ELSEVIER

Theoretical Computer Science 151 (1995) 125–162

Theoretical Computer Science

The monadic second-order logic of graphs IX: Machines and their behaviours☆

Bruno Courcelle*

LaBRI (CNRS URA 1304), Université Bordeaux-I, 351 Cours de la Libération, 33405 Talence, France

Abstract

We establish that every monadic second-order property of the *behaviour* of a *machine* (transition systems and tree automata are typical examples of machines) is a monadic second-order property of the machine *itself*. In this way, we clarify the distinction between "dynamic" properties of machines (i.e., properties of their behaviours), and their "static" properties (i.e., properties of the graphs or relational structures representing them). It is important for program verification that the dynamic properties that one wants to verify can be formulated statically, in the simplest possible way. As a corollary of our main result, we also obtain that the monadic theory of an algebraic tree is decidable.

0. Introduction

Abstract computing devices like Turing machines, automata, transition systems, or program schemes will be called *machines*. Each machine has an associated *behaviour*: the computed function for a Turing machine, the recognized language for an automaton, a (usually infinite) tree for a transition system or a program scheme. Machines are usually finite whereas their behaviours are infinite, except in degenerated cases. Semantics deals with properties of behaviours whereas verification deals with machines: the task is to verify on a given machine some desired property of its behaviour. It is thus important to obtain a simplest possible expression *in terms of the machine* of properties of its behaviour. More precisely, for every property P of the behaviours of the machines M of a class $\mathscr{C}$, one obtains a property P_* of the machines of $\mathscr{C}$ defined by:

$$P_*(M) :\Leftrightarrow P(\mathbf{bhv}(M)),$$

☆ Work done for a cooperation project between Bordeaux-I University and Technion (Haïfa, Israel), supported by the French and Isreali ministries of research.

*E-mail: courcell@labri.u-bordeaux.fr.

SSDI 0304-3975(95)00049-6

where **bhv**(M) denotes the behaviour of M. Knowing $\mathscr{C}$, **bhv** and P, what can be said about P_*? Nothing if $\mathscr{C}$ is the class of Turing machines because Rice's Theorem tells us that P_* is undecidable except if P is identically true or false. The situation is much better as we shall prove for automata, transition systems, and program schemes if their properties and those of their behaviours are expresssed in monadic second-order logic. Such an expression is possible if machines and behaviours are represented by logical structures, which is easy to achieve. Why monadic second-order logic rather than another logical language? There are two reasons: first because it is decidable on many structures like finite or infinite trees and "tree-like" finite or infinite graphs (essentially by applications of Rabin's Theorem, see the survey by Thomas [21] for trees and Courcelle [6] for graphs) and secondly because it subsumes many languages like temporal logics or μ-calculus [13].

All our main results are of the following form (Theorem 4.1, expressed here in a nontechnical way):

For every monadic second-order property P of the infinite trees representing the behaviours of deterministic transition systems, the property P_* of these transition systems can be expressed by a monadic second-order formula. This formula can be effectively constructed from a given one expressing P.

The translation is uniform in the sense that a unique formula gives a correct expression of P_* for all deterministic transition systems. The extension to nondeterministic systems is a conjecture, see Conjecture 4.2.

The construction establishing this theorem works for infinite as well as finite deterministic transition systems. We give an application to program schemes that uses the infinite case. The behaviour of a recursive program scheme is an infinite tree, called *algebraic* in [3, 4], and that is not in general regular, i.e., that is not the behaviour of a finite transition system. The decidabililty of the monadic theory of an algebraic nonregular tree is not an immediate consequence of Rabin's Theorem. However, we establish it in the following way. An algebraic tree is the behaviour of a infinite deterministic transition system, that, considered as a graph, is "close enough to a regular tree" to have a decidable monadic theory. The result follows then from Theorem 4.1.

This paper is organized as follows: Section 1 is devoted to preliminaries concerning monadic second-order logic and trees; Section 2 establishes some results concerning automata defining sets of finitc words and trees; Section 3 deals with automata defining infinite trees; Section 4 gives applications to transition systems and Section 5 to algebraic trees; Section 6 is a final survey and a conclusion. An appendix contains the proof of the central technical result (Theorem 3.1).

1. Preliminaries

Let $\mathscr{L}$ be a logical language appropriate for describing the properties of the logical structures in a class $\mathscr{A}$. We mean by this that a *satisfaction relation* $\models$ is defined. This

relation is a subset of $\mathfrak{s} \times \mathcal{L}$. We write $S \models \varphi$ if and only if (S, φ) belongs to $\models$. Since the behaviour of a machine is frequently a *set of structures* we also define, for $\mathcal{M} \subseteq \mathfrak{s}$:

$\mathcal{M} \models \varphi$ if and only if $\mathcal{S} \models \varphi$ for all $\mathcal{S} \in \mathcal{M}$,

$\mathcal{M} \models_\exists \varphi$ if and only if $\mathcal{S} \models \varphi$ for some $\mathcal{S} \in \mathcal{M}$.

It follows in particular that for $\mathcal{M}$ and $\mathcal{L}$ as above, we have

$\emptyset \models \varphi$ for every formula φ in $\mathcal{L}$,

$\mathcal{M} \models_\exists$ **true** if and only if $\mathcal{M} \neq \emptyset$ and

$\mathcal{M} \models$ **true** for every $\mathcal{M} \subseteq \mathfrak{s}$.

Note also that $\mathcal{M} \models \varphi$ holds if and only if $\mathcal{M} \models_\exists \neg\varphi$ does not. The *$\mathcal{L}$-theory of a class $\mathcal{M}$* as above is the set of formulas in $\mathcal{L}$ such that $\mathcal{M} \models \varphi$ holds. This theory is *decidable* if it is a recursive set of formulas. The *$\mathcal{L}$-satisfiability problem for a class $\mathcal{M}$* is the problem of deciding whether a given formula φ satisfies $\mathcal{M} \models_\exists \varphi$. If $\mathcal{L}$ is closed under negation, then the $\mathcal{L}$-theory of $\mathcal{M}$ is decidable if and only if its $\mathcal{L}$-satisfiability problem is decidable.

A *transduction f from a set A to a set B* is a multivalued mapping from A to B, which one can also consider as a subset of $A \times B$. The *image* of a subset C of A *under* f is the set of all images under f of the elements of C. *The inverse image* of a subset C of B *under* f is the set of all elements of A having at least one image under f in C.

Let $\mathfrak{s}$ and $\mathfrak{s}'$ be two classes of structures and f be a transduction from $\mathfrak{s}$ to $\mathfrak{s}'$. Let $\mathcal{L}$ and $\mathcal{L}'$ be two languages expressing properties of structures in $\mathfrak{s}$ and $\mathfrak{s}'$ respectively. We say that f is *$(\mathcal{L}, \mathcal{L}')$-compatible* if there exists an algorithm that associates with every formula ϕ of $\mathcal{L}'$ a formula ψ of $\mathcal{L}$ such that for every structure S in $\mathfrak{s}$.

$S \models \psi$ if and only if $f(S) \models_\exists \varphi$.

We use the term *$\mathcal{L}$-compatible* for $(\mathcal{L}, \mathcal{L})$-compatible. If f is $(\mathcal{L}, \mathcal{L}')$-compatible and if a subclass $\mathcal{M}$ of $\mathfrak{s}$ has a decidable $\mathcal{L}$-satisfiability problem, then the class $f(\mathcal{M})$ has a decidable $\mathcal{L}'$-satisfiability problem.

These general definitions will be used for relational structures and monadic second-order logic. We review these notions together with that of a monadic second-order definable transduction of structures. Let R be a finite ranked set of symbols where each element r in R has a rank $\rho(r)$ in $\mathbb{N}_+$. A symbol r in R is considered as a $\rho(r)$ -ary relation symbol. An *R-(relational) structure* is a tuple $S = \langle \mathbf{D}_S, (r_S)_{r \in R} \rangle$ where $\mathbf{D}_S$ is a (possibly empty) set, called the *domain* of S, and r_S is a subset of $\mathbf{D}_S^{\rho(r)}$ for each r in R. We denote by $\mathfrak{s}(R)$ the class of R-structures.

We review *monadic second-order logic* (*MS logic*) briefly. Its formulas (called *MS formulas* for short), intended to describe properties of structures S as above, are written with variables of two types namely lower case symbols x, x', y, ... called *object variables*, denoting elements of $\mathbf{D}_S$, and upper case symbols X, Y, Y', ... called *set*

variables, denoting subsets of $\mathbf{D}_S$. The atomic formulas are of the forms $x = y$, $r(x_1, \ldots, x_n)$ where r is in R and $n = \rho(r)$), and $x \in X$, and formulas are formed with propositional connectives and quantifications over the two kinds of variables. For every finite set W of object and set variables, we denote by $\mathscr{L}(R, W)$ the set of all formulas that are written with relational symbols from R and have their free variables in W; we also let $\mathscr{L}(R) := \mathscr{L}(R, \emptyset)$ denote the set of closed formulas.

Let S be an R-structure, let $\varphi \in \mathscr{L}(R, W)$, and let γ be a W-assignment in S, (i.e., $\gamma(X)$ is a subset of $\mathbf{D}_S$ for every set variable X in W, and $\gamma(x) \in \mathbf{D}_S$ for every object variable x in W; we write this $\gamma : W \to S$ to be short). We write $(S, \gamma) \models \varphi$ if and only if φ holds in S for γ. We write $S \models \varphi$ in the case where φ has no free variable. A property of R-structures is *MS-definable* if there is a formula φ in $\mathscr{L}(R)$ such that, for every R-structure S, this property holds if and only if $S \models \varphi$; a class of R-structures is *MS-definable* if it is the class of R-structures that satisfies an MS-definable property. The theory of a class of structures with respect to MS logic is usually called its *monadic theory*. The corresponding satisfiability problem will be called its *monadic satisfiability problem*.

We review from [10] MS-*definable tranductions of relational structures*, which will turn out to be MS-compatible. Let R and $\mathscr{Q}$ be two finite ranked sets of relation symbols. The objective is to specify a transduction from $\sigma(R)$ to $\sigma(\mathscr{Q})$ by MS formulas. A simple way to do that is by means of a tuple of formulas $(\psi, (\theta_q)_{q \in \mathscr{Q}})$ where $\psi \in \mathscr{L}(R, \{x_1\})$, $\theta_q \in \mathscr{L}(R, \{x_1, \ldots, x_{\rho(q)}\})$, for $q \in \mathscr{Q}$. For every S in $\sigma(R)$, we let T in $\sigma(\mathscr{Q})$ be defined as follows:

$$\mathbf{D}_T = \{d \mid d \in \mathbf{D}_S,\ S \models \psi(d)\},$$

for each q in $\mathscr{Q}$:

$$q_T = \{(d_1, \ldots, d_{\rho(q)}) \in \mathbf{D}_T^{\rho(q)} \mid S \models \theta_q(d_1, \ldots, d_{\rho(q)})\}.$$

The idea is to transform a structure S into a structure T by defining T "inside" S by means of MS formulas. This is nothing but the classical notion of *semantic intepretation* (see for instance Rabin [20]). However, we extend this notion in three ways. First we introduce an MS formula φ that specifies the structures S for which T is to be defined; second we use formulas written with a set W of free set variables, called the *parameters*, so that T is defined not only from S but also from subsets of its domain taken as values of the parameters; third, we define T inside an intermediate structure made of k disjoint copies of S (for some fixed k). This makes it possible to construct T with a domain larger than that of S (larger within the factor k). We now give a formal definition. We let W be a finite set of set variables called the parameters. A $(\mathscr{Q}, R)$-*definition scheme* is a tuple of formulas of the form

$$\Delta = (\varphi, \psi_1, \ldots, \psi_k, (\theta_z)_{z \in \mathscr{Q}^* k}),$$

where

$$k > 0, \quad \mathscr{Q}^* k := \{(q, j) \mid q \in \mathscr{Q}, j \in [k]^{\rho(q)}\}, \quad [k] = \{1, \ldots, k\},$$

$$\varphi \in \mathscr{L}(R, W),$$

$\psi_i \in \mathcal{L}(R, W \cup \{x_1\})$ for $i = 1, \ldots, k$,

$\theta_z \in \mathcal{L}(R, W \cup \{x_1, \ldots, x_{\rho(q)}\})$, for $z = (q, j) \in \mathcal{Q}^* k$.

These formulas are intended to define a structure T in $\mathcal{I}(\mathcal{Q})$ from a structure S in $\mathcal{I}(R)$ and values of the parameters in the following way: T is defined only if φ holds true in S for the considered values of the parameters; assuming this condition fulfilled, the formulas $\psi_1, \ldots \psi_k$, define the domain of T as the disjoint union of the sets $D_1, \ldots, D_k$, where D_i is the set of elements of the domain of S that satisfy ψ_i. Finally, the formulas θ_z for $z = (q, j), j \in [k]^{\rho(q)}$, define the relation q_T, $\theta_{(q,j)}$ specifies the $\rho(q)$-tuples $(d_1, \ldots, d_{\rho(q)})$ such that $((d_1, i_1), \ldots, (d_{\rho(q)}, i_{\rho(q)})) \in q_T$ where $j = (i_1, \ldots, i_{\rho(q)})$. Here are the formal definitions.

Let $S \in \mathcal{I}(R)$, let γ be a W-assignment in S such that $(S, \gamma) \models \varphi$. The $\mathcal{Q}$-structure T with domain $\mathbf{D}_T \subseteq \mathbf{D}_S \times [k]$ *defined in* (S, γ) by Δ is such that

(i) $\mathbf{D}_T = \{(d, i) \mid d \in \mathbf{D}_S, i \in [k], (S, \gamma, d) \models \psi_i\}$

(ii) for each q in $\mathcal{Q}$:

$q_T = \{((d_1, i_1), \ldots, (d_t, i_t)) \in \mathbf{D}_T^t \mid (S, \gamma, d_1, \ldots, d_t) \models \theta_{(q,j)}\}$ where $j = (i_1, \ldots, i_t)$ and $t = \rho(q)$.

(By $(S, \gamma, d_1, \ldots, d_t) \models \theta_{(q,j)}$, we mean $(S, \gamma') \models \theta_{(q,j)}$, where γ' is the assignment extending γ, such that $\gamma'(x_i) = d_i$ for all $i = 1, \ldots, t$: a similar convention is used for $(S, \gamma, d) \models \psi_i$.) Since T is associated in a unique way with S, γ and Δ whenever $(S, \gamma) \models \varphi$, we can use the functional notation $\mathbf{def}_\Delta(S, \gamma)$ for T. We write $\mathbf{def}_\Delta(S) = T$ when there are no parameters so that no assignment γ is involved. The *transduction defined by* Δ is the relation $\mathbf{def}_\Delta := \{(S, T) \mid T = \mathbf{def}_\Delta(S, \gamma)$ for some W-assignment γ in $S\} \subseteq \mathcal{I}(R) \times \mathcal{I}(\mathcal{Q})$. A transduction $\tau \subseteq \mathcal{I}(R) \times \mathcal{I}(\mathcal{Q})$ is MS-*definable* if it is equal to $\mathbf{def}_\Delta$ for some $(\mathcal{Q}, R)$-definition scheme Δ. A *bidefinable coding* between two classes of structures is a bijection τ between these classes such that τ and its inverse are MS-definable.

Example. We consider the functional transduction that maps a word u in $\{a, b\}^+$ to the word u^3 where $\{a, b\}^+$ denotes the set of nonempty words written with a and b. In order to define it as a transduction of structures, we represent the words in $\{a, b\}^+$ as structures in the following way. If u has length n, then the associated structure $\|u\|$ is $\langle \{1, 2, \ldots, n\}, \mathbf{suc}, \mathbf{p}_a, \mathbf{p}_b \rangle$, where

the domain $\{1, 2, \ldots, n\}$ is the set of positions of letters in u,

$\mathbf{p}_a(i)$ holds if and only if a is the letter at ith position,

$\mathbf{p}_b(i)$ holds if and only if b is the letter at ith position,

$\mathbf{suc}(i, j)$ holds if and only if $j = i + 1$.

For example, $S = \langle \{1, 2, 3, 4\}, \mathbf{suc}, \mathbf{p}_a, \mathbf{p}_b \rangle$ represents the word *abba*. We shall construct a definition scheme Δ such that $\mathbf{def}_\Delta(S)$ is the structure

$T = \langle \{1_1, 2_1, 3_1, 4_1, 1_2, 2_2, 3_2, 4_2, 1_3, 2_3, 3_3, 4_3\}, \mathbf{suc}', \mathbf{p}'_a, \mathbf{p}'_b \rangle$

where we write i_j instead of (i, j), so that i_j is the jth copy of i for $j = 1, 2, 3$, and

$\mathbf{suc}'(i_j, k_m)$ holds if and only if k_m is the successor of i_j in the above enumeration of the domain of T.
$\mathbf{p}'_a(i_j)$ holds if $i \in \{1, 4\}$, $j \in \{1, 2, 3\}$ and
$\mathbf{p}'_b(i_j)$ holds otherwise.

Clearly, T represents the word $u^3 = abbaabbaabba$.

We let Δ be the definition scheme without parameter $(\varphi, \psi_1, \psi_2, \psi_3, (\theta_{(\mathbf{suc}, i, j)})_{i,j=1,2,3}, (\theta_{(\mathbf{p}_a,i)})_{i=1,2,3}\ (\theta_{(\mathbf{p}_b,i)})_{i=1,2,3})$ such that

φ expresses that the input structure indeed represents a word in $\{a, b\}^+$,
ψ_1, ψ_2, ψ_3 are identical to the Boolean constant **true**,
$\theta_{(\mathbf{suc},i,j)}(x_1, x_2)$ is $\mathbf{suc}(x_1, x_2)$ if $i = j$,
$\theta_{(\mathbf{suc},i,j)}(x_1, x_2)$ expresses that x_1 is the last position and that x_2 is the first one if $i = 1$ and $j = 2$, or if $i = 2$ and $j = 3$,
$\theta_{(\mathbf{suc},i,j)}(x_1, x_2)$ is the constant **false** otherwise,
$\theta_{(\mathbf{p}_a,i)}(x_1)$ is $\mathbf{p}_a(x_1)$ for $i = 1, 2, 3$,
$\theta_{(\mathbf{p}_b,i)}(x_1)$ is $\mathbf{p}_b(x_1)$ for $i = 1, 2, 3$.

It is clear that for every word u, the structure $\mathbf{def}_\Delta(\|u\|)$ is isomorphic to the structure $\|u^3\|$.

Proposition 1.1 (Courcelle [10]). (1) *Every MS-definable transduction is MS-compatible. In other words, the inverse image of and MS-definable class of structures under an MS-definable transduction is an MS-definable class of structures.*

(2) *If a class of structures has a decidable monadic theory or a decidable monadic satisfiability problem, then so has its image under an MS-definable transduction.*

The behaviour mappings of machines that we shall consider below will be MS-compatible without being MS-definable.

1.1. Trees

We review the various notions of trees that we shall use. We define trees in terms of relational structures in order to express their properties by logical formulas. All trees will be rooted and directed in such a way that every node is reachable from the root by a unique directed path.

A *tree* is a relational structure $t = \langle N, \mathbf{Suc}\rangle$ the domain of which, N, is the set of *nodes* of t and where **Suc** is a binary relation on N called the *successor relation*, subject to conditions listed below. (If (x, y) belongs to **Suc**, we say that y is a *successor* of x and that x is a *predecessor* of y. We shall denote by $\mathbf{Suc}(x)$ the set of successors of a node x.) Here are the conditions that **Suc** must satisfy:

(1) there exists a node r such that (r, x) belongs to $\mathbf{Suc}^*$ for every x in N (where $\mathbf{Suc}^*$ denotes the reflexive and transitive closure of **Suc**),

(2) there is no pair (x, y) in **Suc** such that (y, x) belongs to **Suc***,

(3) every node has at most one predecessor.

If these conditions hold, then there exists a unique node r satisfying (1), it will be called the *root* of t; it has no predecessor and all other nodes have exactly one predecessor. Considering t as a simple directed graph with set of vertices N and set of edges **Suc** (handled here as a subset of $N \times N$), one can express conditions (1)–(3) as follows (in the language of graph theory): there is a vertex r from which every vertex is reachable by a directed path, the graph has no loop and no circuit, and every vertex has indegree at most 1. It follows that every vertex is reachable from the root by a unique directed path called its *access path.*

In certain cases it is useful to attach labels to nodes. A *node labelled tree* can be defined as a relational structure $t = \langle N, \mathbf{Suc}, (P_a)_{a\in A}\rangle$ where A is the set of node labels, P_a is a unary relation on N, i.e., a set of nodes and x belongs to P_a if and only if x has label a.

We now associate trees with directed graphs, called their *unfoldings.* Let G be a directed graph, possibly with multiple edges, given by a 4-tuple $\langle V, E, \mathbf{src}, \mathbf{tgt}\rangle$ where V is the set of vertices, E is the set of edges, **src** and **tgt** are total mappings from E and V that define respectively the *source* and the *target* of every edge. A *path* in G is a finite sequence of edges $(e_1, \ldots, e_n)$ such that $\mathbf{tgt}(e_i) = \mathbf{scr}(e_{i+1})$ for every $i = 1, \ldots, n-1$. The *origin* of this path is $\mathbf{src}(e_1)$ and its *end* is $\mathbf{tgt}(e_n)$. We shall denote by $\mathbf{Paths}(G, x)$ the set of paths in G that have origin x. (We put in this set the empty sequence and consider it as a path with origin and end x.) The *unfolding of G from a vertex x* is the tree $\langle \mathbf{Paths}(G, x), \mathbf{Suc}\rangle$ where (p, p') belongs to **Suc** if and only if p' extends p by exactly one edge. Figs. 3 and 4 below in Section 4 show a directed a graph and one of its unfoldings.

We now introduce trees equipped with edge labels distinguishing the different successors of the nodes. We let D be a finite set of labels, called the set of *directions.* A *D-tree* is a tree, the successor relation of which is the union of pairwise disjoint relations $\mathbf{Suc}_d$ for d in D, in such a way that for every x in N and d in D, there is at most one node y such that (x, y) belongs to $\mathbf{Suc}_d$. This node y will be called the *d-successor* of x; if one considers t as a graph, one can consider d as a label attached to the edge (x, y). Note that every node x has at most $\mathbf{Card}(D)$ successors. A D-tree will be given by a relational structure $t = \langle N, (\mathbf{Suc}_d)_{d\in D}\rangle$ where (x, y) belongs to $\mathbf{Suc}_d$ if and only if y is the d-successor of x. For every node x of t, we let $w(x)$ denote the word in D^* consisting of the sequence of labels of the edges forming the access path of x. It is clear that w is a one-to-one mapping of N onto a prefix closed language included in D^*. By this bijection $(x, y) \in \mathbf{Suc}_d$ if and only if $w(y) = w(x)\, d$.

We shall make a special use of the $[k]$-tree corresponding in this way to the language $[k]^*$. (We recall that $[k]$ denotes $\{1, \ldots, k\}$.) We shall call it the *complete k-ary* tree. We shall denote by $\mathbb{T}$ the complete binary tree.

We shall also use ordered trees. An *ordered tree* is a triple $\langle N, \mathbf{Suc}, \leqslant\rangle$ such that $\langle N, \mathbf{Suc}\rangle$ is a tree and $\leqslant$ is a partial order on N such that:

(1) two nodes are related by $\leqslant$ if and only if they have the same predecessor,

(2) each set $\mathbf{Suc}(x)$ is linearly ordered with respect to $\leqslant$ and has the order type ω if it is infinite.

By fixing a linear order $\leqslant'$ on D, one can make a D-tree into an ordered tree: it suffices to define a partial order on nodes as follows:

$x \leqslant y$ if and only if, either $x = y$ or x is the d-successor and y is the d'-successor of some node z, for some d and d' in D such that $d \leqslant' d'$.

There is also a bijection between the set of ordered trees and a certain set of binary trees. We let $D = \{\mathbf{first}, \mathbf{next}\}$. With an ordered tree $t = \langle N, \mathbf{Suc}, \leqslant \rangle$ we associate the D-tree $\mathbf{bin}(t) = \langle N, \mathbf{Suc}_{\mathbf{first}}, \mathbf{Suc}_{\mathbf{next}} \rangle$ such that:

$$\mathbf{Suc}_{\mathbf{first}} = \{(x, y) \mid y \text{ is the first element in the set } \mathbf{Suc}(x)\}$$
(first means smallest with respect to $\leqslant$)

$$\mathbf{Suc}_{\mathbf{next}} = \{(x, y) \mid x \text{ and } y \text{ belong to } \mathbf{Suc}(z) \text{ for some } z$$
and y follows x in the linear order $\leqslant$ on $\mathbf{Suc}(z)\}$.

Note that $\mathbf{bin}(t)$ belongs to the set $\mathbb{S}$ of D-trees, the root of which has no **next**-successor. We now define a mapping from $\mathbb{S}$ to ordered trees that will turn out to be the inverse to **bin**. With a D-tree in $\mathbb{S}$ of the form $\langle N, \mathbf{Suc}_{\mathbf{first}}, \mathbf{Suc}_{\mathbf{next}} \rangle$, we associate the ordered tree $\mathbf{flat}(t) = \langle N, \mathbf{Suc}, \leqslant \rangle$ where $\mathbf{Suc} = \mathbf{Suc}_{\mathbf{first}}(\mathbf{Suc}_{\mathbf{next}})^*$ and $x \leqslant y$ if and only if (x, y) belongs to $(\mathbf{Suc}_{\mathbf{next}})^*$.

Lemma 1.2. *The mappings* **bin** *and* **flat** *are MS-definable. They are one-to-one and inverse of each other.*

The proof consists in easy verfications. We only illustrate the transformations with a picture. The left part of Fig. 1 shows an ordered tree t (where the successors of nodes are ordered from left to right in a natural way) and the right part shows the D-tree $\mathbf{bin}(t)$, with the following convention: its **first**-edges are oriented top-down to the left and its **next**-edges are oriented horizontally to the right.

We finally recall the definition of trees built over ranked alphabets. A *ranked alphabet of function symbols* is an alphabet A, given with a *rank mapping* ρ *from* A *into* $\mathbb{N}$. Let k be the maximal rank of a symbol in A. A *tree over* A is a node labelled $[k]$-tree

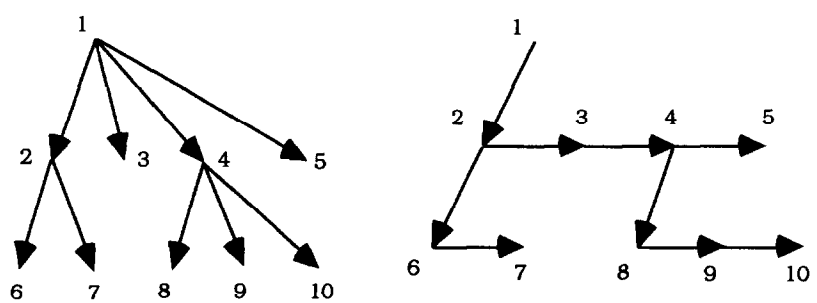

Fig 1.

of the form $t = \langle N, \mathbf{Suc}_1, \ldots, \mathbf{Suc}_k, (P_a)_{a\in A}\rangle$ such that the followingg conditions hold:

(1) every node x belongs to one and only one set P_a, and the corresponding a is called the *label* of x,

(2) if m is the rank of the label of a node x, then x has an i-successor for each $i = 1, \ldots, m$ and no i-successor for any $i > m$.

We let $\mathbf{T}^\infty$(A) denote the set of such trees, and by $\mathbf{T}(A)$ the subset of those that are finite. The trees in $\mathbf{T}(A)$ correspond in a well-known way to finite terms built over A, considered as a set of function symbols. If A is a ranked alphabet with all symbols of rank $k > 0$, then the trees in $\mathbf{T}^\infty(A)$ are formed by attaching labels from A to the nodes of the complete k-ary tree, hence they are nothing but total mappings from $[k]^*$ to A. If all symbols in A have rank 2, we denote $\mathbf{T}^\infty(A)$ by $\mathbb{T}(A)$.

2. Automata that define finite words or trees

Let A be a finite alphabet. *An automaton over* A is a relational structure

$$\mathscr{A} = \langle \mathscr{Q}, (R_a)_{a\in A}, R_\varepsilon, I, F\rangle$$

consisting of a *possibly infinite* domain $\mathscr{Q}$ (usually called the *set of states*), of unary relations $I \subseteq \mathscr{Q}$ and $F \subseteq \mathscr{Q}$ representing the sets of initial and final states, and of binary relations R_ε and R_a (for $a \in A$) representing the transitions on inputs, respectively, ε (the empty word) and a. We denote by $L(\mathscr{A})$ the language (a subset of A^*) *defined* (or *recognized*) by $\mathscr{A}$. If $\mathscr{A}$ is *finite* (i.e., if $\mathscr{Q}$ is finite) then $L(\mathscr{A})$ is a regular language.

Lemma 2.1. *The emptiness of* $L(\mathscr{A})$ *is an MS-definable property of* $\mathscr{A}$.

Proof. It suffices to observe that $L(\mathscr{A}) = \emptyset$ if and only if the transitive closure of $R_\varepsilon \cup \bigcup\{R_a \mid a \in A\}$ contains no pair (q, q') with $q \in I$ and $q' \in F$. This can be expressed in MS logic since the transitive closure of an MS-definable relation is MS-definable. We recall the proof of this last fact. Let R be a binary relation on a set $\mathscr{Q}$. A subset X of $\mathscr{Q}$ is *R-closed*, if it contains all elements y of $\mathscr{Q}$ such that $R(x, y)$ holds for some x in X. For any x, y in $\mathscr{Q}$, the pair (x, y) belongs to the transitive closure of R if and only if there exists an element z such that $R(x, z)$ holds and y belongs to every R-closed subset of $\mathscr{Q}$ containing z. This characterization of transitive closure is expressible by an MS formula using a universal quantification on all R-closed sets where, of course, we assume that R is given by an MS formula. □

Let $\mathscr{B} = \langle \mathscr{Q}', (R'_a)_{a\in A}, I', F'\rangle$ be another automaton over A, without ε-transitions. Let

$$\mathscr{A} \times \mathscr{B} = \langle \mathscr{Q} \times \mathscr{Q}', (R_a \times R'_a)_{a\in A}, R_\varepsilon \times \Delta_{\mathscr{Q}'}, I \times I', F \times F'\rangle$$

be the *product-automaton* of $\mathscr{A}$ and $\mathscr{B}$. (If $R \subseteq \mathscr{Q} \times \mathscr{Q}$ and $R' \subseteq \mathscr{Q}' \times \mathscr{Q}'$, then $R \times R'$ is the binary relation on $\mathscr{Q} \times \mathscr{Q}'$ such that $((p, q), (p', q')) \in R \times R'$ if and only if $(p, p') \in R$ and $(q, q') \in R'$. We denote by $\Delta_{\mathscr{Q}'}$ the *diagonal* relation, namely $\{(q, q) \mid q \in \mathscr{Q}'\}$.) It is well-known that $L(\mathscr{A} \times \mathscr{B}) = L(\mathscr{A}) \cap L(\mathscr{B})$.

Lemma 2.2. *For every fixed finite automaton $\mathscr{B}$ without ε-transitions, the mapping $\mathscr{A} \mapsto \mathscr{A} \times \mathscr{B}$ is an MS-definable transduction.*

Proof. Easy verification. The complete construction is given in [10]. □

Theorem 2.3. *The transduction associating with an automaton the language it defines is MS-compatible. In other words, for every MS-formula φ, the properties $L(\mathscr{A}) \models_3 \varphi$ and $L(\mathscr{A}) \models \varphi$ of an automaton $\mathscr{A}$ are MS-definable. There is an algorithm that constructs from φ the corresponding MS formulas*

Proof. By the basic theorem of Büchi and Elgot saying that a set of words is regular if and only if it is MS-definable [21, Theorem 3.2], one can construct from φ a finite automaton $\mathscr{B}$ without ε-transitions such that, for every $w \in A^*$

$$\|w\| \models \varphi \text{ if and only if } w \in L(\mathscr{B}).$$

(A word w is represented by the logical structure $\|w\|$ as explained in Section 1.) It follows that

$$\begin{aligned} &L(\mathscr{A}) \models_3 \varphi \\ &\quad \text{if and only if} \quad L(\mathscr{A}) \cap L(\mathscr{B}) \neq \emptyset \\ &\quad \text{if and only if} \quad L(\mathscr{A} \times \mathscr{B}) \neq \emptyset. \end{aligned}$$

Since the mapping $\mathscr{A} \mapsto \mathscr{A} \times \mathscr{B}$ is an MS-definable transduction (Lemma 2.2), and since the property $L(\mathscr{C}) \neq \emptyset$ of an automaton $\mathscr{C}$ is MS-definable by Lemma 2.1, it follows from Proposition 1.1 that $L(\mathscr{A} \times \mathscr{B}) \neq \emptyset$ is MS-definable as a property of $\mathscr{A}$. Note that for every given formula φ, the automaton $\mathscr{B}$ is finite, fixed and can be effectively constructed. For the case of $L(\mathscr{A}) \models \varphi$ one constructs similarly $\mathscr{B}'$ from $\neg\varphi$ and one gets

$$\begin{aligned} &L(\mathscr{A}) \models \varphi \\ &\quad \text{if and only if} \quad L(\mathscr{A}) \cap L(\mathscr{B}') = \emptyset \\ &\quad \text{if and only if} \quad L(\mathscr{A} \times \mathscr{B}') = \emptyset. \end{aligned}$$

The latter condition is also MS-definable as a property of $\mathscr{A}$. □

Theorem 2.3 proves in particular that for every regular language $K \subseteq A^*$ the property $L(\mathscr{A}) \subseteq K$ of a finite or infinite automaton $\mathscr{A}$ is MS-definable. For the opposite inclusion, we have the following fact.

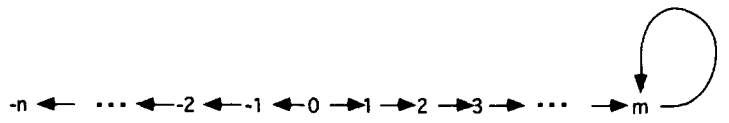

Fig 2.

Proposition 2.4. *Let K be a fixed regular language. The property $L(\mathscr{A}) \supseteq K$ of a finite automaton $\mathscr{A}$ is not MS-definable in general. It is on the class of deterministic automata.*

Proof. We let $\mathscr{A}_{n,m}$ be the nondeterministic automaton shown on Fig. 2, where the final states are $m, 0, -1, -2, \ldots, -n$. The initial state is 0. All arrows indicate transitions with input letter a.

Clearly $L(\mathscr{A}_{n,m}) = \{\varepsilon, a, a^2, \ldots a^n, a^m, a^{m+1}, a^{m+2}, \ldots\}$ hence $L(\mathscr{A}_{n,m}) \supseteq a^*$ if and only if $m \leqslant n + 1$. One can construct an MS-definable transduction mapping a word of the form $a^n bc^m$ to the automaton $\mathscr{A}_{n,m}$; if an MS-formula on $\mathscr{A}_{n,m}$ could express that $L(\mathscr{A}_{n,m}) \supseteq a^*$, then the language $\{a^n bc^m \mid m \leqslant n + 1\}$ would be MS-definable by Proposition 1.1, hence regular by Büchi–Elgot's theorem, which is not the case.

Here is a definition scheme without parameters for this transduction. We take $\Delta = (\varphi, \psi, \theta_{R_a}, \theta_I, \theta_F)$ where φ expresses that the input structure of the form $S = \langle \mathbf{D}_S, \mathbf{suc}, \mathbf{p}_a, \mathbf{p}_b, \mathbf{p}_c \rangle$ indeed represents a word of the form $a^n bc^m$ for some n and m. The formula ψ is the constant **true** (hence $\mathbf{D}_S$ will be the domain of the constructed structure, i.e., the set of states of the automaton). The other formulas are such that:

$\theta_{R_a}(x_1, x_2)$ holds if and only if either $\mathbf{suc}(x_2, x_1)$ and $\mathbf{p}_a(x_2)$
or $\mathbf{suc}(x_1, x_2)$ and $\mathbf{p}_c(x_2)$
or $x_1 = x_2$ and x_1 has no successor.

$\theta_I(x_1)$ is the formula $\mathbf{p}_b(x_1)$.

$\theta_F(x_1)$ holds if and only if either $\mathbf{p}_a(x_1)$ or $\mathbf{p}_b(x_1)$
or x_1 has no successor.

Let us now assume that $\mathscr{A}$ is a *deterministic* automaton: an MS-definable transduction can be constructed that associated with $\mathscr{A}$ another deterministic automaton $\mathscr{A}'$, such that $L(\mathscr{A}') = A^* - L(\mathscr{A})$. Hence,

$$L(\mathscr{A}) \supseteq K \text{ if and only if } L(\mathscr{A}') \subseteq A^* - K$$

and this latter property is MS-definable on $\mathscr{A}$ if $\mathscr{A}$ is deterministic by Proposition 1.1 and the proof technique of Theorem 2.3. □

Corollary 2.5. *There is no MS-definable transduction that associates with a finite nondeterministic automaton an equivalent deterministic one, or an automaton defining the complement language.*

Proof. Immediate from Propositions 2.4 and 1.1. □

2.1. Tree-automata

The case of automata defining finite trees is very similar to that of automata defining words. We let A be a ranked alphabet of function symbols, and we recall that the set of *finite trees over* A is denoted by $\mathbf{T}(A)$. A tree t in $\mathbf{T}(A)$ is defined (see Section 1) as a relational structure of the form $\langle N, \mathbf{Suc}_1, \mathbf{Suc}_2, \ldots, \mathbf{Suc}_k, (P_a)_{a\in A)}\rangle$ where N is the set of nodes, $\mathbf{Suc}_i(x, y)$ holds if and only if y is the ith successor of x, and $P_a(x)$ holds if and only if a is the label of x.

A *tree automaton over* A is a relational structure of the form

$$\mathscr{A} = \langle \mathscr{Q}, (R_a)_{a\in A}, F\rangle$$

where $\mathscr{Q}$ is the domain (the set of states), R_a is a $(\rho(a)+1)$-ary relation on $\mathscr{Q}$, F is a subset of $\mathscr{Q}$ called the set of accepting (or final) states. The *set of finite trees defined (or recognized)* by $\mathscr{A}$ is the subset $L(\mathscr{A})$ of $\mathbf{T}(A)$ defined as follows. We need the notion of a *run of* $\mathscr{A}$ on a tree $t \in \mathbf{T}(A)$. Let $t \in \mathbf{T}(A)$ and N be the set of its nodes. A run of $\mathscr{A}$ on t is a mapping $r: N \to \mathscr{Q}$ such that:

if $x \in N$ has label a of rank 0, then $r(x) \in R_a$,
if $x \in N$ has label a of rank k and successors $x_1, \ldots, x_k$
then $(r(x_1), \ldots, r(x_k), r(x)) \in R_a$.

We let $\mathbf{root}(t)$ denote the root of a tree t and we call $r(\mathbf{root}(t))$ the *root-state* of the run r. A run is accepting if $r(\mathbf{root}(t)) \in F$. We let $L(\mathscr{A})$ be the set of trees in $\mathbf{T}(A)$ having an accepting run.

Lemma 2.6. *The property* $L(\mathscr{A}) \neq \emptyset$ *is* MS-*definable on* $\mathscr{A}$.

Proof. Let X be the set of states q of $\mathscr{A}$ that are the root-states of some runs on some trees in $\mathbf{T}(A)$. This set is the least set Y (for inclusion) such that

(1) $R_a \subseteq Y$ for every a of rank 0,

(2) for every $k > 0$, for every $x_1, \ldots, x_k \in Y$ if $(x_1, \ldots, x_k, y) \in R_a$ for some $a \in A$ of rank k, then $y \in Y$.

These two conditions can be written in first-order logic. If $P(Y)$ is an MS-property of sets Y, then it can be expressed in MS-logic that a set Z is the least set (for inclusion) that satisfies property P. It follows that the set X can be characterized by an MS-formula. Since the condition $L(\mathscr{A}) \neq \emptyset$ is equivalent to $X \cap F \neq \emptyset$, this latter condition can also be expressed by an MS-formula on $\mathscr{A}$. □

As for automata defining words, one can define for any two tree automata $\mathscr{A}$ and $\mathscr{B}$ over A a product automaton $\mathscr{A} \times \mathscr{B}$ such that $L(\mathscr{A} \times \mathscr{B}) = L(\mathscr{A}) \cap L(\mathscr{B})$ [15]. Furthermore, for any fixed finite automaton $\mathscr{B}$ the transduction $\mathscr{A} \mapsto \mathscr{A} \times \mathscr{B}$ is MS-definable (by an immediate extension of the construction of Lemma 2.2). By using the theorem by Thatcher et al. saying that a subset of $\mathbf{T}(A)$ is MS-definable if and only

if it is definable by a finite tree automaton [21, Theorem 11.1]), one gets, essentially by the same proof as in Theorem 2.3 the following theorem.

Theorem 2.7. *The transduction that associates with a finite or infinite tree automaton over a ranked alphabet A the subset of* $\mathbb{T}(A)$ *it defines is MS-compatible. Hence, for every MS-formula* φ *the properties* $L(\mathcal{A}) \models_\exists \varphi$ *and* $L(\mathcal{A}) \models \varphi$ *of a tree automaton* $\mathcal{A}$ *are MS-definable. There is an algorithm that constructs from* φ *the corresponding MS formulas.*

3. Automata defining infinite trees

We denote by $\mathbb{T}$ the complete binary tree $\langle \{1,2\}^*, \mathbf{Suc}_1, \mathbf{Suc}_2 \rangle$ where, for $i = 1, 2$, $\mathbf{Suc}_i$ is the binary relation $\{(u, ui) \mid u \in \{1,2\}^*\}$. The root of $\mathbb{T}$ is ε, the empty word and w is the left (resp. right) successor of u if and only if (u, w) belongs to $\mathbf{Suc}_1$ (resp. to $\mathbf{Suc}_2$). If A is a finite set of function symbols, all of rank 2, then $\mathbb{T}(A)$ denotes the set of trees built over A, also called *A-trees*, i.e., of labellings by symbols from A of the complete binary tree $\mathbb{T}$. These trees are nothing but mappings $\{1,2\}^* \to A$.

A *Rabin automaton over A of index n* is a relational structure

$$\mathcal{A} = \langle \mathcal{Q}, (R_a)_{a \in A}, F, (U_i)_{i=1,\ldots,n}, (L_i)_{i=1,\ldots,n} \rangle$$

where $\mathcal{Q}$ is the finite or infinite domain (the set of *states*), R_a is a ternary relation on $\mathcal{Q}$ for each a, and F, U_i, L_i for i = 1, ..., n are subsets of $\mathcal{Q}$. A *run* of $\mathcal{A}$ on $t \in \mathbb{T}(A)$ is a mapping $r: \{1,2\}^* \to \mathcal{Q}$ such that, for every $u \in \{1,2\}^*$, we have $(r(u1), r(u2), r(u)) \in R_a$ where $a = t(u) \in A$. This run is *accepting* if its root-state $r(\varepsilon)$ belongs to F and if, in the case where $n \geqslant 1$, the following condition (C) also holds:

(C) for every infinite word $w \in \{1,2\}^\omega$ (i.e., every infinite branch w in $\mathbb{T}$) there is an integer $i \in \{1, \ldots, n\}$ such that $\mathbf{In}(r, w) \cap U_i \neq \emptyset$ and $\mathbf{In}(r, w) \cap L_i = \emptyset$

where $\mathbf{In}(r, w)$ is the set of states q such that $q = r(u)$ for infinitely many finite prefixes u of w. We let $L(\mathcal{A})$ denote the set of A-trees on which $\mathcal{A}$ has an accepting run. For every $q \in \mathcal{Q}$, we let also $L(\mathcal{A}, q)$ denote the set of A-trees on which there exists a run with root-state q that satisfies condition (C) (when $n \geqslant 1$). Hence, $L(\mathcal{A}) = \bigcup \{L(\mathcal{A}, q) \mid q \in F\}$. A subset of $\mathbb{T}(A)$ is *Rabin-recognizable* if it is of the form $L(\mathcal{A})$ for some Rabin automaton $\mathcal{A}$ having finitely many states.

Note that a Rabin automaton of index 0 over A is nothing but a tree automaton as defined in Section 2, where all symbols of A are of rank 2. For such an automaton, condition (C) is not required. However, the automata is Section 2 accept finite trees whereas here we consider automata accepting infinite ones.

Theorem 3.1. *Let A be a finite alphabet of symbols of rank 2. The transduction associating with a finite or infinite tree automaton* $\mathcal{A}$ *over A the set of A-trees it defines is*

MS-compatible. In other words, if K is a Rabin recognizable set of A-trees, then the property of a finite or infinite tree automaton $\mathscr{A}$ over A that $L(\mathscr{A}) \cap K \neq \emptyset$ is MS-definable. An MS formula expressing it can be effectively constructed from a finite Rabin automaton defining K.

The proof uses many additional definitions, lemmas and results from Niwinski [18, 19]. We give it in the appendix in order not to break the exposition. Applications to transition systems and algebraic trees will be given in the next sections. The following result is almost a generalization of Theorem 3.1, except that it concerns only finite automata $\mathscr{A}$ whereas Theorem 3.1 concerns also infinite automata.

Theorem 3.2. *Let n be an integer and A be a finite alphabet of symbols of rank 2. The transduction associating with a finite Rabin automaton over A of index n the set of A-trees it defines is MS-compatible.*

Proof. We let K be an MS-definable set of A-trees. It is Rabin recognizable. We need only prove that the property $L(\mathscr{A}) \cap K \neq \emptyset$ of a finite Rabin automaton $\mathscr{A}$ over A of index n is MS-definable.

We let $B_n = \{\mathbf{u}, \mathbf{l}, \perp\}^n$ and $A' = A \times B_n$. We transform a Rabin automaton $\mathscr{A} = \langle \mathscr{Q}, (R_a)_{a \in A}, F, (U_i)_{i=1,\ldots,n} (L_i)_{i=1,\ldots,n} \rangle$ into the tree automaton $\mathscr{A}' = \langle \mathscr{Q}, (R_{a'})_{a' \in A'}, F \rangle$ such that for $a \in A$ and $b \in B_n$:

$R_{(a,b)}$ is the set of triples (q_1, q_2, q) in R_a such that,
for every $i = 1, \ldots, n$ the ith component of b is
- $\mathbf{u}$ if $q \in U_i - L_i$,
- $\mathbf{l}$ if $q \in L_i$,
- $\perp$ if $q \notin U_i \cup L_i$.

We let $\pi_1 : A' \to A$ be the first projection ($\pi_1(a, b) = a$) and we let π_1 denote also its extension: $\mathbb{T}(A') \to \mathbb{T}(A)$, i.e., $\pi_1(t) = \pi_1 \circ t$ for t in $\mathbb{T}(A')$. We let π_2 denote the other projection, extended similarly: $\mathbb{T}(A') \to \mathbb{T}(B_n)$.

We let $D \subseteq \mathbb{T}(B_n)$ be the set of B_n-trees t such that, for every infinite branch of t, there exists an integer $i \in [1, n]$ such that

(1) infinitely many nodes of this branch have a label, the ith component of which is $\mathbf{u}$,

(2) only finitely many nodes of this branch have a label, the ith component of which is $\mathbf{l}$.

Claim. $L(\mathscr{A}) \cap K = \pi_1(L(\mathscr{A}') \cap \pi_2^{-1}(D) \cap \pi_1^{-1}(K))$.

Note in particular that we have $L(\mathscr{A}) = \pi_1(L(\mathscr{A}') \cap \pi_2^{-1}(D))$. The idea of the construction of $\mathscr{A}'$ and D is the following. A tree t belongs to $L(\mathscr{A})$ if and only if a state of $\mathscr{A}$ can be associated with each of its nodes in such a way that conditions of two types are satisfied. Some conditions are local: they express that the chosen states form a run with root-state in F. The others are global: they concern the memberships

in the sets U_i and L_i of the states that occur infinitely many times on branches. The information concerning these memberships can be "guessed" in the form of n-tuples in B_n associated with each node. That the guess is correct with respect to the considered run is a local condition, and the automaton $\mathscr{A}'$ performs this verification. The condition concerning infiniteness of repetitions on each branch is a global condition, that now does not depend on $\mathscr{A}$ or $\mathscr{A}'$. To summarize, we can separate the verification that t belongs to $L(\mathscr{A})$ into three steps:

(1) a guess at each node of some information depending on the index n of $\mathscr{A}$,

(2) a verification of the "local" correctness of the guess, by an automaton $\mathscr{A}'$ of index 0 constructed from $\mathscr{A}$,

(3) a verification of the "global", "infinitary" condition, which can be done by a fixed Rabin automaton of index n, independent of $\mathscr{A}$, that takes the guesses as input.

Proof of the claim. ($\supseteq$) Let $t \in L(\mathscr{A}') \cap \pi_2^{-1}(D) \cap \pi_1^{-1}(K)$ and r be a run of $\mathscr{A}'$ on t. Then r is a run of $\mathscr{A}$ on $\pi_1(t)$, with a root-state in F. Let w be an infinite branch of t. Let $\mathscr{Q}'$ be the set of states that occur infinitely many times on this branch. Let $i \in [1, n]$ be some index such that on this branch infinitely many nodes have a label (a, b) with the ith element b_i of b equal to $\mathbf{u}$ and only finitely many with b_i equal to $\mathbf{l}$. Then,

(1) On the infinite set of nodes such that $b_i = \mathbf{u}$, some state $q \in \mathscr{Q}$ occurs infinitely many times (because $\mathscr{Q}$ is assumed to be finite) and by the construction of $\mathscr{A}'$ this state q belongs to $U_i - L_1$:

(2) Let us now consider a state q that occurs infinitely many times. If this state belongs to L_i then the component b_i is equal to $\mathbf{l}$ infinitely many times. But this contradicts the initial assumption on i. This proves that $q \notin L_i$.

Hence this branch fulfills the accepting condition of $\mathscr{A}$. Hence $\pi_1(t) \in L(\mathscr{A})$. Since $t \in \pi_1^{-1}(K)$ we also have $\pi_1(t) \in K$ hence $\pi_1(t) \in L(\mathscr{A}) \cap K$.

($\subseteq$) Let conversely $t \in L(\mathscr{A}) \cap K$. Let r be an accepting run of $\mathscr{A}$ on t. For each node x of t, we let $b \in B_n$ be the sequence $(b_1, \ldots, b_n)$ such that for all $i = 1, \ldots, n$:

$$b_i = \mathbf{u} \quad \text{if } r(x) \in U_i - L_i,$$

$$b_i = \mathbf{l} \quad \text{if } r(x) \in L_i,$$

$$b_i = \perp \quad \text{if } r(x) \notin U_i \cup L_i,$$

and we replace the node label a of x by (a, b). In this way we define a tree $t' \in \mathbb{T}(A')$ such that $\pi_1(t') = t$ and $t' \in L(\mathscr{A}')$. Clearly $t' \in \pi_1^{-1}(K)$, hence we need only prove that $t' \in \pi_2^{-1}(D)$. Since the run r of $\mathscr{A}$ on t is accepting, for every infinite branch w of t there is an index i such that the set P of states of $\mathscr{A}$ that occur infinitely often on w satisfies $P \cap U_i \neq \emptyset$ and $P \cap L_i = \emptyset$. From the definition of t' the ith element b_i of the second component of the labels of the nodes on the branch w is equal to $\mathbf{u}$ infinitely many times and equal to $\mathbf{l}$ finitely many times (we use here the fact that L_i is finite). Hence $t' \in \pi_2^{-1}(D)$. It follows that $t' \in L(\mathscr{A}') \cap \pi_2^{-1}(D) \cap \pi_1^{-1}(K)$ as was to be proved. □

Proof of Theorem 3.2 (*conclusion*). It follows that $L(\mathscr{A}) \cap K \neq \emptyset$ if and only if $L(\mathscr{A}') \cap K' \neq \emptyset$ where K' is the Rabin recognizable set $\pi_2^{-1}(D) \cap \pi_1^{-1}(K)$. This latter property of $\mathscr{A}'$ is MS-definable by Theorem 3.1 and so is the property $L(\mathscr{A}) \cap K \neq \emptyset$ since the transduction $\mathscr{A} \mapsto \mathscr{A}'$ is MS-definable, and by Proposition 1.1. □

4. Transition systems

Considering the unfolding of a finite or infinite directed graph as its behaviour, we establish (under some additional conditions) that every MS property of the behaviour of a graph can be expressed by an MS formula *on the graph itself*. In view of applications, we shall formulate this result in terms of *transition systems*, which are nothing but directed graphs with some information attached to vertices and edges.

Let n and m be nonnegative integers. A *transition system of type* (n, m) is a tuple of the form

$$\mathscr{R} = \langle S, T, \mathbf{init}, \mathbf{src}, \mathbf{tgt}, P_1, \ldots, P_n, \mathscr{Q}_1, \ldots, \mathscr{Q}_m \rangle$$

where S and T are two sets, called respectively the set of *states* and the set of *transitions*, **init** is a state called the *initial state*, **src** and **tgt** are two mappings from T to S that define respectively the *source* and the *target* of a transition, $P_1, \ldots, P_n$ are subsets of S and $\mathscr{Q}_1, \ldots, \mathscr{Q}_m$ are subsets of T that specify properties of states and of transitions respectively. We shall denote by $\mathscr{T}(n, m)$ the class of transition systems of type (n, m).

A *path* in $\mathscr{R}$ is a finite sequence of transitions $(t_1, t_2, \ldots, t_k)$ such that the source of t_1 is the initial state **init** and $\mathbf{tgt}(t_i) = \mathbf{src}(t_{i+1})$ for every $i = 1, \ldots, k-1$. The empty sequence is considered as a path starting and ending at the initial state. We shall denote by $\mathbf{Paths}(\mathscr{R})$ the set of paths in $\mathscr{R}$ and by $\prec$ the *prefix order* on $\mathbf{Paths}(\mathscr{R})$. We let $p \prec_1 p'$ if and only if $p \prec p'$ and p' has exactly one more transition than p. (Hence, $\prec$ is the reflexive and transitive closure of $\prec_1$.) For every property P_i of states we let P_i^* be the property of a path saying that the last state of this path satisfies P_i. For every property $\mathscr{Q}_i$ of transitions we let $\mathscr{Q}_i^*$ if be the property of a path saying that the last transition of this path satisfies $\mathscr{Q}_i$. We let the *behaviour* of $\mathscr{R}$ be the structure

$$\mathbf{bhv}(\mathscr{R}) = \langle \mathbf{Paths}(\mathscr{R}), \prec_1, P_1^*, \ldots, P_n^*, \mathscr{Q}_1^*, \ldots, \mathscr{Q}_m^* \rangle$$

which is a tree (see Section 1) with information attached to its nodes.

Fig. 3 shows a transition system represented as a labelled graph in the usual way [1]. Integers name the transitions. The initial state is the common source of transitions 1, 2, 3. The target of transition 3 satisfies property P and transition 7 satisfies property $\mathscr{Q}$. Fig. 4 shows the behaviour of this transition system. The marks P^* and $\mathscr{Q}^*$ show the nodes of the tree (i.e., the paths in $\mathscr{R}$) satisfying P^* and $\mathscr{Q}^*$.

We conjecture that every MS-property of the behaviour of a transition system $\mathscr{R}$ is equivalent to an MS-property of an appropriate relational structure representing

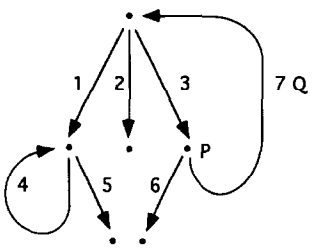

Fig 3.

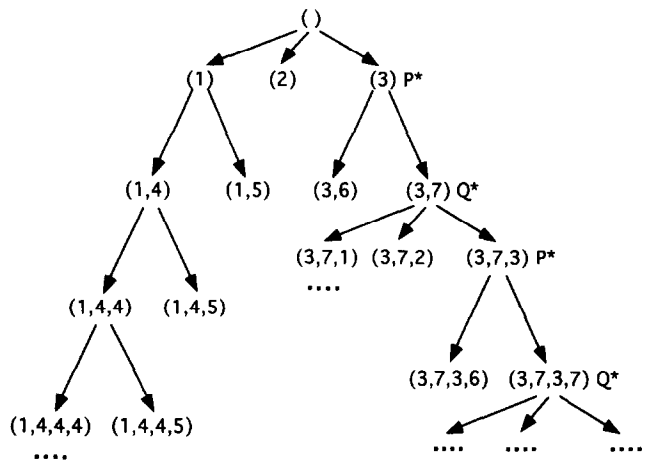

Fig 4.

$\mathscr{R}$ itself. (See Conjecture 4.2 below for a more precise statement.) We shall prove special cases of this conjecture.

A transition system is *deterministic* if it is of the form

$$\mathscr{R} = \langle S, T, \mathbf{init}, \mathbf{src}, \mathbf{tgt}, P_1, \ldots, P_n, \mathscr{Q}_1, \ldots, \mathscr{Q}_m \rangle$$

where $P_1, \ldots, P_n \subseteq S$, $\mathscr{Q}_1, \ldots, \mathscr{Q}_m \subseteq T$ and the following conditions hold:

(1) $\mathscr{Q}_1, \ldots, \mathscr{Q}_m$ form a partition of T,

(2) for every state x, for every $i \in \{1, \ldots, m\}$ there is at most one transition in $\mathscr{Q}_i$ with source x.

We say that $\mathscr{R}$ as above is *complete* if, instead of (2), we have the stronger condition:

(2') for every state x and every $i = 1, \ldots, m$, there exists one and only one transition in $\mathscr{Q}_i$ with source x.

In order to specify the parameter m, we shall say that a system $\mathscr{R}$ as above is *m-ary*. We shall denote by $\mathscr{D}(n,m)$ the class of deterministic transition systems of type (n, m).

The behaviour of a transition system $\mathscr{R}$ in $\mathscr{D}(n,m)$ is an $[m]$-tree, where $[m] = \{1, \ldots, m\}$ is used as a set of directions (see Section 1; the directions are edge labels that distinguish the different outgoing edges). The subsets $P_1, \ldots, P_n$ is S define subsets $P_1^*, \ldots, P_n^*$ of the set of nodes of the tree $\mathbf{bhv}(\mathscr{R})$. Hence $\mathbf{bhv}(\mathscr{R})$ will be handled as the structure:

$$\langle \mathbf{Paths}(\mathscr{R}), \mathbf{Suc}_1, \ldots, \mathbf{Suc}_m, P_1^*, \ldots, P_n^* \rangle,$$

which differs slightly from the one defined initially. Here, $\mathbf{Suc}_i(p, p')$ holds if and only if $p \prec_1 p'$ and the last transition of p' belongs to $\mathscr{Q}_i$. (We recall that $p \in P_i^*$ if and only if the end state of p belongs to P_i.)

Fig. 5 shows an infinite, complete transition system in $\mathscr{D}(1,2)$ and Fig. 6 shows its behaviour. The states marked a are those that satisfy P_1. The state marked **init** is the

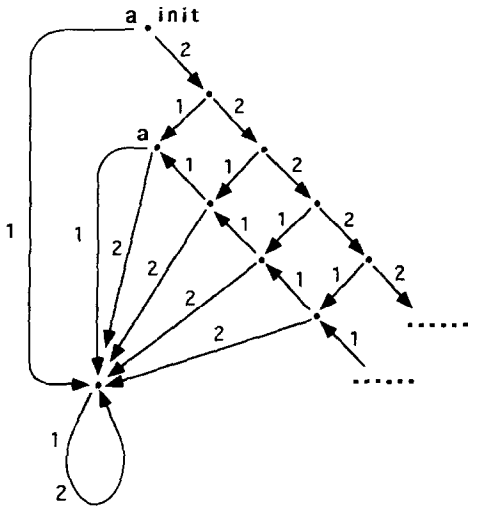

Fig 5.

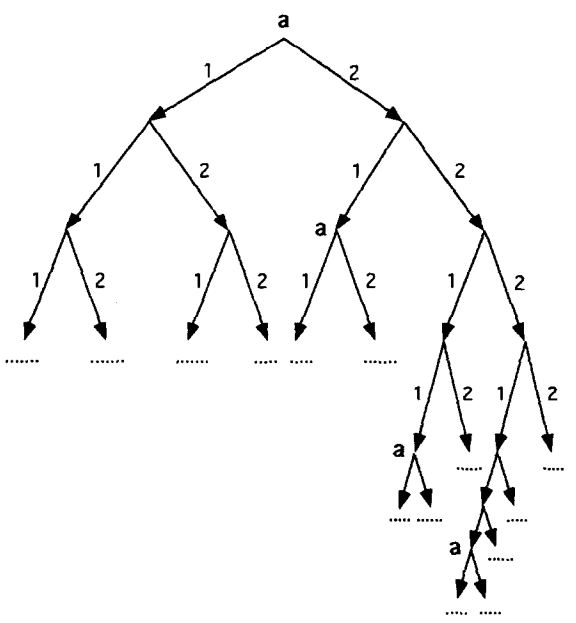

Fig 6.

initial one. In Fig. 6, the nodes of the tree marked a are those satisfying P_1^*. These nodes are characterized by an access path of the form 2^n1^n.

A transition system $\mathscr{R}$ in $\mathscr{D}(n,m)$ is characterized up to isomorphism by the relational structure:

$$|\mathscr{R}|_1 := \langle S, \mathbf{Init}, R_1, \ldots, R_m, P_1, \ldots, P_n \rangle$$

where **Init** = {**init**} (a subset of S), and, for $i = 1, \ldots, m$, the relation R_i is the set of pairs (**src**(x), **tgt**(x)) such that x is a transition satisfying $\mathcal{Q}_i$. One can consider $\mathcal{R}$ as a deterministic automaton with input letters $1, \ldots, m$, and n distinguished sets of states $P_1, \ldots, P_n$. One can also consider it as a directed labelled graph: see Fig. 5. The vertices of the graph are the states of $\mathcal{R}$. The behaviour of $\mathcal{R}$ is its unfolding (its tree of finite paths, see Section 1) from the vertex corresponding to the initial state.

In the following theorem, MS_1 denotes MS logic relative to the representation of a transition system $\mathcal{R}$ by the structure $|\mathcal{R}|_1$ allowing quantifications on sets of states but not on sets of transitions. Behaviours, which are "tree-shaped" transition systems are represented similarly.

Theorem 4.1. *For every $n, m \in \mathbb{N}$, the transduction* **bhv** is *MS_1-compatible on the class $\mathcal{D}(n,m)$.*

Proof. We first consider the case of a complete system $\mathcal{R}$ belonging to $\mathcal{D}(n,2)$. The behaviour of $\mathcal{R}$ is a structure of the form

$$\mathbf{bhv}(\mathcal{R}) = \langle \{1,2\}^*, \mathbf{Suc}_1, \mathbf{Suc}_2, P_1^*, \ldots, P_n^* \rangle$$

that we make into the tree:

$$h(\mathbf{bhv}(\mathcal{R})) = \langle \{1,2\}^*, \mathbf{Suc}_1, \mathbf{Suc}_2, (P_a)_{a \in A} \rangle$$

in $\mathbb{T}(A)$ where $A = \{\mathbf{true}, \mathbf{false}\}^n$ and for every $w \in \{1,2\}^*$, every $a \in A$:

$$w \in P_a \Leftrightarrow a = (a_1, \ldots, a_n) \text{ and for every } i = 1, \ldots, n, a_i = \mathbf{true} \text{ if and only if } w \in P_i^*.$$

From $|\mathcal{R}|_1 = \langle S, \mathbf{Init}, R_1, R_2, P_1, \ldots, P_n \rangle$ we define the tree-automaton

$$\mathcal{A}(\mathcal{R}) := \langle S, (R_a)_{a \in A}, \mathbf{Init} \rangle,$$

where for each $a = (a_1, \ldots, a_n)$ in A, the relation R_a is the set of triples (s_1, s_2, s) such that $R_1(s, s_1)$ and $R_2(s, s_2)$ hold and for every $i = 1, \ldots, n$, $a_i = \mathbf{true}$ if and only if $s \in P_i$. It is thus clear that $L(\mathcal{A}(R)) = \{h(\mathbf{bhv}(R))\}$.

Let us now consider the MS-formula φ expressing a property of the behaviour of systems in $\mathcal{D}(n,2)$. One can construct an MS-formula φ' such that

$$\mathbf{bhv}(\mathcal{R}) \models \varphi$$

if and only if $h(\mathbf{bhv}(\mathcal{R})) \models \varphi'$

if and only if $h(\mathbf{bhv}(\mathcal{R})) \in K$ (1)

if and only if $L(\mathcal{A}(\mathcal{R})) \cap K \neq \emptyset$ (2)

where K is the (Rabin recognizable) set of trees in $\mathbb{T}(A)$ that satisfy φ'. The equivalence of (1) and (2) holds because $L(\mathcal{A}(\mathcal{R})) = \{h(\mathbf{bhv}(\mathcal{R}))\}$. It follows from Theorem 3.1 that (2) is equivalent to $\mathcal{A}(\mathcal{R}) \models \theta$ for some MS-formula θ that one can construct from the

automaton defining K, and finally, that (2) is equivalent to $|\mathscr{R}|_1 \models \psi$ for some MS-formula ψ constructible from θ (by Proposition 1.1).

We now extend this proof to handle the case of a transition system $\mathscr{R} \in \mathscr{D}(n,2)$ that is not complete. The idea is to make $\mathscr{R}$ complete by adding an extra state (usually called a "sink"). Formally, we do not modify $\mathscr{R}$ but we let $A' = A \cup \{\#\}$ and we construct from $|\mathscr{R}|_1$ the complete deterministic binary tree-automaton

$$\mathscr{B}(\mathscr{R}) = \langle S \cup \{\bot\}, \{R_a\}_{a\in A}, R_\#, \mathbf{Init}\rangle$$

where $\bot$ is the new state, R_a is as in $\mathscr{A}(R)$ and $(s_1, s_2, s) \in R_\#$ if and only if

either $s_1 = s_2 = s = \bot$
or $s_1 = s_2 = \bot$ and $s \in S$ and
$R_1(s, v)$ and $R_2(s, v)$ both do not hold for any $v \in S$
or $s_1 = \bot, s_2, s \in S$, and
$R_2(s, s_2)$ holds whereas $R_1(s, v)$ does not hold for any $v \in S$
or vice versa by exchanging the indices 1 and 2.

It is clear that $L(\mathscr{B}(\mathscr{R}))$ consists of a unique tree in $\mathbb{T}(A')$ that we shall denote by $t(\mathscr{R})$ and that, by deleting the nodes with label $\#$, one obtains from this tree the tree $h(\mathbf{bhv}(\mathscr{R}))$. It is not hard to see that the mapping from $t(\mathscr{R})$ to $\mathbf{bhv}(\mathscr{R})$ is an MS-transduction, and so is the transformation from $|\mathscr{R}|_1$ to $\mathscr{B}(\mathscr{R})$. (We omit the routine details). The proof now goes as follows. From the MS-formula φ that one wants to test on $\mathbf{bhv}(\mathscr{R})$ for $\mathscr{R} \in \mathscr{D}(n,2)$, one constructs an MS-formula φ'' such that

$\mathbf{bhv}(\mathscr{R}) \models \varphi$
if and only if $t(\mathscr{R}) \models \varphi''$
if and only if $t(\mathscr{R}) \in K'$
if and only if $L(\mathscr{B}(\mathscr{R})) \cap K' \neq \emptyset$,

where we let K' be the Rabin recognizable set of trees in $\mathbb{T}(A')$ that satisfy φ''; the formula φ'' is obtained by Proposition 1.1 from the remark that the transformation from $t(\mathscr{R})$ to $\mathbf{bhv}(\mathscr{R})$ is MS-definable. One concludes that these conditions are MS-properties of $|\mathscr{R}|_1$ by Theorem 3.1 and Proposition 1.1, using the fact that the transformation from $|\mathscr{R}|_1$ to $\mathscr{B}(\mathscr{R})$ is MS-definable.

For the case of $\mathscr{R} \in \mathscr{D}(n,m)$, with $m > 2$, one can get the conclusion in two ways. Either by observing that Theorem 3.1 holds for m-ary trees and not only for binary ones, or by using Lemma 1.2 in order to reduce the general case to the one of binary trees. (This latter technique will be used in the proof of Theorem 4.3 below.) □

We would like to extend Theorem 4.1 to transition systems outside of the classes $\mathscr{D}(n,m)$. However, the relational structure $|\mathscr{R}|_1$ used to represent a transition system $\mathscr{R}$ in $\mathscr{D}(n,m)$ is no longer satisfactory for certain systems that are not deterministic. (In particular, it does not contain information on the numbers of transitions with same source, same target and same properties; the behaviour of the transition system depends on such numbers.) We define another representing structure, the domain of

which is the set of transitions and not that of states. If

$$\mathscr{R} = \langle S, T, \textbf{init}, \textbf{src}, \textbf{tgt}, P_1, \ldots, P_n, \mathscr{Q}_1, \ldots, \mathscr{Q}_m \rangle$$

we let

$$|\mathscr{R}|_2 := \langle T, \textbf{Init-tr}, \textbf{Follow}, P'_1, \ldots, P'_n, \mathscr{Q}_1, \ldots, \mathscr{Q}_m \rangle$$

where **Init-tr** is the set of *initial transitions*, i.e., of the transitions, the source of which is the initial state **init** and where **Follow** is the binary relation on T defined as the set of pairs (x, y) such that $\textbf{tgt}(x) = \textbf{src}(y)$, i.e., such that y can follow immediately x in a path. We let P'_i be the set of transitions the target of which satisfies P_i. For technical convenience, we shall assume that the initial state of $\mathscr{R}$ satisfies none of the properties P_i: hence, the structure $|\mathscr{R}|_2$ defines $\mathscr{R}$ completely, up to isomorphism. Note also that for every $x, y \in T$:

$$\textbf{Follow}(x) \cap \textbf{Follow}(y) \neq \emptyset \Rightarrow \textbf{Follow}(x) = \textbf{Follow}(y)$$

where we denote by $\textbf{Follow}(x)$ the set of transitions y such that (x, y) belongs to **Follow**.

In the following conjecture, MS_2 denotes MS logic relative to the representation of a transition system $\mathscr{R}$ by the structure $|\mathscr{R}|_2$.

Conjecture 4.2. For every $n, m \in \mathbb{N}$, the transduction **bhv** is (MS_2, MS_1)-compatible on the class $\mathscr{T}(n, m)$.[1]

This conjecture cannot be proved by a straightforward extension of the proof of Theorem 4.1 for the following reason. The proof of Theorem 3.1, upon which Theorem 4.1 is based, makes an essential use of the results by Niwinski [18, 19] saying that the μ-calculus is equivalent to MS-logic on D-trees, a result which is no longer true on arbitrary trees (and in particular on the behaviours of general transition systems). For an example, two finite trees that are not isomorphic but are related by a bisimulation [1] cannot be distinguished by any formula of the μ-calculus, whereas they can be by an MS-formula. We shall prove this conjecture for certain *ordered* transition systems that we now define.

4.1. Ordered transition systems

An ordered transition system is a tuple of the form

$$\mathscr{R} = \langle S, T, \textbf{init}, \textbf{src}, \textbf{tgt}, \leqslant, P_1, \ldots, P_n, \mathscr{Q}_1, \ldots, \mathscr{Q}_m \rangle$$

where $\langle S, T, \textbf{init}, \textbf{src}, \textbf{tgt}, P_1, \ldots, P_n, \mathscr{Q}_1, \ldots, \mathscr{Q}_m \rangle$ is a transition system and $\leqslant$ is a partial order on T such that each set $\textbf{out}(x) := \{ y \in T \mid \textbf{src}(y) = x \}$ where x is a state, is

[1] Note added at revision: This conjecture has been established by B. Courcelle and I. Walukiewicz [11].

linearly ordered by $\leqslant$ and is either finite or has the order type of ω. We denote by $\mathcal{O}(n,m)$ the class of ordered transition systems of type (n,m).

The behaviour of an ordered transition system is a tree, as is that of any transition system, but in addition, this tree is ordered in the following way:

we let $p \leqslant p'$ (where p and p' are nodes of the tree, hence are finite paths in $\mathcal{R}$) if and only if either $p = p'$ or $p = p'' \cdot t$, $p' = p'' \cdot t'$, where the last transitions of these paths, t and t', are such that $t \leqslant t'$ (for the given ordering $\leqslant$ on transitions).

In the structure $|\mathcal{R}|_2$ representing an ordered transition system $\mathcal{R}$ we shall include the order $\leqslant$ on the domain of transitions.

If we order the transition system $\mathcal{R}$ of Fig. 3 by taking for $\leqslant$ the usual order on integers (transitions are denoted by integers in this example), then its behaviour is the tree of Fig. 4 where the left-right branching on the drawing corresponds to the above-defined order of $\mathbf{bhv}(\mathcal{R})$.

Theorem 4.3. *For every $n, m \in \mathbb{N}$, the transduction* **bhv** *is (MS_2, MS_1)-compatible on $\mathcal{O}(n,m)$.*

Proof. Our goal is to use Theorem 4.1, by means of an MS-definable transduction θ that transforms an ordered transition system in $\mathcal{O}(n,m)$ into one belonging to $\mathcal{D}(n+m,2)$. We shall construct θ such that, for every such transition system $\mathcal{R}$ in $\mathcal{O}(n,m)$:

$$\mathbf{bhv}(\mathcal{R}) = \mathbf{flat}(\mathbf{bhv}(\theta(|\mathcal{R}|_2))),$$

where $\theta(|\mathcal{R}|_2)$ is a transition system in $\mathcal{D}(n+m,2)$. The MS-definable transduction **flat** is defined in Section 1: it transforms a binary tree into an ordered one, by redefining certain edges. The result follows then immediately from Proposition 1.1, Lemma 1.2 and Theorem 4.1.

Construction of $\theta(|\mathcal{R}|_2)$. With the structure

$$|\mathcal{R}|_2 = \langle T, \mathbf{Init\text{-}tr}, \mathbf{Follow}, \leqslant, P'_1, \ldots, P'_n, \mathcal{Q}_1, \ldots, \mathcal{Q}_m \rangle$$

representing an ordered transition system $\mathcal{R}$, we associate the structure $\langle T \cup \{*\}, \{*\}, R_{\mathbf{first}}, R_{\mathbf{next}}, P'_1, \ldots, P'_n, \mathcal{Q}_1, \ldots, \mathcal{Q}_m \rangle$ where $*$ is a new object and

$$R_{\mathbf{first}} = \{(x,y) \mid y \text{ is the } \leqslant\text{-smallest element in the set } \mathbf{Follow}(x)\} \cup \{(*,y) \mid y \text{ is the } \leqslant\text{-smallest element in } \mathbf{Init\text{-}tr}\},$$

$$R_{\mathbf{next}} = \{(x,y) \mid x \text{ and } y \text{ belong to } \mathbf{Follow}(z) \text{ for some } z, \text{ and } y \text{ is the } \leqslant\text{-successor of } x \text{ in this set}\} \cup \{(x,y) \mid x \text{ and } y \text{ belong to } \mathbf{Init\text{-}tr} \text{ and } y \text{ is the } \leqslant\text{-successor of } x \text{ in this set}\}.$$

Remark that $R_{\mathbf{first}}$ and $R_{\mathbf{next}}$ are disjoint, and that $R_{\mathbf{first}}$ can contain a pair of the form (x,x) whereas $R_{\mathbf{next}}$ cannot. This structure is of the form $|\mathcal{T}|_1$ for some transition system $\mathcal{T}$ in $\mathcal{D}(n+m,2)$. The transitions of $\mathcal{R}$ are states of $\mathcal{T}$ and the initial state of

$\mathcal{T}$ is $*$. Properties $P_1, \ldots, P_n$ of the states and properties $\mathcal{Q}_1, \ldots, \mathcal{Q}_m$ of the transitions of $\mathcal{R}$ become properties $P'_1, \ldots, P'_n, \mathcal{Q}_1, \ldots, \mathcal{Q}_m$ of the states of $\mathcal{T}$. For ease of understanding the construction, we use **first** and **next** instead of 1 and 2. We shall identify $\mathcal{T}$ and $|\mathcal{T}|_1$ and denote them both by $\theta(|\mathcal{R}|_2)$ where θ is a transduction of relational structures such that $\theta(|\mathcal{R}|_2$ is defined for every $\mathcal{R}$ in $\mathcal{O}(n,m)$. Fig. 7 shows $\theta(|\mathcal{R}|_2)$ where $\mathcal{R}$ is the transition system of Fig. 3, and Fig. 8 shows its behaviour. Note that property P of states and property Q of transitions of $\mathcal{R}$ become properties P' and Q of states of $\mathcal{T}$.

Claim. *The transduction θ is MS-definable. We have* $\mathbf{flat}(\mathbf{bhv}(\theta(|\mathcal{R}|_2))) = \mathbf{bhv}(\mathcal{R})$.

Proof. That θ is MS-definable is clear from the definition. We let $\mathcal{R}$ be a transition system in $\mathcal{O}(n,m)$ and $\mathcal{T}$ be the system constructed by θ. We now define an isomorphism h of $\mathbf{flat}(\mathbf{bhv}(\mathcal{T}))$ onto $\mathbf{bhv}(\mathcal{R})$.

A node of $\mathbf{flat}(\mathbf{bhv}(\mathcal{T}))$ is a finite path π in $\mathcal{T}$. Since $R_{\mathbf{first}}$ and $R_{\mathbf{next}}$ are disjoint, this path can be defined as a sequence of states. We let $\pi = (* = t_0, t_1, t_2, \ldots, t_k)$. Note that $t_1, t_2, \ldots, t_k$ are transitions of $\mathcal{R}$. We let J be the set of indices j such that $\mathbf{Follow}(t_j, t_{j+1})$ (in $\mathcal{R}$), augmented with the index k if $k \geqslant 1$. These transitions are those such that the edge from t_j to t_{j+1} is of type **first**. We let $j_1, j_2, \ldots, j_p$ be the enumeration of J in increasing order. Then the sequence $t_{j_1}, t_{j_2}, \ldots, t_{j_p}$ is a path π' in $\mathcal{R}$. We let $h(\pi) = \pi'$. If $k = 0$, then J is empty and π' is the empty path in $\mathcal{R}$. The mapping h is a bijection and also an isomorphism of trees. Also π satisfies a property $P'_i{}^*$ or $\mathcal{Q}_j^*$, which means that its end state satisfies P'_i or that it last transition satisfies $\mathcal{Q}_j^*$, if and

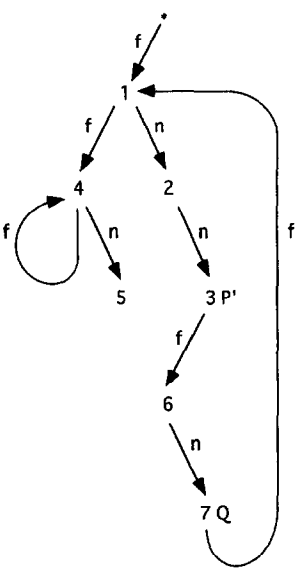

Fig 7.

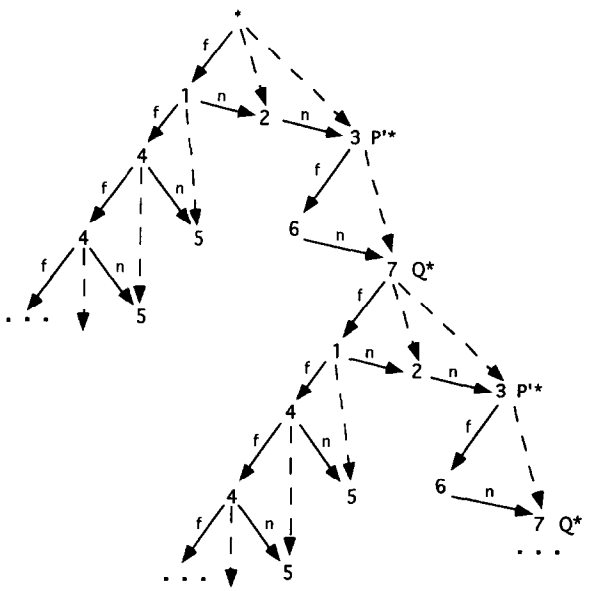

Fig 8.

only if the target of t_k satisfies P_i or t_k itself satisfies $\mathcal{Q}_j$, if and only if π' satisfies $P_i'^*$ or $\mathcal{Q}_j^*$ respectively. Hence h also preserve the node labels encoding their properties. □

Fig. 8 shows the top part of $\mathbf{bhv}(\theta(|\mathcal{R}|_2))$ where $\mathcal{R}$ is as in Fig. 3. For sake of readability, we use the edge labels **n** and **f** instead of **next** and **first**, and at each node, we only indicate the number of the last transition, and not the corresponding path. The broken edges show the pairs (x, y) such that $x \prec_1 y$ and (x, y) does not belong to $\mathbf{Suc}_{\mathbf{first}}$. These edges together with those labelled by **first** yield the tree of Fig. 4.

Let $d \in \mathbb{N}$. A transition system is *d-bounded* if every state is the source of at most d transitions.

Corollary 4.4. *For every d, n, m* $\in \mathbb{N}$, *the mapping* **bhv** *is* (MS_2, MS_1)*-compatible on d-bounded transition systems of type* (n, m).

Proof. It is enough to construct an MS-definable transduction α that transforms a d-bounded transition system $\mathcal{R}$ (given by $|\mathcal{R}|_2$) into an ordered one with same behaviour (up to the order on successors of nodes that has no meaning in $\mathbf{bhv}(\mathcal{R})$). We shall get $\mathbf{bhv}(\mathcal{R}) = \mathbf{flat}(\mathbf{bhv}(\theta(\alpha(|\mathcal{R}|_2))))$ and the result will follow from Proposition 1.1 and Theorem 4.3.

We now sketch the definition of α, intended to produce a relational structure of the form $|\mathcal{T}|_2$ for some transition system $\mathcal{T}$. The corresponding definition scheme will use d parameters $X_1, \ldots, X_d$ subject to satisfy the following conditions:

(1) $X_1, \ldots, X_d$ form a partition of T,

(2) for every $x, y, z \in T$ and $i \in \{1, \ldots, d\}$
if $x \in \mathbf{Follow}(z)$, $y \in \mathbf{Follow}(z)$, $x \in X_i$ and $y \in X_i$, then $x = y$.
(We recall that $\mathbf{Follow}(x)$ denotes the set of transitions y the source of which is the target of x.) Assuming these conditions we can now define

(3) $x \leqslant y$ if $x = y$ or if for some $z \in T$, some i and j with $1 \leqslant i < j \leqslant d$ we have $x \in \mathbf{Follow}(z)$, $y \in \mathbf{Follow}(z)$, $x \in X_i$ and $y \in X_j$.

The existence of $X_1, \ldots, X_d$ satisfying Conditions (1) and (2) follows from the hypothesis that $\mathscr{R}$ is d-bounded (the sets $\mathbf{Follow}(x)$ have at most d elements) and the facts that two sets $\mathbf{Follow}(x)$ and $\mathbf{Follow}(y)$ are either equal or disjoint). These two conditions are expressible in MS-logic. From every tuple $(X_1, \ldots, X_d)$ satisfying them, one obtains by (3) the definition in MS-logic of the desired partial order. From these definitions the formal construction of a definition scheme for α is straightforward. □

4.2. *Unordered transition systems*

Let $\mathscr{F}(n,m)$ be the class of transition systems of type (n, m) such that each set $\mathbf{Follow}(x)$ is finite. Let φ be an MS-formula (without $\leqslant$) expressing a property of behaviours of transition systems from $\mathscr{F}(n, m)$. An immediate corollary of Theorem 4.3 is that the condition $\mathbf{bhv}(\mathscr{R}) \models \varphi$ is expressible by an MS-formula ψ on the relational structure $|(\mathscr{R}, \leqslant)|_2$ where $(\mathscr{R}, \leqslant)$ is $\mathscr{R}$ augmented with a linear ordering $\leqslant$ of its set of transitions. However, for any two such orders $\leqslant$ and $\leqslant'$

$$|(\mathscr{R}, \leqslant)|_2 \models \psi \text{ if and only if } |(\mathscr{R}, \leqslant')|_2 \models \psi \qquad (*)$$

because these conditions are both equivalent to the condition $\mathbf{bhv}(\mathscr{R}) \models \varphi$ which is independent of any ordering of the transitions of $\mathscr{R}$. It follows that the property $\mathbf{bhv}(\mathscr{R}) \models \varphi$ of $\mathscr{R}$ is expressible in $|\mathscr{R}|_2$ by an MS-formula, with the help of an *auxiliary but arbitrary* linear ordering of the domain. (*Arbitrary* means that *any* linear order of the domain works, see $(*)$ above.) With the terminology of Courcelle [9], this means that the corresponding property of $|\mathscr{R}|_2$ is an MS($\leqslant$)-property, hence an $\mathrm{MS}_2(\leqslant)$-property of $\mathscr{R}$.

Corollary 4.5. *For every n, $m \in \mathbb{N}$, the transduction* **bhv** *is $(MS_2(\leqslant), MS_1)$-compatible on the class $\mathscr{F}(n,m)$.*

The property that a set X has an even cardinality is MS($\leqslant$) but not MS [7,9]. In the case of the property $\mathbf{bhv}(\mathscr{R}) \models \varphi$, for $\mathscr{R}$ in $\mathscr{F}(n,m)$ we do not know whether the use of an auxiliary but arbitrary linear ordering is really necessary, and we do not know either whether this property can be expressed on $|\mathscr{R}|_2$ in *counting monadic second-order logic*: this language is an extension of MS-logic considered in [7], where special predicates can test whether a set has a cardinality that is a multiple of a fixed integer. (One can equivalently define it as the extension of MS-logic with "counting modulo" first-order existential quantifications: $\exists_{\mathbf{mod}\, q} x \cdot \psi(x)$ says that the set of objects x satisfying $\psi(x)$ has a cardinality that is a multiple of q.)

4.3. The logic CTL*

Temporal logics like CTL* are often presented as logics expressing properties of transition systems, whereas they actually express properties of their behaviours. We discuss informally CTL* referring the reader to Arnold [1], Dam [12] or Emerson [13] for formal definitions. The formulas of CTL* express properties of states and paths of transition systems. *Paths* may be infinite and they start from any state, whereas above we only considered finite paths starting from the initial state. There are two types of formulas, the *state formulas*, denoted by σ, σ', σ'', ... and the *path formulas* denoted by π, π', π'', ... The main constructions (in addition to the usual Boolean combinations) are as follows. For every state formula σ and for every path formulas π and π':

- $\mathbf{E}\pi$ is a *state* formula expressing that there exists a maximal path (maximal for the prefix ordering) satisfying π and starting at the considered state,
- $\mathbf{N}\pi$ is a *path* formula expressing that the considered path truncated from its first transition satisfies π,
- $\pi\mathbf{U}\pi'$ is a *path* formula expressing that the considered path satisfies π' or is of the form $t_1 \ldots t_n p$, where $t_1, \ldots, t_n$ are transitions, p satisfies π', and for every $i = 1, \ldots, n$, the path $t_i \ldots t_n p'$ satisfies π,
- $\mathbf{D}\sigma$ is a *path* formula expressing that the origin of the considered path satisfies the state formula σ.

Let us associate with a transition system $\mathscr{R} = \langle S, T, \mathbf{init}, \mathbf{src}, \mathbf{tgt}, P_1, \ldots, P_n \rangle$ of type $(n, 0)$ the relational structure

$$\mathbf{bhv}^{\infty}(\mathscr{R}) = \langle S \cup P, \prec, \mathbf{trunc}, \mathbf{fin}, P_1^*, \ldots, P_k^*, P_1^{\#}, \ldots, P_k^{\#} \rangle$$

where P is the set of *paths* in $\mathscr{R}$, i.e., of finite or infinite nonempty sequences of transitions $(t_1, t_2, \ldots, t_n, \ldots)$ such that $\mathbf{tgt}(t_i) = \mathbf{src}(t_{i+1})$ for every $i \geqslant 1$. The *origin* of a path is the source of its first transition. We let $\prec$ be the *prefix order* on paths, extended by letting $s \prec p$, whenever the state s is the origin of the path p. The unary relation **fin** characterizes the finite paths, and **trunc** is the set of pairs (p, q), such that q is obtained from the path p by deleting the first transition. The component q of such a pair is a state if p is reduced to a single transition with target q. For every property P_i of states, we let P_i^* be the property of a finite path saying that the last state of this path satisfies P_i, and we let $P_i^{\#}$ be the property of a path saying that its origin satisfies P_i. We let also $P_i^{\#}$ hold for the states that satisfy P_i.

It is then straightforward to translate an arbitrary formula ψ of CTL* into a *first-order* formula θ such that, for every transition system $\mathscr{R}$, θ holds true in $\mathbf{bhv}^{\infty}(\mathscr{R})$ if and only if ψ holds true in $\mathscr{R}$. Since infinite paths can be handled as linearly ordered sets of finite paths, θ can itself be translated into an MS formula ξ to be interpreted in $\mathbf{bhv}^{+}(\mathscr{R})$, where $\mathbf{bhv}^{+}(\mathscr{R})$ is like $\mathbf{bhv}(\mathscr{R})$ except that its domain consists of all finite paths, and not only of those starting from the initial state. If we assume that $\mathscr{R}$ is such that every state of $\mathscr{R}$ is reachable by a path from the initial state, then one can

transform θ into a formula μ such that μ holds true in $\mathbf{bhv}(\mathscr{R})$ if and only if ψ holds true in $\mathscr{R}$.

In words this means that every CTL*-property of a transition system $\mathscr{R}$ (every state of which is accessible) is an MS-property of its behaviour. For certain systems $\mathscr{R}$ (see Theorems 4.1–4.3), we obtain that it is an MS-property of the systems $\mathscr{R}$ themselves (appropriately represented by relational structures). The validity of Conjecture 4.2 would give an extension of this result to arbitrary transition systems. This extension actually holds by the result of Dam [12] saying that every CTL*-property of a transition system $\mathscr{R}$ is expressible in the μ-calculus (on $\mathscr{R}$), since the μ-calculus is a fragment of MS-logic.

5. Algebraic trees

Algebraic trees are infinite trees associated with certain recursive program schemes. It is proved in [3] (and also in [14]) that an infinite tree is algebraic if and only if its set of finite branches (where each branch is represented by a word in a precise way) is a deterministic context-free language. Here, we shall use this characterization as a definition.

Let A be a ranked alphabet with all symbols of rank at most k. A tree t in $\mathbf{T}^{\infty}(A)$ is completely defined by the partial mapping: $[k]^* \to A$, also denoted t, that associates with every node (represented by its access path, see Section 1) the symbol in A that labels it. The *set of branches* of t is the language $\mathbf{Brch}(t) := \{wt(w) \mid w \in [k]^*\} \subseteq [k]^*A$. It is clear that t is characterized in a unique way by the language $\mathbf{Brch}(t)$. We shall say that the tree t is *algebraic* if the language $\mathbf{Brch}(t)$ is a deterministic context-free language. (See [16] for background on languages).

The tree t such that $\mathbf{Brch}(t) = La \cup (\{1,2\}^* - L)b$ where $L = \{2^n1^n \mid n \geqslant 0\}$ is shown above in Fig. 6. (The symbols a and b have rank 2; only the labels a are shown in Fig. 6.)

The following result is from [6] restated in the terminology of the present paper.

Proposition 5.1. *Let A be a ranked alphabet of cardinality at most n and k be the maximal rank of its elements. If a tree in $\mathbf{T}^{\infty}(A)$ is algebraic, then it is the behaviour of a transition system in $\mathscr{D}(n,k)$ that, considered as a graph, is equational.*

The transition system of Fig. 5 is an equational graph that we shall denote by H. Its behaviour, i.e., its unfolding from the initial state, is the algebraic tree shown in Fig. 6. The graph H is equational because it is defined by the two equations $H = f(G)$ and $G = g(G)$ illustrated below in Fig. 9. The top part of the figure shows the equation $H = f(G)$. The mapping f embeds a graph G with 3 distinguished vertices called the *sources* in a "context" graph consisting of 7 edges. The bottom part of the figure shows the equation $G = g(G)$. The mapping g embeds G as above in a "context" graph consisting of 4 edges. The Italic numbers 1, 2, 3 mark the sources and should not be

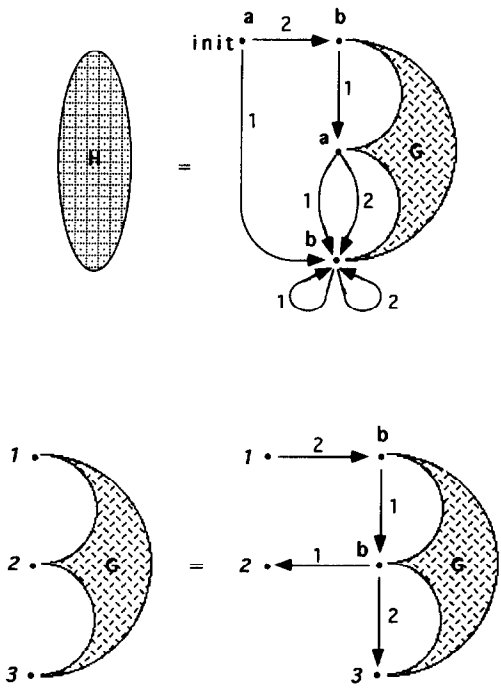

Fig 9.

confused with the Roman numbers 1, 2 which are edge labels. We refer the reader to [6, 8] for more details about equational graphs (i.e., graphs defined by systems of equations).

Theorem 5.2. *The monadic theory of an algebraic tree is decidable.*

Proof. Let t be an algebraic tree, given by Proposition 5.1 as the behaviour of a k-ary deterministic transition system H that is equational. This transition system is infinite but can be given by a finite system Σ of (finite) graph equations. Let ψ be an MS-formula the validity in t of which one wants to decide. It follows from Theorem 4.1 that this validity is equivalent to the validity in H of an MS_2-formula θ that one can construct from ψ. This validity can be decided because the MS_2 theory of an equational graph, given by a relevant system of equations, in decidable [6]. □

We now discuss some links with the theory of context-free languages. Let us fix an integer k. For every language $L \subseteq [k]^*$, we let **Tree**(L) be the tree in $\mathbf{T}^\infty(\{a, b\})$ such that **Brch(Tree**$(L)) = La \cup ([k]^* - L)b$. We get immediately the following corollary since the constructions of [3] defining a system of equations in trees from a deterministic language, and of [6] defining a system of graph equations from a system of equations in trees are effective.

Corollary 5.3. *For every deterministic context-free language $L \subseteq [k]^*$, the monadic theory of the tree* **Tree**(L) *is decidable.*

Remark 5.4. For every finite automation $\mathcal{A}$ over $[k]$, the monadic theory of the tree **Tree**$(L(\mathcal{A}))$ is decidable because **Tree**$(L(\mathcal{A}))$ is a regular tree. However the transduction from $\mathcal{A}$ to **Tree**$(L(\mathcal{A}))$ is not MS-compatible. If it would be, then the class of finite automata $\mathcal{A}$ over $[1]$ such that **Tree**$(L(\mathcal{A}))$ has no occurrence of b would be MS-definable. This condition is equivalent to the property that $L(\mathcal{A}) = 1^*$ which is not MS-definable by Proposition 2.4.

However, the mapping associating the tree **Tree**(L($\mathcal{A}$)) with a finite or infinite deterministic automation $\mathcal{A}$ over $[k]$ is MS-compatible. This follows from Theorem 4.1.

One may ask: *For which languages L is the monadic theory of* **Tree**(L) *decidable*? Here are some answers.

Proposition 5.5. (1) *The monadic theory of* **Tree**(L) *where L is the noncontext-free language $\{2^n1^n2^n \mid n > 0\}$ is decidable.*

(2) *There exists a context-free language M for which the monadic theory of* **Tree**(M) *is undecidable.*

Proof. (1) We let $L = \{2^n1^n2^n \mid n > 0\}$. This language is defined by the deterministic infinite automaton shown on Fig. 10, with initial state marked **init** and final state marked a. Let $\mathcal{R}$ be the complete deterministic transition system obtained from that of Fig. 10 by the addition of a "sink" state denoted by $\perp$ receiving all missing transitions. Let us mark by b each state of $\mathcal{R}$, except the one already marked by a. The tree **Tree**(L) is thus the behaviour of $\mathcal{R}$. But the structure $|\mathcal{R}|_1$ is the image under an MS-definable transduction α of the algebraic tree t of Fig. 6.

Let us describe α. It takes as input any tree s in $\mathbb{T}(\{a, b\})$ and produces a complete deterministic transition system $\mathcal{T}$ of type $(2, 2)$ with set of states S and set of transitions T defined as follows, where we denote by $\mathbf{n}(w)$ the node of s with access path w, for $w \in \{1, 2\}^*$:

S is the set $\{x, \perp\} \cup W \cup W'$ where $x = \mathbf{n}(1)$, $\perp = \mathbf{n}(11)$, W is the set of nodes $\mathbf{n}(2^n)$ for all $n \geqslant 0$, and W' is the set of nodes $\mathbf{n}(2^n1^m)$ for all n and $m \geqslant 1$ such that $s(2^n1^p) = b$ for every p, $0 \leqslant p < m$; in $\mathcal{T}$ the state x has label a and all other states have label b;

T consists of the following transitions:

(a) the transitions of s with source and target in $W \cup W'$ with same types as in s,

(b) a transition of type 2 from $\mathbf{n}(21)$ to x,

(c) transitions of type 2 from u to v, where u, v belong to W' and have label a in s, and there exist u' and v' in W with paths from u' to u and from v' to v and such that there is a transition (necessarily of type 2) from v' to u',

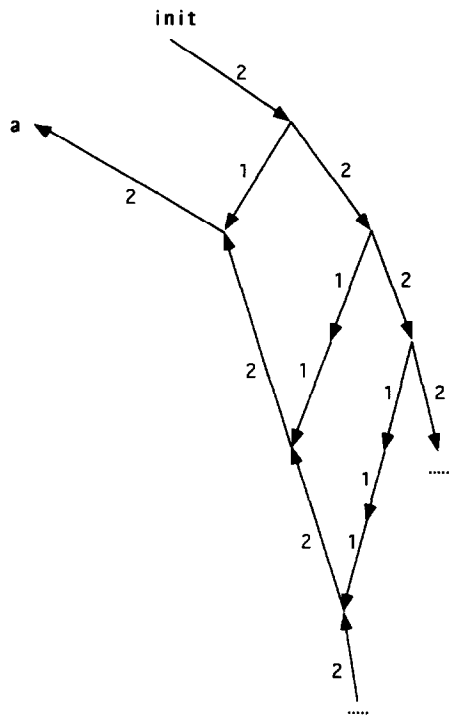

Fig 10.

(d) a set of transitions with target $\perp$ such that the resulting transition system is complete and deterministic, of type (2, 2).

It is clear that $\mathscr{T} = \mathscr{R}$ if $s = t$.

By Theorem 3.1, the MS-theory of **Tree**(L) reduces to that of $|\mathscr{R}|_1$ and by Proposition 1.1, the MS-theory of $|\mathscr{R}|_1$ reduces to that of t. Since the MS-theory of t is decidable (by Theorem 5.2), so is that of **Tree**(L). (Thomas uses in [22] the graph of Fig. 10 as an example of nonequational graph, the monadic theory of which is decidable; the decidability of its monadic theory is proved in [22] in a completely different way.)

(2) For proving the second assertion, we observe that there exists a context-free language $M \subseteq \{1, 2\}^*$, such that one cannot decide, for given u in $\{1, 2\}^*$, whether the language $M/u := \{v/uv \in M\}$ is equal to $\{1, 2\}^*$. (This is actually an easy variation on the proof that one cannot decide whether a context-free language is equal to $\{1, 2\}^*$; one uses the modified Post correspondence problem associated with a universal Turing machine; see [16]; we omit details.) We now consider the tree **Tree**(M). For every word u in $\{1, 2\}^*$, one can construct a closed MS formula ψ such that **Tree**$(M) \models \psi$ if and only if $M/u = \{1, 2\}^*$. It follows then from the choice of M that the monadic theory of **Tree**(M) is undecidable. □

6. Conclusion

Languages like *CTL** [1, 12, 13] are usually presented as describing properties of transition systems, whereas they actually express properties of their behaviours, the behaviour of a transition system being defined as its set of finite and infinite paths. In this paper, we have made explicit the distinction between a machine and its behaviour, especially in the perspective of the logical expression of properties of behaviours of machines. We have exhibited several classes of machines for which the behaviour mapping is *monadic second-order compatible* which means that monadic second-order properties of behaviours of machines can be translated "back" into monadic second-order properties of the machines themselves. These cases are summarized in Table 1.

We have also answered a question discussed by Thomas [22], by establishing that the monadic theory of an algebraic tree is decidable.

Table 1

Machine	Behaviour	Proof
Finite or infinite automaton	Language of finite words	Theorem 2.3
Finite or infinite tree automaton	Set of finite trees	Theorem 2.7
Finite or infinite tree automaton	Set of infinite trees	Theorem 3.1
Finite Rabin automaton	Set of infinite trees	Theorem 3.2
Deterministic transition system	Tree of finite paths	Theorem 4.1
Ordered transition system	Tree of finite paths	Theorem 4.3
d-bounded transition system	Tree of finite paths	Corollary 4.4

Acknowledgements

Many thanks to A. Arnold, J. Makowsky and D. Niwinski for helpful comments on first versions of this paper.

Appendix

This appendix contains the proof of Theorem 3.1 that we now restate. We let A be a finite ranked alphabet consisting of symbols all of rank 2.

Theorem 3.1. *Let K be a Rabin recognizable subset of $\mathbb{T}(A)$. The property of a finite or infinite tree automaton $\mathscr{A}$ over A that $L(\mathscr{A}) \cap K \neq \emptyset$ is MS-definable. An MS formula expressing this property can be constructed from a finite Rabin automaton defining K.*

We shall use a fragment of MS-logic called the *μ-calculus*, and establish that the set of states q of $\mathscr{A}$ such that $L(\mathscr{A}, q) \cap K \neq \emptyset$ is the value of a *μ-term* in a certain

algebraic structure $\mathscr{P}(\mathscr{A})$ built on subsets of the set of states of $\mathscr{A}$. The following definitions and lemmas are essentially from Niwinski [18, 19].

We let $\mathscr{A} = \langle \mathscr{Q}, (R_a)_{a\in A}, F\rangle$ be a tree-automaton. We first define $\mathscr{P}(\mathscr{A})$. We let $\mathscr{P}(\mathscr{Q})$ denote the powerset of $\mathscr{Q}$. We let

$$\mathscr{P}(\mathscr{A}) := \langle \mathscr{P}(\mathscr{Q}), \subseteq, \cup, (\tilde{a})_{a\in A}, \bot, \top\rangle$$

where $\subseteq$ is set inclusion, $\bot := \emptyset$ (the least element of $\mathscr{P}(\mathscr{Q})$) and $\top := \mathscr{Q}$ (the largest one). The binary operation $\cup$ is set union and for all a in A:

$$\tilde{a}(\mathscr{Q}_1, \mathscr{Q}_2) := \{q\in\mathscr{Q} \mid (q_1, q_2, q)\in R_a \text{ for some } q_1\in\mathscr{Q}_1,\ q_2\in\mathscr{Q}_2\}.$$

Remark that $\langle \mathscr{P}(\mathscr{Q}), (\tilde{a})_{a\in A}\rangle$ is nothing but the bottom-up deterministic tree automaton (without accepting states) associated with $\mathscr{A}$ [15]. On the other hand, the ordered set $\langle \mathscr{P}(\mathscr{Q}), \subseteq, \cup, \bot, \top\rangle$ is a complete lattice with $\cup$ as binary join operation. The binary functions $\tilde{a}$ where $a\in A$ are all monotone and continous.

We now recall the definition of *μ-terms*. (The μ-terms we shall use are not of the most general form: they are written without intersection and complementation.) A μ-term N with free variables in $\{x_1, \ldots, x_k\}$ is intended to define a monotone mapping $N_{\mathscr{P}(\mathscr{A}),k}: \mathscr{P}(\mathscr{Q})^k \to \mathscr{P}(\mathscr{Q})$. We define simultaneously the syntax and semantics of μ-terms.

(1) Every variable x_i is a μ-term, x_i is free in this term and $(x_i)_{\mathscr{P}(\mathscr{A}),k} := \pi_i$, the ith projection: $\mathscr{P}(\mathscr{Q})^k \to \mathscr{P}(\mathscr{Q})$.

(2) The constants $\bot$ and $\top$ are μ-terms without free variables and they define respectively the constant functions equal to $\emptyset$ and to $\mathscr{Q}$.

(3) If N and N' are μ-terms then $N\cup N'$ and $a(N, N')$ are μ-terms: their free variables are those of N and N' and their values are the functions such that

$$(N\cup N')_{\mathscr{P}(\mathscr{A}),k}(\vec{S}) = N_{\mathscr{P}(\mathscr{A}),k}(\vec{S})\cup N'_{\mathscr{P}(\mathscr{A}),k}(\vec{S}),$$

$$a(N, N')_{\mathscr{P}(\mathscr{A}),k}(\vec{S}) = \tilde{a}(N_{\mathscr{P}(\mathscr{A}),k}(\vec{S}),\ N'_{\mathscr{P}(\mathscr{A}),k}(\vec{S}))$$

for every $\vec{S}\in\mathscr{P}(\mathscr{Q})^k$ and $a\in A$.

(4) If N is a μ-term and y is a variable, then $\mu y.N$ and $\nu y.N$ are μ-terms, their free variables are those of N except y and for every $\vec{S}\in\mathscr{P}(\mathscr{Q})^k$:

$(\mu y.N)_{\mathscr{P}(\mathscr{A}),k}(\vec{S})$ is the least solution of the equation

$$X = N_{\mathscr{P}(\mathscr{A}),k+1}(\vec{S}, X) \tag{A.1}$$

and

$(\nu y.N)_{\mathscr{P}(\mathscr{A}),k}(\vec{S})$ is the greatest one,

where $(\vec{S}, X)$ denotes $(S_1, \ldots S_k, X)$ if $\vec{S} = (S_1, \ldots, S_k)$. When writing $N_{\mathscr{P}(\mathscr{A}),k+1}$, we assume that y is numbered as x_{k+1} which is compatible with the assumption that all free variables of $\mu y.N$ and $\nu y.N$ are in $\{x_1, \ldots, x_k\}$. It may happen that y has no free occurrence in N. That Eq. (A.1) has indeed a least and a greatest solution is ensured by Knaster–Tarski's lemma. See Arnold and Niwinski for details [2].

μ-terms can also denote sets of finite and infinite trees with variables. The symbols in A are all of rank 2 and the variables are of rank 0. We shall denote by X_k the set of variables $\{x_1, \ldots, x_k\}$. Every μ-term N with free variables in X_k defines a subset of $\mathbf{T}^\infty(A \cup X_k)$ that we shall denote by $N_{\mathbf{Tree},k}$. We first review some definitions.

For every a in A, we have a binary mapping on $\mathbf{T}^\infty(A \cup X_k)$, also denoted by a and defined as follows. For t, t' in $\mathbf{T}^\infty(A \cup X_k)$, the tree $a(t, t')$ is obtained by taking the union of two disjoint copies of t and t' (considered as graphs) augmented with a new vertex, say r, labelled by a that is the root of $a(t, t')$ and edges from r to **root**(t) and **root**(t'); **root**(t) is the first successor of r and **root**(t') is the second. Every symbol of X_k can be considered as a tree reduced to a single node, labelled by this symbol, hence belongs to $\mathbf{T}^\infty(A \cup X_k)$.

We shall also use substitutions of sets of trees for variables. If t belongs to $\mathbf{T}^\infty(A \cup X_k)$ and $L_1, \ldots, L_k$ are subsets of $\mathbf{T}^\infty(A \cup Y)$ for some set of variables Y, then $t[L_1/x_1, \ldots, L_k/x_k]$ denotes the set of trees obtained by substituting in t an element of L_i for each occurrence of variables x_i. Different trees in L_i may be substituted for different occurrences of x_i. If L is a subset of $\mathbf{T}^\infty(A \cup X_k)$, then $L[L_1/x_1, \ldots, L_k/x_k]$ denotes the union of the sets $t[L_1/x_1, \ldots, L_k/x_k]$ for all trees t in L. See [4] for more details on infinite trees and operations on them.

We can now define $M_{\mathbf{Tree},k}$ by induction on the structure of a μ-term M assumed to have all its free variables in X_k

(1) If M is the variable x_i, then $M_{\mathbf{Tree},k}$ is the set $\{x_i\}$.

(2) The values of the constants $\bot$ and $\top$ are respectively the empty set and the set $\mathbf{T}^\infty(A \cup X_k)$.

(3) $(N \cup N')_{\mathbf{Tree},k} = N_{\mathbf{Tree},k} \cup N'_{\mathbf{Tree},k}$

and

$$a(N,N')_{\mathbf{Tree},k} = a(N_{\mathbf{Tree},k}, N'_{\mathbf{Tree},k})$$

$$= \{a(t, t')/t \in N_{\mathbf{Tree},k}, t' \in N'_{\mathbf{Tree},k}\}$$

for every $a \in A$.

(4) $(\mu y.N)_{\mathbf{Tree},k}$ is the least solution of the equation

$$X = N_{\mathbf{Tree},k+1}[X/y] \tag{A.2}$$

and $(\nu y.N)_{\mathbf{Tree},k}$ is the greatest one, where X ranges over subsets of $\mathbf{T}^\infty(A \cup X_k)$. If y has no free occurrence in N, then $(\mu y.N)_{\mathbf{Tree},k} = (\nu y.N)_{\mathbf{Tree},k} = N_{\mathbf{Tree},k}$.

Remark A.1. Let $M \subseteq \mathbf{T}^\infty(\mathrm{A} \cup \mathrm{X}_k \cup \{y\})$. The largest subset L of $\mathbf{T}^\infty(A \cup X_k)$ such that $L = M[L/y]$ can be described concretely as follows. There are two cases

Case 1: $y \in M$. (Note that y belongs to $\mathbf{T}^\infty(A \cup X_k \cup \{y\})$.) It is clear that $\mathbf{T}^\infty(A \cup X_k)$ is a solution of the equation $Y = M[Y/y]$ and is necessarily maximal, hence $L = \mathbf{T}^\infty(A \cup X_k)$.

Case 2: $y \notin M$. A tree belongs to L if and only if it is the limit of an infinite sequence $t_0, t_1, \ldots, t_n, \ldots$ of trees in $\mathbf{T}^\infty(A \cup X_k)$ constructed as follows:

t_0 is a tree in M, $t_1 \in t_0[M/y], \ldots, t_{n+1} \in t_n[M/y], \ldots$

If t_n has no occurrence of y then $t_{n+i} = t_n$ for all $i \geqslant 1$. Since M does not contain y, the occurrences of y in t_n are at depth at least $n+1$. It follows that the limit of $t_0, t_1, \ldots, t_n, \ldots$ is well-defined. (See [2] for more details.)

We now define the semantics of infinite trees relative to a tree automaton $\mathscr{A}$. For every tree t in $\mathbb{T}(A)$, we let the *value* of t in $\mathscr{A}$ be the set of root-states of the runs of $\mathscr{A}$ on t; we denote this set by $\mathbf{val}_{\mathscr{A}}(t)$. In other words, a state q belongs to $\mathbf{val}_{\mathscr{A}}(t)$ if and only if the tree t belongs to $L(\mathscr{A}, q)$.

For the purpose of an inductive proof, we need to extend this definition to trees with variables. Let t be a tree in $\mathbf{T}^\infty(A \cup X_k)$. Let $S_1, \ldots, S_k$ be subsets of $\mathscr{Q}$. An $(S_1, \ldots, S_k)$-*run of* $\mathscr{A}$ *on* t is a mapping from the set of nodes of t to $\mathscr{Q}$ such that, the value of a node labelled by a variable x_i belongs to S_i and the value of a node labelled by a symbol a from A is related with those of its two successors as in the definition of Section 3, for runs of automata on trees without variables. For every tree t in $\mathbf{T}^\infty(A \cup X_k)$, the *value of* t *in* $\mathscr{A}$ is the mapping $\mathbf{val}_{\mathscr{A},k}(t) : \mathscr{P}(\mathscr{Q})^k \to \mathscr{P}(\mathscr{Q})$ such that:

$\mathbf{val}_{\mathscr{A},k}(t)(S_1, \ldots, S_k)$ is the set of root-states of the $(S_1, \ldots, S_k)$-runs of $\mathscr{A}$ on t.

If L is a subset of $\mathbf{T}^\infty(A \cup X_k)$, we let its *value in* $\mathscr{A}$ be the mapping $\mathbf{val}_{\mathscr{A},k}(L) : \mathscr{P}(\mathscr{Q})^k \to \mathscr{P}(\mathscr{Q})$ such that

$\mathbf{val}_{\mathscr{A},k}(L)(S_1, \ldots, S_k)$ is the union of the sets $\mathbf{val}_{\mathscr{A},k}(t)(S_1, \ldots, S_k)$ for t in L.

Lemma A.2. *For every* k *and* $n \in \mathbb{N}$, *if* $L_1, \ldots, L_k \subseteq \mathbf{T}^\infty(A \cup X)_n$ *and* $L \subseteq \mathbf{T}^\infty(A \cup X_k)$ *then*

$$\mathbf{val}_{\mathscr{A},n}(L[L_1/x_1, \ldots, L_k/x_k])(S_1, \ldots, S_n)$$
$$= \mathbf{val}_{\mathscr{A},k}(L)(\mathbf{val}_{\mathscr{A},n}(L_1)(S_1, \ldots, S_n), \ldots, \mathbf{val}_{\mathscr{A},n}(L_k)(S_1, \ldots, S_n))$$

for every $S_1, \ldots, S_n \subseteq \mathscr{Q}$.

Proof. The proof is a straightforward verification from the definitions. □

We shall say that an automaton $\mathscr{A}$ with set of states $\mathscr{Q}$ is *reduced* if for each q in $\mathscr{Q}$ the set $L(\mathscr{A}, q)$ is nonempty, equivalently, if $\mathbf{val}_{\mathscr{A}}(\mathbf{T}^\infty(A)) = \mathscr{Q}$. If it is not, then its restriction to the set of states $\mathbf{val}_{\mathscr{A}}(\mathbf{T}^\infty(A))$ is a reduced automaton $\mathscr{B}$ called the *reduced automaton of* $\mathscr{A}$, such that for every state q of $\mathscr{B}$, $L(\mathscr{B}, q) = L(\mathscr{A}, q)$.

Lemma A.3. *Let* $\mathscr{A}$ *be a reduced automaton. For every integer* k *and every* μ*-term* N *with free variables in* X_k *we have* $N_{\mathscr{P}(\mathscr{A}),k} = \mathbf{val}_{\mathscr{A},k}(N_{\mathbf{Tree},k})$

This lemma means that the value of a μ-term N in $\mathscr{A}$ is the value of the set of trees it defines. Mezei and Wright [17] have established a result of this type for sets of finite

trees defined as least fixed points of systems of polynomial equations. There are many theorems following this scheme. They have numerous applications, in particular to decision problems [5]. This lemma is proved in [18, Theorem 6.2] in the special case where $\mathscr{A}$ is deterministic.

Proof of Lemma A.3. The proof is by induction on the structure of μ-terms. We only consider a few cases.

If $N = \top$, then $N_{\mathbf{Tree},k} = \mathbf{T}^\infty(A \cup X_k)$ and $\mathbf{val}_{\mathscr{A},k}(\mathbf{T}^\infty(A \cup X_k))$ is the constant function equal to $\mathscr{Q}$ since $\mathbf{T}^\infty(A)$ is a subset of $\mathbf{T}^\infty(A \cup X_k)$ and the automaton $\mathscr{A}$ is reduced. On the other hand, $N_{\mathscr{P}(\mathscr{A}),k}$ is defined as the constant function equal to $\mathscr{Q}$. Hence, the result holds.

Now, we let $N = vy.N'$ be a μ-term with free variables in X_k. (We consider y as identical with x_{k+1}). We want to prove that for every $S_1, \ldots, S_k \subseteq \mathscr{Q}$ we have

$$N_{\mathscr{P}(\mathscr{A}),k}(S_1, \ldots, S_k) = \mathbf{val}_{\mathscr{A},k}(N_{\mathbf{Tree},k})(S_1, \ldots, S_k). \tag{A.3}$$

Let us denote respectively by S and S' the left- and the right-hand sides of this equality. We have

$$S = N'_{\mathscr{P}(\mathscr{A}),k+1}(S_1, \ldots, S_k, S), \tag{A.4}$$

$$S' = \mathbf{val}_{\mathscr{A},k}(L)(S_1, \ldots, S_k), \tag{A.5}$$

where L is defined as $N_{\mathbf{Tree},k}$ and satisfies the equality

$$L = N'_{\mathbf{Tree},k+1}[L/y]. \tag{A.6}$$

Furthermore, S is the largest subset of $\mathscr{Q}$ that satisfies (A.4), and L is the largest subset of $\mathbf{T}^\infty(A \cup X_k)$ that satisfies (A.6). By the induction hypothesis, we can assume that

$$N'_{\mathscr{P}(\mathscr{A}),k+1} = \mathbf{val}_{\mathscr{A},k+1}(N'_{\mathbf{Tree},k+1}) \tag{A.7}$$

Hence we get

$$\begin{aligned} S' &= \mathbf{val}_{\mathscr{A},k}(L)(S_1, \ldots, S_k) \\ &= \mathbf{val}_{\mathscr{A},k}(N'_{\mathbf{Tree},k+1}[L/y])(S_1, \ldots, S_k) \\ &= \mathbf{val}_{\mathscr{A},k+1}(N_{\mathbf{Tree},k+1})(S_1, \ldots, S_k, \mathbf{val}_{\mathscr{A},k}(L)(S_1, \ldots, S_k)) \quad \text{by Lemma A.2} \\ &= N'_{\mathscr{P}(\mathscr{A}),k+1}(S_1, \ldots, S_k, S') \quad \text{by (A.7) and (A.5).} \end{aligned}$$

Hence S' is a solution of (A.4). It follows that $S' \subseteq S$. It remains to establish that $S \subseteq S'$. We consider two cases.

Case 1: $y \in N'_{\mathbf{Tree},k+1}$. Then L defined as $(vy.N')_{\mathbf{Tree},k}$ is equal to $\mathbf{T}^\infty(A \cup X_k)$, since we are in Case 1 of Remark A.1. By (A.5) $S' = \mathbf{val}_{\mathscr{A},k}(L)(S_1, \ldots, S_k) \supseteq \mathbf{val}_{\mathscr{A}}(\mathbf{T}^\infty(A)) = \mathscr{Q}$. Hence we have $S \subseteq S'$ and $S = S'$.

Case 2: $y \notin N'_{\mathbf{Tree},k+1}$. Let $q \in S$ defined as $N_{\mathscr{P}(\mathscr{A}),k}(S_1, \ldots, S_k)$. By (A.4) and (A.7) $q \in \mathbf{val}_{\mathscr{A},k+1}(N'_{\mathbf{Tree},k+1})(S_1, \ldots, S_k, S)$. Hence, there exists a tree $t \in N'_{\mathbf{Tree},k+1}$ and an

$(S_1, \ldots, S_k, S)$-run r of $\mathscr{A}$ on t with root state q. For every node w of t that is an occurence of y, we let $q_w = r(w)$. We have $q_w \in S$. By the same argument as above, we can find $t_w \in N'_{\mathbf{Tree},k+1}$ and an $(S_1, \ldots, S_k, S)$-run r_w of $\mathscr{A}$ on t_w with root-state q_w. We let t' be obtained by substituting in t the tree t_w for each w as above. The runs r and r_w can be combined and form an $(S_1, \ldots, S_k, S)$-run of $\mathscr{A}$ on t' with root-state q. The above argument can be repeated: the occurrences of y in t' can be replaced by trees in $N'_{\mathbf{Tree},k+1}$. Since $y \notin N'_{\mathbf{Tree},k+1}$, the occurrences of y are deeper and deeper and by repeating the argument infinitely many times, one obtains a tree in L (by Remark A.1) and an $(S_1, \ldots, S_k)$-run of $\mathscr{A}$ on this tree with root-state q. This proves that $q \in S'$.

We have proved that $S \subseteq S'$, hence that $S = S'$.

Finally, we consider the case of $N = \mu y.N'$ where N' is a μ-term with free variables in X_k and y as identical to x_{k+1}. We let $S_1, \ldots, S_k \subseteq \mathscr{Q}$ and we have

$$N_{\mathscr{P}(\mathscr{A}),k}(S_1, \ldots, S_k) = \bigcup \{ f^\alpha(\emptyset)/\alpha \text{ is an ordinal} \}, \tag{A.8}$$

where for every $S, f(S) = N'_{\mathscr{P}(\mathscr{A}),k+1}(S_1, \ldots, S_k, S)$, $f^0(S) = S$, $f^{\alpha+1}(S) = f(f^\alpha(S))$, $f^\beta(S) = \bigcup \{ f^\alpha(S)/\alpha < \beta \}$ if β is a limit ordinal. Similarly,

$$N_{\mathbf{Tree},k} = \bigcup \{ g^\alpha(\emptyset)/\alpha \text{ is an ordinal} \}, \tag{A.9}$$

where $g(L) = N'_{\mathbf{Tree},k+1}[L/y]$ for every set of trees L. We claim that for every ordinal α

$$\mathbf{val}_{\mathscr{A},k}(g^\alpha(\emptyset))(S_1, \ldots, S_k) = f^\alpha(\emptyset). \tag{A.10}$$

This is easily proved by induction on α: for the case $\alpha = \beta + 1$, we observe that for every subset L of $\mathbf{T}^\infty(A \cup X_k)$

$$\mathbf{val}_{\mathscr{A},k}(g(L))(S_1, \ldots, S_k) = f(\mathbf{val}_{\mathscr{A},k}(L))(S_1, \ldots, S_k)) \tag{A.11}$$

which follows from (A.7) (the induction hypothesis on N) and Lemma A.2, and we use this for $L = g^\beta(\emptyset)$: this gives (A.10) by using the induction hypothesis on α. One proves (A.10) for a limit ordinal α by using the fact that the mapping associating $\mathbf{val}_{\mathscr{A},k}(L)(S_1, \ldots, S_k)$ with L is monotone and continous. □

Lemma A.4 (Niwinski [19]). *For every Rabin recognizable set K of A-trees, one can construct a closed μ-term N such that $N_{\mathbf{Tree},0} = K$.*

The set of states of the reduced automaton of a tree automaton $\mathscr{A}$ is defined by a μ-term as $(\nu x.(a(x, x) \cup b(x, x) \cup \ldots\ldots))_{\mathscr{P}(\mathscr{A})}$ where the union extends to all symbols $a, b, \ldots$ of A.

Lemma A.5. *For every Rabin recognizable set K of A-trees, one can construct a closed μ-term N such that, for every finite or infinite tree automaton $\mathscr{A}$ over A, we have*

$$N_{\mathscr{P}(\mathscr{B})} = \{ q \in \mathscr{Q} \mid L(\mathscr{A}, q) \cap K \neq \emptyset \}$$

where $\mathscr{B}$ is the reduced automaton of $\mathscr{A}$.

Proof. We let N be the μ-term that defines K (by Lemma A.4). We have by Lemma A.3:

$$\begin{aligned} N_{\mathscr{P}(\mathscr{B})} &= \mathbf{val}_{\mathscr{B}}(N_{\mathrm{Tree},0}) \\ &= \mathbf{val}_{\mathscr{B}}(K) \\ &= \{q \in \mathscr{Q}' \mid L(\mathscr{B}, q) \cap K \neq \emptyset\} \\ &= \{q \in \mathscr{Q} \mid L(\mathscr{A}, q) \cap K \neq \emptyset\} \end{aligned}$$

since for every state q in $\mathscr{Q}'$, the set of states of $\mathscr{B}$, we have $L(\mathscr{B}, q) = L(\mathscr{A}, q)$ and $L(\mathscr{A}, q)$ is empty for q not in $\mathscr{Q}'$. □

Lemma A.6. *For every μ-term M over A with free variables in $\{x_1, \ldots, x_k\}$, one can construct an MS-formula $\varphi_M(X, Y, X_1, \ldots, X_k)$ such that, for every finite or infinite tree automaton $\mathscr{A}$ over A with set of states $\mathscr{Q}$, for all subsets $X, Y, X_1, \ldots, X_k$ of $\mathscr{Q}$ we have*

$$\mathscr{A} \models \varphi_M(X, Y, X_1, \ldots, X_k)$$

if and only if $X_1, \ldots, X_k \subseteq Y$ and $X = N_{\mathscr{P}(\mathscr{B}),k}(X_1, \ldots, X_k)$ where $\mathscr{B}$ is the restriction of $\mathscr{A}$ to Y.

Proof. Straightforward construction using an induction on the structure of M. □

Proof of Theorem 3.1. It follows from Lemmas A.5 and A.6 that one can construct an MS-formula $\psi(X)$ such that, for every tree automaton $\mathscr{A}$ over A we have

$$\mathscr{A} \models \psi(X) \text{ if and only if } X = \{q \in \mathscr{Q} \mid L(\mathscr{A}, q) \cap K \neq \emptyset\}.$$

In the construction of ψ using Lemma A.6, we use as set Y the set of states of the reduced automaton of $\mathscr{A}$ and the fact that this set is definable by a μ-term, hence by an MS formula. Finally

$$\mathscr{A} \models \exists X(\psi(X) \wedge \text{“}X \cap F \neq \emptyset\text{”}) \text{ if and only if } L(\mathscr{A}) \cap K \neq \emptyset,$$

where “$X \cap F \neq \emptyset$” stands for the formula saying that $X \cap F \neq \emptyset$, and F denotes the set of accepting states of $\mathscr{A}$. □

References

[1] A. Arnold, *Systèmes de transitions finis et sémantique des processus communicants* (Masson, Paris, 1992); English translation: *Finite Transition Systems* (Prentice Hall, Englewood Cliffs, NJ, 1994).

[2] A. Arnold and D. Niwinski, Fixed-point characterization of weak monadic logic definable sets of trees, in: M. Nivat and A. Podelski, eds., *Tree Automata and Languages* (Elsevier, Amsterdam 1992) 159–188.

[3] B. Courcelle, A representation or trees by languages, *Theoret. Comput. Sci.* **6** (1978) 255–279; **7** (1978) 25–55.

[4] B. Courcelle, Fundamental properties of infinite trees, *Theoret. Comput. Sci.* **25** (1983) 95–169.

[5] B. Courcelle, Equivalences and transformations of regular systems, *Theoret. Comput. Sci.* **42** (1986) 1–122.
[6] B. Courcelle, The monadic second-order logic of graphs II: Infinite graphs of bounded width, *Math. Systems Theory* **21** (1989) 187–222.
[7] B. Courcelle, The monadic second-order logic of graphs I: Recognizable sets of finite graphs, *Inform. and Comput.* **85** (1990) 12–75.
[8] B. Courcelle, The monadic second-order logic of graphs IV: Definability properties of equational graphs, *Ann. Pure Appl. Logic* **49** (1990) 193–255.
[9] B. Courcelle, The monadic second-order logic of graphs X: Linear orders, *Theoret. Comput. Sci.*, to appear.
[10] B. Courcelle, Monadic second order graph transductions: A survey, *Theoret. Comput. Sci.* **126** (1994) 53–75
[11] B. Courcelle and I. Walukiewicz, Coverings of graphs, behaviours of transition systems and monadic second-order logic, manuscript, October 1994.
[12] M. Dam, CTL* and ECTL* as fragments of the modal μ-calculus, *Theoret. Comput. Sci.* **126** (1994) 77–96
[13] A. Emerson, Temporal and modal logics, in: J. Van Leeuwen, ed., *Handbook of Theoretical Computer Science, Vol. B* (Elsevier, Amsterdam, 1990) 995–1072.
[14] J. Gallier, DPDAS's in atomic normal form and applications to equivalence problems, *Theoret. Comput. Sci.* **14** (1981) 155–186; Corrigendum **19** (1982) 229.
[15] F. Gecseg and M. Steinby, *Tree Automata* (Academie Kiado, Budapest 1984).
[16] J. Hopcroft and J. Ullman, *Introduction to Automata Theory, Languages and Computation* (Addison-Wesley, Reading, MA, 1990).
[17] J. Mezei and J. Wright, Algebraic automata and context-free sets, *Inform. and Control* **11** (1967) 3–29.
[18] D. Niwinski, On fixed point clones, in: *Proc 13th ICALP*, Lecture Notes in Computer Science, Vol. 226 (Springer, Berlin, 1986) 464–473.
[19] D. Niwinski, Fixed points vs. infinite generation, in: *Proc 3rd Symp. on Logic in Comput. Sci.* (1988) 402–409.
[20] M. Rabin, A simple method for undecidability proofs and some applications, in: Y. Bar-Hillel, ed., *Logic, Methodology and Philisophy of Science II* (North-Holland, Amsterdam, 1965) 58–68.
[21] W. Thomas, Automata on infinite objects, in: J. Van Leeuwen, ed., *Handbood of Theoretical Computer Science, Vol. B* (Elsevier, Amsterdam, 1990) 133–192.
[22] W. Thomas, On logics, tillings and automata, in: *Proc. 18th ICALP*, Lecture Notes Computer Science. Vol. 510 (Springer, Berlin, 1991) 441–454.