdown process, then $\mathrm{Lang}(\mathcal{G})$ would be a context–free language [5]. Thus, $\mathcal{G}$ is not the graph of a pushdown process.

In order to model the behaviour of the portion of a railway network, we need the following construction. We associate to any LBM $\mathcal{L} = (Q, \Sigma, \Gamma_1, \ldots, \Gamma_k, \delta, q_0)$ the LBM $\mathcal{L}^{\mathbb{1}} = (Q, \Sigma, \Gamma_1, \ldots, \Gamma_k, \delta \cup \delta^{\mathbb{1}}, q_0)$ where

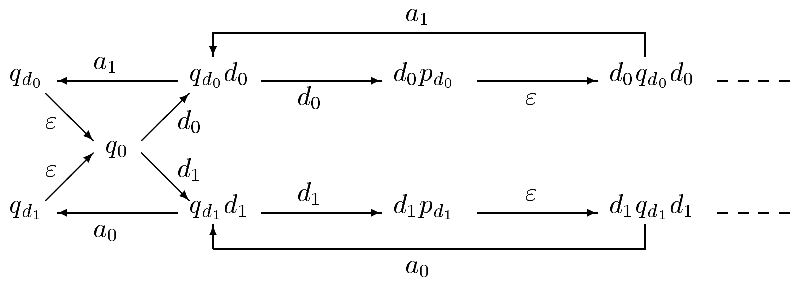
$$\delta^{\mathbb{1}} = \{(q, \varepsilon, X_1, \ldots, X_k, X_1, \blacksquare, \ldots, X_k, \blacksquare, q) \mid q \in Q \text{ and } \forall i \in [k],\ X_i \in \Gamma_i\}$$

Observe that $\mathcal{G}_{\mathcal{L}^{\mathbb{1}}}$ differs from $\mathcal{G}_{\mathcal{L}}$ only by ε-loops added to each vertex by $\delta^{\mathbb{1}}$.

Let us now specify each part of this railway portion using LBM's. The behaviour of the track between S_0 and S_1 can be modelled by the machine $\mathcal{L}^{\mathbb{1}}_{S_0S_1}$ where $\mathcal{L}_{S_0S_1} = \big(\{q_0, q_{d_0}, q_{a_1}, q_{d_1}, q_{a_0}\}, \{a_0, d_0, a_1, d_1\}, \{\square, d_0, d_1\}, \delta_{S_0S_1}, q_0\big)$ with

$$\delta_{S_0S_1} = \left\{ \begin{array}{l} (q_0, d_0, \square, d_0, \blacksquare, q_{d_0}), (q_0, d_1, \square, d_1, \blacksquare, q_{d_1}), (q_{d_0}, d_0, d_0, d_0, \blacktriangleright, p_{d_0}), \\ (q_{d_0}, \varepsilon, \square, \square, \blacksquare, q_0), (q_{d_0}, a_1, d_0, \square, \blacktriangleleft, q_{d_0}), (p_{d_0}, \varepsilon, \square, d_0, \blacksquare, q_{d_0}), \\ (q_{d_1}, d_1, d_1, d_1, \blacktriangleright, p_{d_1}), (q_{d_1}, \varepsilon, \square, \square, \blacksquare, q_0), (q_{d_1}, a_0, d_1, \square, \blacktriangleleft, q_{d_1}), \\ (p_{d_1}, \varepsilon, \square, d_1, \blacksquare, q_{d_1}) \end{array} \right\}.$$

The graph of $\mathcal{L}_{S_0S_1}$ may be depicted as follows:



Notice that $\mathrm{Lang}(\mathcal{G}_{\mathcal{L}^{\mathbb{1}}_{S_0S_1}})$, the set of the labels of the paths in $\mathcal{G}_{\mathcal{L}^{\mathbb{1}}_{S_0S_1}}$ from q_0 to any other vertex, is L^* where

$$L = \big\{ w \in \{d_0, a_1\}^* \;\big|\; \forall u, v \in \{d_0, a_1\}^* \; (uv = w \Rightarrow |u|_{a_1} \leq |u|_{d_0}) \big\} \; \cup \\ \big\{ w \in \{d_1, a_0\}^* \;\big|\; \forall u, v \in \{d_1, a_0\}^* \; (uv = w \Rightarrow |u|_{a_0} \leq |u|_{d_1}) \big\} \; .$$

This means that there can't be more train arrivals than train departures in each direction on this portion of the track and that any train arriving at S_1 may go back to S_0. The behaviour of the track between S_1 and S_2 can be modelled by the LBM $\mathcal{L}^{\mathbb{1}}_{S_1S_2}$, the input alphabet of which is $\{a'_1, d'_1, a_2, d_2\}$, defined in the same way as $\mathcal{L}^{\mathbb{1}}_{S_0S_1}$.

Station S_1 is composed of two platforms. One of them is modelled by the LBM $\mathcal{L}^{\mathbb{1}}_1 = \big(\{q_a, q_d\}, \{a, d\}, \{\square\}, \{(q_a, a, \square, \square, \blacktriangleright, q_d), (q_d, d, \square, \square, \blacktriangleleft, q_a)\} \cup \delta^{\mathbb{1}}_1, q_a\big)$ which is such that $\mathrm{Lang}(\mathcal{G}_{\mathcal{L}^{\mathbb{1}}_1}) = (ad)^*$. The other one is modelled by the LBM $\mathcal{L}^{\mathbb{1}}_2$, the input alphabet of which is $\{a', d'\}$, which is defined in the same way as $\mathcal{L}^{\mathbb{1}}_1$. Letters a, d, a' and d' have the same meaning as $a_0 \ldots$ above.

Station S_1 can be modelled by the synchronized product of $\mathcal{L}^{\mathbb{1}}_1$ and $\mathcal{L}^{\mathbb{1}}_2$ with respect to constraint $\{a, d, \varepsilon\} \times \{a', d', \varepsilon\}$ because it is composed of its two platforms that work in parallel. Let $\mathcal{L}_{S_1}$ be this product.

The whole portion of the railway network can now be modelled by the synchronized product of $\mathcal{L}^{\mathbb{1}}_{S_0S_1}$, $\mathcal{L}^{\mathbb{1}}_{S_1S_2}$ and $\mathcal{L}_{S_1}$ with respect to the synchronization constraint $\mathcal{C}$ described as follows: any arrival to S_1 from a track corresponds to an arrival on a platform and any departure from a platform corresponds to a departure from S_1 to a track. Let $\Sigma_{S_0S_1}$, Σ_{S_1} and $\Sigma_{S_1S_2}$ denote the input alphabet of $\mathcal{L}^{\mathbb{1}}_{S_0S_1}$, $\mathcal{L}^{\mathbb{1}}_{S_1}$ and $\mathcal{L}^{\mathbb{1}}_{S_1S_2}$ respectively. Then, constraint $\mathcal{C}$ is the set of the tuples $(c_1, (c_{21}, c_{22}), c_3) \in \Sigma_{S_0S_1} \cup \{\varepsilon\} \times \Sigma_{S_1} \times \Sigma_{S_1S_2} \cup \{\varepsilon\}$ such that

$$\big(c_1 = a_1 \Leftrightarrow (c_{21} = a \text{ or } c_{21} = a')\big), \; \big(c_3 = a'_1 \Leftrightarrow (c_{22} = a \text{ or } c_{21} = a')\big), \\ \big(c_1 = d_1 \Leftrightarrow (c_{21} = d \text{ or } c_{21} = d')\big) \text{ and } \big(c_3 = d'_1 \Leftrightarrow (c_{22} = d \text{ or } c_{22} = d')\big) \; .$$

We do not give a complete description of the machine that models the whole portion of the network because it is quite a big machine (to have an idea, one may compute its number of states which is equal to $|Q_{S_0S_1}| \times |Q_{S_1S_2}| \times |Q_{S_1}| = 100$ where $Q_{S_0S_1}$, $Q_{S_1S_2}$ and Q_{S_1} stand for the set of states of $\mathcal{L}^{\mathbb{1}}_{S_0S_1}$, $\mathcal{L}^{\mathbb{1}}_{S_1S_2}$ and $\mathcal{L}^{\mathbb{1}}_{S_1}$ respectively).

6 First–order Logic on Graphs of Linear Bounded Machines

Up to now, we have only addressed the problem of the specification of communicating processes within an approach based on linear bounded machines. In the present section, we discuss the problem of formal verifications within this approach. More precisely we assume that a system of communicating processes has been specified, *viz* each sequential process has been described by an LBM and their interaction has been expressed as some synchronization constraint. Up to ε–equivalence, such a system may be represented by an LBM, the graph of which is a synchronized product of the graphs of composing processes. The verification problem consists then in checking the truth of a formula of some logic on the resulting graph considered as a model–theoretic structure.

Concerning the verification problem, we claim that in the area of LBM specifications, even for rather weak logics, one should not expect algorithmic solutions but rather semi–algorithmic ones. More precisely we establish that the first–order theory of the graphs of LBM's is not recursive (even not recursively enumerable).

Formally, the *first–order theory* of a rooted graph $\mathcal{G} \subseteq D \times C \times D$ where $C = \Sigma \cup \{\varepsilon\}$, is defined as follows. The variables form an infinite countable set X and are interpreted as vertices of $\mathcal{G}$. The binary predicates $\mathbf{s}_c$ for each $c \in C$, $=$ and the unary predicate $\mathbf{r}$ allow to build atomic formulae. These predicates are interpreted on $\mathcal{G}$ resp. as $\xrightarrow{c}$, the identity relation on D and the singleton $\{e\}$, where e stands for the root of $\mathcal{G}$. Using classical connectives and quantifiers, from atomic formulae that are of the form $\mathbf{s}_c(x, y)$, $x = y$ or $\mathbf{r}(x)$, first order formulae are constructed in the usual way. The set of valid sentences on $\mathcal{G}$, *i.e.* valid formulae on $\mathcal{G}$ with no free variable, is called *the first–order theory of* $\mathcal{G}$.

Example 6.1. The sentence $\forall x \, \exists y \, \bigvee_{c \in C} \mathbf{s}_c(x, y)$ belongs to the first–order theory of a graph $\mathcal{G}$ if and only if $\mathcal{G}$ has no sink.

The indecidability result that we have to establish involves linear bounded automata (LBA for short). Formally an LBA $\mathcal{L}_f$ is a single–work–tape[2] LBM $\mathcal{L} = (Q, \Sigma, \Gamma, \delta, q_0)$ with a distinguished state $f \in Q$, called the *final state*. The language accepted by $\mathcal{L}_f$, written $\mathrm{Lang}(\mathcal{L}_f)$, is the set of labels of all paths in the graph $\mathcal{G}_{\mathcal{L}}[\iota]$ from the initial configuration ι to $\mu f \nu$ for some $\mu, \nu \in \Gamma^*$. Two LBA's $\mathcal{L}_f$ and $\mathcal{L}'_{f'}$ are *equivalent* if $\mathrm{Lang}(\mathcal{L}_f) = \mathrm{Lang}(\mathcal{L}'_{f'})$.

The emptiness problem for LBA's is the following decision problem.
Instance: An LBA $\mathcal{L}_f$.
Question: $\mathrm{Lang}(\mathcal{L}_f) = \varnothing$?
It is well known that this problem is not recursively enumerable. Using this fact, we can establish the following.

Theorem 6.2. *The problem*
Instance: *An LBM $\mathcal{L}$ and a first–order sentence φ.*
Question: *Does φ belong to the first–order theory of $\mathcal{G}_{\mathcal{L}}$?*
is not recursively enumerable.

Proof. We show that the emptiness problem for LBA's is many–one reducible to the problem of the statement.

Let $\mathcal{L}_f = (Q, \Sigma, \Gamma, \delta, q_0, f)$ be an LBA. Let $\mathcal{L}'_f = (Q', \Sigma, \Gamma, \delta', q_0, f)$ be an LBA equivalent to $\mathcal{L}_f$ and satisfying the following properties:

(1) $\delta' \cap (\{f\} \times C \times \Gamma \times \Gamma \times \{◀, ▶, ■\} \times Q') = \varnothing$,

[2] In fact an LBA can be multitape but for the purpose of the paper single–work–tape LBA's suffice.

(2) $\delta' \cap (\{q\} \times C \times \{X\} \times \Gamma \times \{◀, ▶, ■\} \times Q') \neq \varnothing$ for all $(q, X) \in (Q \setminus \{f\}) \times \Gamma$.

Such an LBA may readily be constructed. Concerning (1), for each rule of δ that has the form $(q_1, c_1, X_1, Y_1, ♦_1, f)$, we add a rule $(q_1, c_1, X_1, Y_1, ♦_1, q')$ where $q' \notin Q$ is a new state and each rule $(f, c_2, X_2, Y_2, ♦_2, q_2) \in \delta$ is replaced by the rule $(q', c_2, X_2, Y_2, ♦_2, q_2)$. Concerning (2), we add a new state $p \notin Q$, the rules $(p, \varepsilon, X, X, ■, p)$ for all $X \in \Gamma$ and for each $(q, X) \in (Q \setminus \{f\}) \times \Gamma$ such that $\delta \cap (\{q\} \times \Sigma \times \{X\} \times \Gamma \times \{◀, ▶, ■\} \times Q) = \varnothing$, we add the rule $(q, \varepsilon, X, X, ■, p)$.

Now $\mathrm{Lang}(\mathcal{L}_f) = \mathrm{Lang}(\mathcal{L}'_f)$ and the following holds: a vertex $\mu q \nu$ of $\mathcal{G}_{\mathcal{L}'_f}$ is a sink if and only if $q = f$. Consequently $\mathrm{Lang}(\mathcal{L}_f) = \varnothing$ if and only if the sentence of Example 6.1 belongs to the first–order theory of $\mathcal{G}_{\mathcal{L}'_f}$. □

Taking into account the fact that the property of being a sink is expressible in the Hennessy–Milner logic [9], the following corollary may be derived from above proof.

Corollary 6.3. *The Hennessy–Milner logic is not semi–decidable on the graphs of LBM's.*

We close this section by the following remark. If the existence of sinks is expressible within a logic, then it cannot exist a complete formal system for checking the truth of the formulae of the logic on graphs of arbitrary LBM's.

7 Conclusion

We have defined transition graphs associated to a peculiar kind of linear bounded machines that read their input performing all computations on work tapes. The closure under synchronized product of the family of graphs thus defined has been established up to observational equivalence (considering ε–transitions as non observable) using two transformations. We hope that both transformations may be improved using some speedup and tape compression techniques so as to preserve bisimulation (or even isomorphism) instead of observational equivalence.

As a consequence of the closure result and similarly to the Arnold–Nivat approach, the linear bounded machines provide a uniform framework for the specification of communicating processes. Moreover the expressive power within this framework seems to be very satisfactory. However, this has a counterpart in the undecidability result. We have established that the first–order theory of the graphs of linear bounded machines is not recursively enumerable. This result extends to any logic in which the existence of sinks may be stated. It may be observed that this is one of the weakest safety properties that one should be able to express within a logic usable for verification purposes, since it corresponds to the existence of deadlocks. In spite of this negative result, we believe that some semi–decision techniques adequate for the graphs of linear bounded machines may be developed for various logics. An elementary example of this kind may be found in [10].

The transition graph of linear bounded machine has been defined as the maximal subgraph of the configuration graph that is accessible from the initial configuration. When this accessibility requirement is dropped, we have a transition graph the vertices of which are all configurations. Since our undecidability result was related to the reachability problem, more precisely to the language of an LBA, the emptiness problem for LBA does not lead to a similar result in the latter case. Is the first-order theory of such graphs still undecidable ? Currently, we do not know the answer to this question.

References

1. A. Arnold. *Finite Transition Systems.* Prentice Hall Int., 1994.
2. A. Arnold and M. Nivat. Comportements de processus. In *Colloque AFCET "Les mathématiques de l'Informatique"*, pages 35–68, 1982.
3. G. Boudol. Notes of algebraic calculi of processes. In *Logics and Models of Concurrent Systems*, volume F–13 of *NATO ASI series*, pages 261–303. 1985.
4. R. Bryant. Binary decision diagrams and beyond: Enabling technologies for formal verification. In *Proceedings of the International Conference on Computer Aided Design, ICCAD'95*, 1995.
5. D. Caucal. On infinite transition graphs having a decidable monadic second–order theory. In F. M. auf der Heide and B. Monien, editors, *23th International Colloquium on Automata Languages and Programming*, LNCS 1099, pages 194–205, 1996.
6. G. Chaitin. A Theory of Program Size Formally Identical to Information Theory. *J. Assoc. Compt. Mach.*, (22):329–340, 1975.
7. B. Courcelle. The monadic second–order logic of graphs, II: Infinite graphs of bounded width. *Mathematical System Theory*, 21:187–221, 1989.
8. A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Formal Models and Semantics*, volume B of *Handbook of Theoretical Computer Science*, pages 997–1072. Elsevier, 1990.
9. M. Hennessy and R. Milner. Algebraic laws for nondeterminism and concurrency. *J. ACM*, 32:137–162, 1985.
10. T. Knapik. Domains of word–functions and Thue specifications. Technical Report INF/96/11/05/a, 1997.
11. D. Kozen and J. Tiuryn. Logics of programs. In J. van Leeuwen, editor, *Formal Models and Semantics*, volume B of *Handbook of Theoretical Computer Science*, pages 789–840. Elsevier, 1990.
12. R. P. Kurshan. *Computer–aided Verification of Coordinating Processes.* Princeton University Press, 1994.
13. K. McMillan. *Symbolic Model Checking.* Kluwer, 1993.
14. R. Milner. *Calculus of Communicating Systems.* LNCS 82. Springer Verlag, 1980.
15. D. E. Muller and P. E. Schupp. The theory of ends, pushdown automata and second–order logic. *Theoretical Comput. Sci.*, 37:51–75, 1985.
16. M. Nivat. Sur la synchronisation des processus. *Revue technique Thomson-CSF*, (11):899–919, 1979.