# Automata Theory and Infinite Transition Systems *

Wolfgang Thomas
Lehrstuhl Informatik 7, RWTH Aachen, Germany
thomas@informatik.rwth-aachen.de

**Abstract**

These lecture notes report on the use of automata theory in the study of infinite transition systems. This application of automata is an important ingredient of the current development of "infinite-state system verification", and it provides an introduction into the field of "algorithmic model theory", a study of infinite models with an emphasis on computability results.

## 1 Introduction

The analysis of infinite transition systems is of central interest in infinite-state system verification and at the same time one of the most promising application domains of automata theory. These lecture notes give an overview over some topics which are currently studied in this field.

The classical set-up of algorithmic verification is defined by two items: a transition system as a model of a "system" (program, protocol, control unit), and a specification given by a logical formula which expresses some desired behaviour. The model-checking problem is the question "Given a transition graph $G$ and a formula $\varphi$, does $G$ satisfy $\varphi$?".

Prominent logics in this context are the temporal logics LTL (linear-time temporal logic) and CTL (computation tree logic). An LTL formula expresses a property of a given infinite path (execution sequence) through the transition graph under consideration, whereas a CTL-formula expresses a property of the infinite tree of execution sequences, starting from a given initial state. Common to both logics is the view that the non-terminating behaviour of the system (captured by its infinite paths) is to be analyzed.

A conceptual basis for solving the model-checking problem is the framework of automata over infinite words and infinite trees. For example, the decidability of the model-checking problem for LTL can simply be obtained by a transformation of LTL-formulas into Büchi automata. The model-checking problem is thus reduced to a problem about the relation of a transition graph $G$ and an

*Lecture Notes for the EMS Summer School CANT (Combinatorics, Automata and Number Theory), May 2006, University of Liège. Research reported in this paper was partially supported by the DAAD Procope Project "Finitely Presented Infinite Graphs" in collaboration with the group of D. Caucal, Rennes.

automaton $\mathcal{A}$ derived from the specification $\varphi$. This relation can be treated completely in the domain of automata (transition systems).

A weakness of this approach is the fact that the system graph under consideration is assumed to be finite. Since more than a decade, a new trend in verification is to consider more modest means of specification but to extend the domain of transition graphs to include certain infinite structures. Since these structures serve as inputs to the (verification) algorithms, a finitary presentation is needed. As it turned out, finite automata are an essential tool for this task, since they offer very natural ways to specify infinite structures.

The purpose of these lectures is to report on some fundamental classes of infinite models (in our case: transition graphs) such that the model-checking problem is decidable for interesting properties. So the emphasis is on infinite systems rather than infinite behaviour. Our presentation is far from complete; it is biased towards state-based models. An alternative and equally fundamental approach for introducing infinite models, which is not discussed here, is to extend finite transition graphs by infinite data structures, for example over the natural or real numbers (as in timed systems) and to use the framework of arithmetic (e.g., equalities or inequalities between polynomials) for specifications.

## 2 Technical Preliminaries

We consider structures in the format of edge-labelled and vertex-labelled transition graphs:

$$G = (V, (E_a)_{a\in\Sigma}, (P_b)_{b\in\Sigma'})$$

where $V$ is the (at most countable) set of vertices (in applications: "states") where $E_a \subseteq V \times V$ (for a symbol $a$ from a finite alphabet $\Sigma$) is the set of $a$-labelled edges, and where $P_b \subseteq V$ is the set of $b$-labelled vertices (in applications representing a state property) with $b \in \Sigma'$. We write $E$ for the union of the $E_a$.

More generally, one can consider relational structures $\mathcal{A} = (A, R_1^{\mathcal{A}}, \ldots, R_k^{\mathcal{A}})$, where the $R_i^{\mathcal{A}}$ are relations of possibly different arities over $A$, say $R_i^{\mathcal{A}}$ of arity $n_i$. In the sequel we stay with transition graphs for ease of notation and for their significance in verification.

Central examples of transition graphs are

- *Kripke structures*, which are graphs of the form $G = (V, E, (P_b))$, where each $P_b$ collects states which satisfy the same atomic propositions under consideration,
- the ordering $(\mathbb{N}, <)$ of the natural numbers,
- the *binary tree* $T_2 = (\{0,1\}^*, S_0, S_1)$ where $S_i = \{(w, wi) \mid w \in \{0,1\}^*\}$ (analogously, the $n$-ary tree $T_n = (\{0, \ldots, n-1\}^*, S_0^n, \ldots, S_{n-1}^n)$ is defined)

Let us list the logical systems considered in these lectures.

*First-order logic* FO over the signature with the symbols $E_a, P_b$ is built up from variables $x, y, \ldots$ and atomic formulas $x = y$, $E_a(x, y)$, $P_b(x)$ where $x, y$ are first-order variables, using the standard propositional connectives $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$ and the quantifiers $\exists, \forall$.

The reachability relation over $G$ is the relation $E^*$ defined by

$$E^*(u, v) \quad \Leftrightarrow \quad \exists v_0 \ldots v_k \in V : v_0 = u, (v_i, v_{i+1}) \in E \ \ (i < k), v_k = v$$

It is well-known that $E^*$ is not FO-definable. We call FO(R) the logic obtained from FO by adjoining a symbol for the reachability relation $E^*$ to the signature.

*Monadic second-order logic* MSO is obtained by adjoining variables $X, Y, \ldots$ for sets of elements (of the universe $V$ under consideration) and atomic formulas $X(y)$, meaning that the element $y$ is in the set $X$. We note that MSO encompasses FO(R), since we can express $E^*(x, y)$ by the formula saying that each set which contains $x$ and is closed under $E$ must contain $y$.

We use the standard notations; e.g. $G \models \varphi[v]$ indicates that $G$ satisfies the formula $\varphi(x)$ with the element $v$ as interpretation of $x$. Given a formula $\varphi(x_1, \ldots, x_n)$, the relation defined by it in $G$ is

$$\varphi^G = \{(v_1, \ldots, v_n) \in V^n \mid G \models \varphi[v_1, \ldots, v_n]\}$$

MSO encompasses most standard temporal logics.

We shall address two methods for constructing infinite transition graphs where model-checking (with respect to MSO or FO(R)) is decidable. First we explain "regular" internal representations of infinite transition graphs, using finite automata over strings or trees for specification. Secondly, we use fundamental model constructions – namely, interpretation and unfolding – to generate infinite models.

# 3 Rational and Automatic Graphs

A convenient way to introduce finitely presented infinite structures is to use words over some alphabet as names of vertices, regular languages for state properties, and automaton-definable relations over words for the edge relations. For the latter, there is a large reservoir of options, considering finite-state tranducers or word rewriting systems of different kinds. In this section we consider the two most prominent examples: the rational relations and the automatic (or: synchronized rational) relations.

## 3.1 Rational Graphs

A relation $R \subseteq \Gamma^* \times \Gamma^*$ is *rational* if it can be defined by a regular expression starting from the atomic expressions $\emptyset$ (denoting the empty relation) and $(u, v)$ for words $u, v$ (denoting the relation $\{(u, v)\}$) by means of the operations union, concatenation (applied componentwise), and iteration of concatenation (Kleene star). An alternative characterization of these relations involves a model of nondeterministic automaton which works one-way from left to right, but asynchronously, on the two components of an input $(w_1, w_2) \in \Gamma^* \times \Gamma^*$.

*Example* 1. Consider the suffix relation $\{(w_1, w_2) \mid w_1 \text{ is a suffix of } w_2\}$. A corresponding automaton (nondeterministic transducer) would progress with its reading head on the second component $w_2$ until it guesses that the suffix $w_1$ starts; this, in turn, can be checked by moving the two reading heads on the two components simultaneously, comparing $w_1$ letter by letter with the remaining suffix of $w_2$.

A *rational transition graph* has the form $G = (V, (E_a)_{a \in \Sigma}, (P_b)_{b \in \Sigma'})$ where $V$ and the sets $P_b$ are regular sets of words over an auxiliary alphabet $\Gamma$ and where each $E_a \subseteq \Gamma^* \times \Gamma^*$ is a rational relation.

Clearly, each rational graph is recursive in the sense that the edge relations and the vertex properties are decidable. However, very simple properties of rational graphs may be undecidable.

**Theorem 2.** *For each instance $(\overline{u}, \overline{v})$ of PCP (Post's Correspondence Problem) one can construct a rational graph $G_{(\overline{u},\overline{v})}$ such that $(\overline{u}, \overline{v})$ has a solution iff $G_{(\overline{u},\overline{v})}$ has a loop edge from a vertex to itself.*

*Proof.* Given a PCP-instance $(\overline{u}, \overline{v}) = ((u_1, \ldots, u_m), (v_1, \ldots, v_m))$ over an alphabet $\Gamma$, we specify a rational graph $G_{(\overline{u},\overline{v})} = (V, E)$ as follows. The vertex set $V$ is $\Gamma^*$. The edge set $E$ consists of the pairs of words of the form $(u_{i_1} \ldots u_{i_k}, v_{i_1} \ldots v_{i_k})$ where $i_1, \ldots, i_k \in \{1, \ldots, m\}$ and $k \geq 1$. Clearly, an asynchronously progressing nondeterministic automaton can check whether a word pair $(w_1, w_2)$ belongs to $E$; basically the automaton has to guess successively the indices $i_1, \ldots, i_k$ and at the same time to check whether $w_1$ starts with $u_{i_1}$ and $w_2$ starts with $v_{i_1}$, whether $w_1$ continues by $u_{i_2}$ and $w_2$ by $v_{i_2}$, etc. So the graph $G_{(\overline{u},\overline{v})}$ is rational. Clearly, in this graph there is an edge from some vertex $w$ back to the same vertex $w$ iff the PCP-instance $(\overline{u}, \overline{v})$ has a solution (namely by the word $w$). □

The existence of a loop edge $(w, w)$ is expressible by the first-order formula $\exists x\ E(x, x)$. Hence we obtain the following result.

**Corollary 3** (Morvan [15])**.** *There is no algorithm which, given a presentation of a rational graph $G$ and a first-order sentence $\varphi$, decides whether $G \models \varphi$.*

Let us construct a single rational graph with an undecidable first-order theory. We give a construction from [21].

**Theorem 4.** *There is a rational graph $G$ with an undecidable first-order theory; in other words, the problem*

*Given a first-order sentence $\varphi$, does $G$ satisfy $\varphi$?*

*is undecidable.*

*Proof.* We use a universal Turing machine $M$ and the encoding of its undecidable halting problem (for different input words $x$) into a family of PCP-instances. For simplicity of exposition, we refer here to the standard construction of the undecidability of PCP as one finds it in textbooks (see [13, Section 8.5]): A Turing machine $M$ with input word $x$ is converted into a PCP-instance $((u_1, \ldots, u_m), (v_1, \ldots, v_m))$ over an alphabet $A$ whose letters are the states and tape letters of $M$ and a symbol $\#$ (for the separation between $M$-configurations in $M$-computations). If the input word is $x = a_1 \ldots a_n$, then $u_1$ is set to be the initial configuration word $c(x) := \#q_0 a_1 \ldots a_n$ of $M$; furthermore we always have $v_1 = \#$, and $u_2, \ldots, u_m, v_2, \ldots, v_m$ only depend on $M$. Then the standard construction (of [13]) ensures the following:

$$M \text{ halts on input } x$$

iff the PCP-instance $((c(x), u_2, \ldots, u_m), (\#, v_2, \ldots, v_m))$ has a special solution.

Here a special solution is given by an index sequence $(i_2, \ldots, i_k)$ such that $c(x)u_{i_2} \ldots u_{i_k} = \#v_{i_2} \ldots v_{i_k}$.

Let $G$ be the graph as defined from these PCP-instances as above: The vertices are the words over $A$, and we have a single edge relation $E$ with $(w_1, w_2) \in E$ iff there are indices $i_2, \ldots, i_k$ and a word $x$ such that $w_1 = c(x)u_{i_2} \ldots u_{i_k}$ and $w_2 = \#v_{i_2} \ldots v_{i_k}$. Clearly $G$ is rational, and we have an edge from a word $w$ back to itself if it is induced by a special solution of some PCP-instance $((c(x), u_2, \ldots, u_m), (\#, v_2, \ldots, v_m))$.

In order to address the input words $x$ explicitly in the graph, we add further vertices and edge relations $E_a$ for $a \in A$. A $c(x)$-labelled path via the new vertices will lead to a vertex of $G$ with prefix $c(x)$; if the latter vertex has an edge back to itself, then a special solution for the PCP-instance $((c(x), u_2, \ldots, u_m), (\#, v_2, \ldots, v_m))$ can be inferred. The new vertices are words over a copy $\underline{A}$ of the alphabet $A$ (consisting of the underlined versions of the $A$-letters). For any word $c(x)$ we shall add the vertices which arise from the underlined versions of the proper prefixes of $c(x)$, and we introduce an $E_a$-edge from any such underlined word $w$ to $w\underline{a}$ (including the case $w = \epsilon$). There are also edges to non-underlined words: We have an $E_a$-edge from the underlined version of $w$ to any non-underlined word which has $wa$ as a prefix. Call the resulting graph $G'$. It is easy to see that $G'$ is rational.

By construction of $G'$, the PCP-instance $((c(x), u_2, \ldots, u_m), (\#, v_2, \ldots, v_m))$ has a special solution iff

> in $G'$ there is a path, labelled with the word $c(x)$, from the vertex $\epsilon$ to a vertex which has an edge back to itself.

Note that the vertex $\epsilon$ is definable as the only one with outgoing $E_a$-edges but without any ingoing $E_a$-edge. Thus the above condition is formalizable by a first-order sentence $\varphi_x$, using variables for the $|c(x)| + 1$ vertices of the desired path. Altogether we obtain that the Turing machine $M$ halts on input $x$ iff $G' \models \varphi_x$. □

These results show that for applications in verification, rational graphs in general are too extensive to allow interesting algorithmic solutions.

## 3.2 Automatic Graphs

In *automatic (or synchronized rational) relations* a more restricted processing of an input $(w_1, w_2)$ by an automaton is required than in the asynchronous mode as mentioned for nondeterministic transducers: We now require that an automaton scans a pair $(w_1, w_2)$ of words strictly in parallel letter by letter. Thus one can assume that the automaton reads letters from $\Gamma \times \Gamma$ if $w_1, w_2 \in \Gamma^*$. In order to cover the case that $w_1, w_2$ are of different length, one assumes that the shorter word is prolonged by dummy symbols \$ to achieve equal length. Let $[w_1, w_2]$ be the word over the alphabet $(\Gamma \times \Gamma) \cup ((\Gamma \cup \{\$\}) \times \Gamma) \cup (\Gamma \times (\Gamma \cup \{\$\}))$ associated with $(w_1, w_2)$.

Thus, a relation $R \subseteq \Gamma^* \times \Gamma^*$ induces the language $L_R = \{[w_1, w_2] \mid (w_1, w_2) \in R\}$. The relation $R$ is called *automatic* if the associated language $L_R$ is regular. From this definition it is immediately clear that the automatic relations share many good properties which are familiar from the theory of regular word languages. For example, one can reduce nondeterministic automata which recognize a word relation synchronously to deterministic automata, a fact which does not hold for the transducers in the context of rational relations.

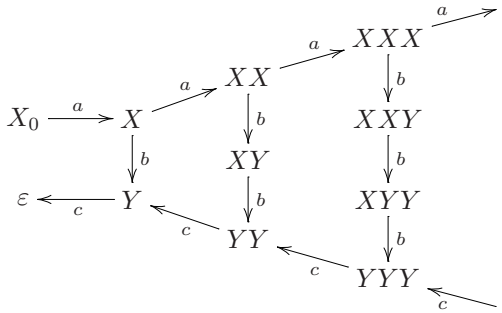



Figure 1: An automatic graph

A graph $(V, (E_a)_{a\in\Sigma}, (P_b)_{b\in\Sigma'})$ is called *automatic* if $V$ and each $P_b \subseteq V$ are regular languages over an alphabet $\Gamma$ and each edge relation $E_a \subseteq \Gamma^* \times \Gamma^*$ is automatic.

*Example* 5. The infinite two-dimensional grid $G_2 := (\mathbb{N} \times \mathbb{N}, E_a, E_b)$ (with $E_a$-edges $((i,j),(i,j+1))$ and $E_b$-edges $((i,j),(i+1,j))$) is automatic: It can be obtained using the words in $X^*Y^*$ as vertices, whence the edge relations become $E_a = \{(X^iY^j, X^iY^{j+1}) \mid i,j \geq 0\}$ and $E_b = \{(X^iY^j, X^{i+1}Y^j) \mid i,j \geq 0\}$, which both are clearly automatic.

*Example* 6. Consider the transition graph over $\Gamma = \{X_0, X, Y\}$ where there is an $a$-edge from $X_0$ to $X$ and from $X^i$ to $X^{i+1}$ (for $i \geq 1$), a $b$-edge from $X^iY^j$ to $X^{i-1}Y^{j+1}$ (for $i \geq 1, j \geq 0$), and a $c$-edge from $Y^{i+1}$ to $Y^i$ (for $i \geq 0$). We obtain the automatic graph of Figure 1. If $X_0$ is taken as initial and $\epsilon$ as final state, the resulting infinite automaton recognizes the context-sensitive language consisting of the words $a^ib^ic^i$ for $i \geq 1$.

*Example* 7. Let $T_2' = (\{0,1\}^*, S_0, S_1, \leq, \lambda)$ be the expansion of the binary tree $T_2 = (\{0,1\}^*, S_0, S_1)$ by the prefix relation $\leq = \{(u,v) \in \{0,1\}^* \mid u \text{ is a prefix of } v\}$ and the "equal length relation" $\lambda = \{(u, v \in \{0,1\}^* \mid \ |u| = |v|\}$. Clearly $T_2'$ is automatic.

In the literature, the automatic relations appear also under several other names, among them "regular", "sequential", "synchronized rational".

We give another example which illustrates the power of automatic relations.

*Example* 8. Given a Turing machine $M$ with state set $Q$ and tape alphabet $\Gamma$, we consider the graph $G_M$ with vertex set $V_M = \Gamma^*Q\Gamma^*$, considered as the set of $M$-configurations. By an appropriate treatment of the blank symbol, we can assume that the length difference between two successive $M$-configurations is at most 1; thus it is easy to see that the relation $E_M$ of word pairs which consist of successive $M$-configurations is automatic. So the configuration graph $G_M = (V_M, E_M)$ is automatic.

Let us show that first-order properties of automatic graphs are decidable:

**Theorem 9.** *The FO-theory of an automatic graph is decidable.*

*Proof.* Let $G = (V, (E_a)_{a\in\Sigma}, (P_b)_{b\in\Sigma'})$ be a graph with an automatic presentation over $\Gamma$. We verify inductively over FO-formulas $\varphi(x_1, \ldots, x_n)$ that the

following relation is automatic:

$$R_\varphi = \{(w_1, \ldots, w_n) \mid (G, w_1, \ldots, w_n) \models \varphi(x_1, \ldots, x_n)\}$$

For the atomic formulas, this is clear by the automatic presentation of $G$. In the induction step, the Boolean connectives are easy due to the closure of regular sets under Boolean operations. (Note that the complement is applied with respect to the set of words $[w_1, w_2]$, i.e. the words where the letter \$ may occur only in one component, and only at the end.) For the step of existential quantification, assume – as a typical case – that the binary relation $R$ is recognized by the finite automaton $\mathcal{A}$, say with final state set $F$. We have to verify that also

$$S = \{w_1 \in \Gamma^* \mid \exists w_2 : (w_1, w_2) \in R\}$$

is automatic (i.e. in this unary case: a regular language).

The automaton checking $S$ is obtained from $\mathcal{A}$ by a projection of the input letters to the first components and by an extension of $F$ to a set $F'$. A state is included in $F'$ if some (possibly empty) sequence of letters $(\$, a)$ leads to $F$. (This covers the case that the component $w_2$ is longer than $w_1$.)

The truth of an FO-sentence $\varphi$ amounts to nonemptiness of an automaton which results from this inductive construction over the subformulas of $\varphi$. For $\varphi$ itself an input-free automaton (with unlabelled transitions) is obtained; the truth of $\varphi$ is signalled by the existence of a successful run of this input-free automaton. □

A copy of this argument shows that Presburger arithmetic, the FO-theory of the structure $(\mathbb{N}, +)$, is decidable ([3]). For this purpose, one codes an $n$-tuple of natural numbers by the $n$-tuple of the reversed binary representations. The atomic formula $x_1 + x_2 = x_3$ defines a ternary relation over $\{0, 1\}^*$ which is automatic, since the usual check that an addition of binary numbers is correct can be done by a finite automaton. For the logical connectives one proceeds as in the proof above.

If we extend the logic FO slightly by including the reachability relation $E^*$, then this decidability result fails.

**Theorem 10.** *There is an automatic graph $G$ such that over $G$ the relation $E^*$ is undecidable.*

*Proof.* Consider the configuration graph $G_U$ of a universal Turing machine $U$, where the vertices are configuration words in $\Gamma^* Q \Gamma^*$ ($\Gamma$ is the tape alphabet of $U$, and $Q$ is its set of states). Assume that the machine $U$ halts precisely in configurations $xq_sy$ with a stop state $q_s$. The one-step relation of pairs $(xqy, x'q'y')$ of configuration words is automatic. Then an arbitrary Turing machine $M$ accepts the input word $w$ iff over $G_U$, from the configuration $q_0\mathrm{code}(M)w$ a halting configuration can be reached. □

This result is one of the main obstacles in developing a framework for model-checking over infinite systems: The automatic graphs are a very natural framework for modelling interesting infinite systems, but most applications of model-checking involve some kind of reachability analysis. Current research tries to find good restrictions of the class of automatic graphs where the reachability

problem is still solvable. An important class of such graphs is presented in the next section.

Let us also look at a more ambitious problem than reachability: decidability of the monadic second-order theory of a given graph (or equivalently: the model-checking problem with respect to MSO-definable properties). Here we get undecidability already for automatic graphs with a much simpler transition structure than that of the graph $G_U$ of the previous theorem. The most prominent example is the infinite two-dimensional grid (introduced as an automatic graph in Example 5). Note that the reachability problem over the grid (say from a given vertex to another given vertex) is decidable.

**Theorem 11.** *The monadic second-order theory of the infinite two-dimensional grid $G_2$ is undecidable.*

*Proof.* The idea is to code the computations of Turing machines in a more uniform way than in the previous result. Instead of coding a Turing machine configuration by a single vertex and capturing the Turing machine steps directly by the edge relation, we now use a whole row of the grid for coding a configuration (by an appropriate coloring of its vertices with tape symbols and a Turing machine state). A computation of a Turing machine, say with $m$ states and $n$ tape symbols, is thus represented by a sequence of colored rows (using $m + n$ colors), i.e., by a coloring of the grid. (We can assume that even a halting computation generates a coloring of the whole grid, by repeating the final configuration ad infinitum.) In this view, the horizontal edge relation is used to progress in space, while the vertical one allows to progress in time. A given Turing machine $M$ halts on the empty tape iff there is a coloring of the grid with $m + n$ colors which

- represents the initial configuration (on the empty tape) in the first row,
- respects the transition table of $M$ between any two successive rows,
- contains a vertex which is colored by a halting state.

Such a coloring corresponds to a partition of the vertex set $\mathbb{N} \times \mathbb{N}$ of the grid into $m + n$ sets. One can express the existence of the coloring by saying "there exist sets $X_1, \ldots, X_{m+n}$ which define a partition and satisfy the requirements of the three items above". In this way one obtains effectively an MSO-sentence $\varphi_M$ such that $M$ halts on the empty tape iff $G_2 \models \varphi_M$. □

# 4 Prefix Rewriting and Pushdown Systems

The undecidability of the reachability problem over automatic graphs is no surprise to anyone who has heard about Semi-Thue systems, i.e. rewriting systems that allow to replace an infix $w$ in a word by another infix $w'$. The one-step relation over Turing machine configuration words is defined by an infix rewriting system. It is well-kown that the derivability relation of infix rewriting systems is in general undecidable.

As observed already by Büchi in 1964, the situation changes when we use prefix rewriting instead. Büchi showed that the words which are generated from a fixed word by a finite prefix rewriting system form a regular language. As an

application one obtains the well-known fact that the reachable global states of a pushdown automaton constitute a regular set. In the theory of program verification, the problem usually arises in a converse (but essentially equivalent) form: Here one starts with a (usually regular) set $T$ of vertices as "target set", and the problem is to find those words from which a word in $T$ is reachable by a finite number of prefix rewriting steps.

We introduce two types of graphs based on the idea of prefix rewriting. The first (and more restricted) version is the notion of pushdown graph, with edges corresponding to moves of a pushdown automaton. The second allows to capture infinitely many instances of prefix rewriting in a single rule.

A graph $G = (V, (E_a)_{a\in\Sigma})$ is called *pushdown graph* (over the label alphabet $\Sigma$) if it is the transition graph of the reachable global states of an $\epsilon$-free pushdown automaton. Here a pushdown automaton is of the form $\mathcal{P} = (P, \Sigma, \Gamma, p_0, Z_0, \Delta)$, where $P$ is the finite set of control states, $\Sigma$ the input alphabet, $\Gamma$ the stack alphabet, $p_0$ the initial control state, $Z_0 \in \Gamma$ the initial stack symbol, and $\Delta \subseteq P \times \Sigma \times \Gamma \times \Gamma^* \times P$ the transition relation. A global state (configuration) of the automaton is given by a control state and a stack content, i.e., by a word from $P\Gamma^*$. The graph $G = (V, (E_a)_{a\in\Sigma})$ is now specified as follows:

- $V$ is the set of configurations from $P\Gamma^*$ which are reachable (via finitely many applications of transitions of $\Delta$) from the initial global state $p_0Z_0$.
- $E_a$ is the set of all pairs $(p\gamma w, qvw)$ from $V^2$ for which there is a transition $(p, a, \gamma, v, q)$ in $\Delta$.

Then the edge relation $E$ coincides with the one-step derivation relation $p_1w_1 \vdash p_2w_2$ over $\mathcal{P}$, and the transitive closure $E^*$ with the derivability relation $\vdash^*$.

A more general class of graphs, which includes the case of vertices of infinite degree, consists of the "prefix-recognizable graphs" (proposed by Caucal in [5]). These graphs are introduced in terms of prefix-rewriting systems in which "control states" (as they occur in pushdown automata) are no longer used and where a word on the top of the stack (rather than a single letter) may be rewritten. Thus, a rewriting step can be specified by a triple $(u_1, a, u_2)$, describing a transition from a word $u_1w$ via letter $a$ to the word $u_2w$. The feature of infinite degree is introduced by allowing generalized rewriting rules of the form $U_1 \xrightarrow{a} U_2$ with regular sets $U_1, U_2$ of words. Such a rule leads to the (in general infinite) set of rewrite triples $(u_1, a, u_2)$ with $u_1 \in U_1$ and $u_2 \in U_2$. A graph $G = (V, (E_a)_{a\in\Sigma})$ is called *prefix-recognizable* if for some finite system $\mathcal{S}$ of such generalized prefix rewriting rules $U_1 \xrightarrow{a} U_2$ over an alphabet $\Gamma$, we have

- $V \subseteq \Gamma^*$ is a regular set,
- $E_a$ consists of the pairs $(u_1w, u_2w)$ where $u_1 \in U_1$, $u_2 \in U_2$ for some rule $U_1 \xrightarrow{a} U_2$ from $\mathcal{S}$, and $w \in \Gamma^*$.

*Example* 12. The structure $(\mathbb{N}, \mathrm{Succ}, <)$ is prefix recognizable. We write the structure as $(\mathbb{N}, E_a, E_b)$ and represent numbers by sequences over the one-letter alphabet with the symbol | only. So $V = |^*$, and the two relations $E_a, E_b$ are defined by the prefix rewriting rules $\varepsilon \xrightarrow{a} |$ and $\varepsilon \xrightarrow{b} |^+$.

The prefix-recognizable graphs coincide with the pushdown graphs when $\varepsilon$-rules are added to pushdown automata and edges are defined in terms of $\xrightarrow{\varepsilon}{}^*$ $\xrightarrow{a}$ $\xrightarrow{\varepsilon}{}^*$.

Let us compare the four classes of graphs introduced so far.

**Theorem 13.** *The pushdown graphs, prefix-recognizable graphs, automatic graphs, and rational graphs constitute, in this order, a strictly increasing inclusion chain of graph classes.*

*Proof.* For the proof, we first note that the prefix-recognizable graphs are clearly a generalization of the pushdown graphs and that the rational graphs generalize the automatic ones. To verify that a prefix-recognizable graph is automatic, we first proceed to an isomorphic graph which results from reversing the words under consideration, at the same time using suffix rewriting rules instead of prefix rewriting ones. Given this format of the edge relations, we can verify that it is automatic: Consider a word pair $(u_1w, u_2w)$ which results from the application of a suffix rewriting rule $U_1 \xrightarrow{a} U_2$, with regular $U_1, U_2$ and $u_1 \in U_1$, $u_2 \in U_2$. A nondeterministic automaton can easily check this property of the word pair by scanning the two components simultaneously letter by letter, guessing when the common prefix $w$ of the two components is passed, and then verifying (again proceeding letter by letter) that the remainder $u_1$ of the first component is in $U_1$ and the remainder $u_2$ of the second component is in $U_2$.

The strictness of the inclusions may be seen as follows. The property of having bounded degree separates the pushdown graphs from the prefix-recognizable ones (see Example 12). To distinguish the other graph classes, one may use logical decidability results. It will be shown in Section 5.1 that the monadic second-order theory of a prefix-recognizable graph is decidable, which fails for the automatic graphs. Furthermore, the first-order theory of an automatic graph is decidable, which fails in general for the rational graphs. □

Let us solve the reachability problem over pushdown graphs (we leave the case of prefix-recognizable graphs as an exercise). For a pushdown automaton $\mathcal{P} = (P, \Sigma, \Gamma, p_0, Z_0, \Delta)$ and $T \subseteq P\Gamma^*$ let

$$\mathrm{pre}^*(T) = \{pv \in P\Gamma^* \mid \exists qw \in T : pv \vdash^* qw\}$$

We prove the following fundamental result which goes back to Büchi [4]:

**Theorem 14.** *Given a pushdown automaton $\mathcal{P} = (P, \Sigma, \Gamma, p_0, Z_0, \Delta)$ and a finite automaton recognizing a set $T \subseteq P\Gamma^*$, one can compute a finite automaton recognizing* $\mathrm{pre}^*(T)$.

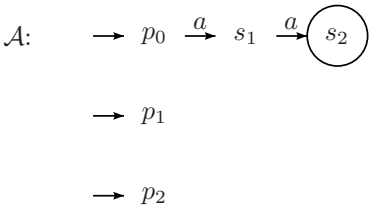
We can decide the reachability of a configuration $p_2w_2$ from $p_1w_1$ by setting $T = \{p_2w_2\}$ and checking whether the automaton recognizing $\mathrm{pre}^*(T)$ accepts $p_1w_1$.

The transformation of a given automaton $\mathcal{A}$ which recognizes $T$ into the desired automaton $\mathcal{A}'$ recognizing $pre^*(T)$ works by a simple process of "saturation", which involves adding more and more transitions but leaves the set of states unmodified. This construction, which improves the original one by Büchi regarding efficiency, appears in several papers, among them [8] and [12]; we follow the latter. It is convenient to work with $P$ as the set of initial states of $\mathcal{A}$; so a stack content $pw$ of the pushdown automaton is scanned by $\mathcal{A}$ starting from state $p$ and then processing the letters of $w$. This use of $P$ as the set of initial states of $\mathcal{A}$ motivates the term $P$-automaton in the literature. The $P$-automata we use for specifying $T$ do not have transitions into $P$; we call them *normalized*.

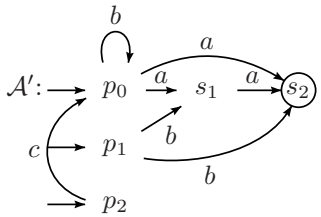
The saturation procedure is based on the following idea: Suppose a pushdown transition allows to rewrite the stack content $p\gamma w$ into $qvw$, and that the latter one is accepted by $\mathcal{A}$. Then the stack content $p\gamma w$ should also be accepted. If $\mathcal{A}$ accepts $qvw$ by a run starting in state $q$ and reaching, say, state $r$ after processing $v$, we enable the acceptance of $p\gamma w$ by adding a direct transition from $p$ via $\gamma$ to $r$. The saturation algorithm performs such insertions of transitions as long as possible.

**Saturation Algorithm:**
Input: $P$-automaton $\mathcal{A}$, pushdown system $\mathcal{P} = (P, \Gamma, \Delta)$
$\mathcal{A}_0 := \mathcal{A}$, $i := 0$
`REPEAT`:
  `IF` $pa \to p'v \in \Delta$ and $\mathcal{A}_i : p' \xrightarrow{v} q$ `THEN`
    add $(p, a, q)$ to $\mathcal{A}_i$ and obtain $\mathcal{A}_{i+1}$
    $i := i + 1$
`UNTIL` no transition can be added
$\overline{\mathcal{A}} := \mathcal{A}_i$
Output: $\mathcal{A}'$

As an example consider $\mathcal{P} = (P, \Gamma, \Delta)$ with $P = \{p_0, p_1, p_2\}$, $\Gamma = \{a, b, c\}$, $\Delta = \{(p_0a \to p_1ba), (p_1b \to p_2ca), (p_2c \to p_0b), (p_0b \to p_0)\}$ and $T = \{p_0aa\}$. The $P$-automaton for $T$ is the following:




Execution of the saturation algorithm introduces edges as indicated in the following figure. Insertion of $p_0 \xrightarrow{b} p_0$ is based on the rule $p_0b \to p_0$ and $\mathcal{A} : p_0 \xrightarrow{\varepsilon} p_0$, insertion of $p_2 \xrightarrow{c} p_0$ on the rule $p_2c \to p_0b$ and $\mathcal{A} : p_0 \xrightarrow{b} p_0$ (we denote by $\mathcal{A}$ here always the current automaton), insertion of $p_1 \xrightarrow{b} s_1$ on the rule $p_1b \to p_2ca$ and $\mathcal{A} : p_2 \xrightarrow{ca} s_1$, insertion of $p_0 \xrightarrow{a} s_2$ on the rule $p_0a \to p_1ba$ and $\mathcal{A} : p_1 \xrightarrow{ba} s_2$, and insertion of $p_1 \xrightarrow{b} s_2$ on the rule $p_1b \to p_2ca$ and $\mathcal{A} : p_2 \xrightarrow{ca} s_2$.




So for $T = \{p_0aa\}$ we extract the following result.

$$pre^*(T) = p_0b^*(a + aa) + p_1b + p_1ba + p_2cb^*(a + aa)$$

**Proposition 15.** *The Saturation Algorithm terminates and gives, for an input automaton $\mathcal{A}$ recognizing $T$, as output an automaton $\mathcal{A}'$ recognizing* $\mathrm{pre}^*(T)$.

*Proof.* Termination of the algorithm is clear since new transitions $(p, a, q)$ can be added only finitely often to the given automaton.

Next we have to show:

$$pw \in \mathrm{pre}^*(T) \Leftrightarrow \mathcal{A}' : p \xrightarrow{w} F$$

For the direction from left to right we use induction over the number $n \geq 0$ of steps to get to $T$:

$$pw \to^n ru \in T \;\Rightarrow\; \mathcal{A}' : p \xrightarrow{w} F$$

The case $n = 0$ is obvious. In the induction step assume $pw \to^{n+1} ru$ and $ru \in T$. We have to show that $\mathcal{A}'$ accepts $pw$. Consider the decomposition of the step sequence to $ru \in T$: $paw' \to p'vw' \to^n ru$ with $w = aw'$ and a pushdown transition $pa \to p'v$. The induction assumption gives $\mathcal{A}' : p' \xrightarrow{vw'} F$. So, there exists an $\mathcal{A}'$-state $q$ with $\mathcal{A}' : p' \xrightarrow{v} q \xrightarrow{w'} F$. Consequently, the saturation algorithm produces the transition $(p, a, q) \in \Delta_{\mathcal{A}'}$, and $pw$ is accepted by $\mathcal{A}'$.

For the direction from right to left we show

$$\mathcal{A}' : p \xrightarrow{w} q \implies \exists p'w' \in P\Gamma^* : \mathcal{A} \text{ such that } p' \xrightarrow{w'} q \wedge pw \vdash^* p'w'$$

For $q \in F$ (the final state-set of $\mathcal{A}$) we obtain the claim; note that $\mathcal{A} : p' \xrightarrow{w'} q$ says that $p'w' \in T$.

We denote by $\mathcal{A}_i$ the $P$-automaton which originates from $\mathcal{A}$ after $i$ insertions of new transitions by the saturation algorithm. We show inductively over $i$:

$$\text{If } \mathcal{A}_i : p \xrightarrow{w} q, \text{ then } \exists p'w' \in P\Gamma^* \text{ such that } \mathcal{A} : p' \xrightarrow{w'} q \wedge pw \vdash^* p'w'$$

The case $i = 0$ obvious. For the induction claim assume that $\mathcal{A}_{i+1} : p \xrightarrow{w} q$. Consider an accepting run $\mathcal{A}_{i+1} : p \xrightarrow{w} q$. Let $j$ be the number of applications of the $(i + 1)$-st transition. We prove the claim inductively over $j$. The case $j = 0$ is obvious (no use of the $(i + 1)$-th transition). For $j + 1$, consider the decomposition of $w$ in $w = uau'$ with

$$\mathcal{A}_i : p \xrightarrow{u} p_1, \quad \mathcal{A}_{i+1} : \underbrace{p_1 \xrightarrow{a} q_1}_{(i+1)\text{-st transition}} \quad \text{and } \mathcal{A}_{i+1} : q_1 \xrightarrow{u'} q$$

By induction (on $i$) we have $pu \vdash^* p'_1u_1$ with $\mathcal{A} : p'_1 \xrightarrow{u_1} p_1$. Since $\mathcal{A}$ is normalized, its initial state $p_1$ has no ingoing transitions, hence $u_1 = \varepsilon$ and $p_1 = p'_1$; thus $pu \vdash^* p_1$.

The saturation algorithm adds $(p_1, a, q_1)$ to $\mathcal{A}_i$. So, there are $p_2$ and a pushdown rule $p_1a \to p_2v$ with $\mathcal{A}_i : p_2 \xrightarrow{v} q_1$.

Finally, in the run on $u'$, the $(i + 1)$-st transition is used $\leq j$ times, so by induction assumption on $j$, we know for the run $\mathcal{A}_{i+1} : p_2 \xrightarrow{v} q_1 \xrightarrow{u'} q$ that there is $p'w'$ with $\mathcal{A} : p' \xrightarrow{w'} q$ and $p_2vu' \vdash^* p'w'$.

Altogether we have $pw = puau' \vdash^* p_1au' \vdash^* p_2vu' \vdash^* p'w' (\in T)$. □

The idea of the saturation algorithm has been transferred to many related problems, for example for checking "recurrent reachability" over pushdown graphs, for two-player reachability games played on pushdown graphs, and for reachability over transition graphs associated with tree rewriting systems (see below).

In the next section we shall introduce another technique for showing decidability results. This technique covers more than reachability properties, in fact all properties which are expressible in MSO-logic. The idea is to transfer the decidability of the MSO-theory from a given standard structure to other structures. We shall two consider types of model transformation: the "MSO-interpretations" and the process of "unfolding".

# 5 Operations preserving decidability

## 5.1 Interpretations

The starting point of this section is a deep and difficult decidability result, called "Rabin's Tree Theorem":

**Theorem 16.** (Rabin [18])
*The MSO-theory of the infinite binary tree $T_2$ is decidable.*

We do not enter the proof; a self-contained exposition is in [20].

Many other theories were shown decidable (already in Rabin's landmark paper [18]) using interpretations in the tree $T_2$. To show that the model-checking problem for a structure $S$ with respect to formulas of a logic $L$ is decidable, one proceeds as follows: One gives an MSO-description of $S$ within the binary tree $T_2$, and using this, one provides a translation of $L$-formulas $\varphi$ into MSO-formulas $\varphi'$ such that $S \models \varphi$ iff $T_2 \models \varphi'$. Taking $L = \mathrm{MSO}$, we see that an MSO-interpretation (i.e., a model description using MSO-formulas) preserves decidability of model-checking with respect to MSO-formulas.

Let us illustrate the idea of MSO-interpretation by showing that the result also holds for the structures $T_n$ for $n > 2$. As typical example consider $T_3 = (\{0,1,2\}^*, S_0^3, S_1^3, S_2^3)$. We obtain a copy of $T_3$ in $T_2$ by considering only the $T_2$-vertices in the set $T = (10 + 110 + 1110)^*$. A word in this set has the form $1^{i_1}0 \ldots 1^{i_m}0$ with $i_1, \ldots, i_m \in \{1,2,3\}$; and we take it as a representation of the element $(i_1 - 1) \ldots (i_m - 1)$ of $T_3$.

The following MSO-formula $\varphi(x)$ (written in abbreviated suggestive form) defines the set $T$ in $T_2$:

$$\forall Y[Y(x) \wedge \forall y((Y(y10) \vee Y(y110) \vee Y(y1110)) \rightarrow Y(y)) \ \rightarrow Y(\epsilon)]$$

It says that $x$ is in the closure of $\epsilon$ under 10-, 110-, and 1110-successors. The relation $\{(w, w10) | w \in \{0,1\}^*\}$ is defined by the following formula:

$$\psi_0(x,y) := \exists z(S_1(x,z) \wedge S_0(z,y))$$

With the analogous formulas $\psi_1$, $\psi_2$ for the other successor relations, we see that the structure with universe $\varphi^{T_2}$ and the relations $\psi_i^{T_2}$ restricted to $\varphi^{T_2}$ is isomorphic to $T_3$.

In general, an MSO-interpretation of a structure $\mathcal{A}$ in a structure $\mathcal{B}$ is given by a "domain formula" $\varphi(x)$ and, for each relation $R^{\mathcal{A}}$ of $\mathcal{A}$, say of arity $m$, an MSO-formula $\psi(x_1, \ldots, x_m)$, such that $\mathcal{A}$ with the relations $R^{\mathcal{A}}$ is isomorphic to the structure with universe $\varphi^{\mathcal{B}}$ and the relations $\psi^{\mathcal{B}}$ restricted to $\varphi^{\mathcal{B}}$.

Then for an MSO-sentence $\chi$ (in the signature of $\mathcal{A}$) one can construct a sentence $\chi'$ (in the signature of $\mathcal{B}$) such that $\mathcal{A} \models \chi$ iff $\mathcal{B} \models \chi'$. In order to obtain $\chi'$ from $\chi$, one replaces every atomic formula $R(x_1, \ldots, x_m)$ by the corresponding formula $\psi(x_1, \ldots, x_m)$ and one relativizes all quantifications to $\varphi(x)$. As a consequence, we note the following:

**Proposition 17.** *If $\mathcal{A}$ is MSO-interpretable in $\mathcal{B}$ and the MSO-theory of $\mathcal{B}$ is decidable, then so is the MSO-theory of $\mathcal{A}$.*

As a second example of MSO-interpretation, consider a pushdown automaton $\mathcal{A}$ with stack alphabet $\{1, \ldots, k\}$ and states $q_1, \ldots, q_m$. Let $G_{\mathcal{A}} = (V_{\mathcal{A}}, E_{\mathcal{A}})$ be its configuration graph. Choosing $n = \max\{k, m\}$, we can exhibit an MSO-interpretation of $G_{\mathcal{A}}$ in $T_n$: Just represent configuration $(q_j, i_1 \ldots i_r)$ by the vertex $i_r \ldots i_1 j$ of $T_n$. Note that then the $\mathcal{A}$-steps lead to local moves in $T_n$, from one $T_n$-vertex to another, e.g. in a push step from vertex $i_r \ldots i_1 j$ to a vertex $i_r \ldots i_1 i_0 j'$. These moves are easily definable in MSO, and so is reachability (from the initial vertex 11).

Hence we obtain the following result:

**Theorem 18.** (Muller, Schupp [17])
*The MSO-theory of a pushdown graph is decidable.*

A straightforward generalization of the proof yields corresponding statements for the prefix-recognizable graphs. The difference to the proof above is just a refinement of the formula expressing the one-step derivation relation between configurations. Instead of describing a single move from one word $u_0 w$ to another, say $v_0 w$, we have to describe all admissible moves from a word $uw$ to $vw$ where a rule $U \to V$ exists with $u \in U, v \in V$. This can be done by describing successful runs of the corresponding automata $\mathcal{A}_U, \mathcal{A}_V$ on the path segments from $uw$ to $w$ and from $vw$ to $w$, respectively. The description of a run of an automaton say with $k$ states $q_1, \ldots, q_k$ can be done in MSO logic by postulating $k$ subsets $X_1, \ldots, X_k$ of the considered path segments, the set $X_i$ containing those vertices where state $q_i$ is assumed.

**Theorem 19.** *A prefix-recognizable graph is MSO-interpretable in $T_2$; thus, the monadic second-order theory of a prefix-recognizable graph is decidable.*

For completeness we note an analogous result for automatic graphs and first-order logic:

**Theorem 20.** *An automatic graph is FO-interpretable in $T_2'$.*

*Proof.* In these notes we only give the basic idea of the proof. We concentrate on the case in which the alphabet $\Gamma$ for the automatic presentation of the given graph $G$ is just $\{0, 1\}$. We show how to give an FO-description of a regular set $V \subseteq \{0, 1\}^*$, recognized, say, by the automaton $\mathcal{A}$, in the structure $T_2'$. For this we shall develop a formula $\varphi(x)$ which expresses whether the vertex $x \in \{0, 1\}^*$ is accepted by $\mathcal{A}$. The letters of $x$ can be recovered from the course of the path leading to $x$; for example, the $i$-th letter of $x$ is 0 (resp. 1) iff on level $i - 1$

the path towards $x$ branches left (resp. right). The vertices on this path are definable with the prefix relation $\leq$ of $T_2'$. For simplicity, suppose that $\mathcal{A}$ has four states, so that a run on $x$ can be identified with a sequence of length $|x+1|$ of pairs of bits. Separating the two components we may code the sequence by two bit-words $z_1, z_2$ of length $|x+1|$. The existence of such words $z_1, z_2$ is directly expressible in first-order logic. Using the equal-length predicate of the signature, it is now easy to express the necessary conditions about $z_1, z_2$, namely that the coded sequence starts with the initial state of $\mathcal{A}$, that the step from a level to the next respects the transition relation of $\mathcal{A}$ (and the corresponding letter of $x$), and that the last coded state is accepting. □

It is remarkable that the last two theorems also hold in the converse direction: Any graph that is MSO-interpretable in $T_2$ is prefix-recognizable, and any graph that is FO-interpretable in $T_2'$ is automatic [2]. In the latter case, this yields a new way to establish the decidability of the FO-theory of an automatic graph: One just has to combine a decidability proof for the FO-theory of $T_2'$ with an FO-interpretation of the considered graph $G$ in $T_2'$.

### 5.2 Unfoldings

In the previous section we explained how to generate a model "within" a given one, via defining formulas. A more "expansive" way of model construction is the unfolding of a graph $(V, (E_a)_{a\in\Sigma}, (P_b)_{b\in\Sigma'})$ from a given vertex $v_0$, yielding a tree $T_G(v_0) = (V', (E_a')_{a\in\Sigma}, (P_b')_{b\in\Sigma'})$: $V'$ consists of the vertices $v_0a_1v_1\dots a_rv_r$ with $(v_{i-1}, v_i) \in E_{a_i}$, $E_a'$ contains the pairs $(v_0a_1v_1\dots a_rv_r, v_0a_1v_1\dots a_rv_rav)$ with $(v_r, v) \in E_a$, and $P_b'$ the vertices $v_0a_1v_1\dots a_rv_r$ with $v_r \in P_b$. The unfolding operation has no effect in bisimulation invariant logics, but is highly nontrivial for MSO-logic. Consider, for example, the singleton graph $G_0$ over $\{v_0\}$ with a 1-labelled and a 2-labelled edge from $v_0$ to $v_0$. Its unfolding is the infinite binary tree. While checking MSO-formulas over $G_0$ is trivial, this is a deep result for $T_2$. A powerful result going back to Muchnik 1985 says that unravelling preserves decidability of the MSO-theory.

**Theorem 21.** (Muchnik 1985, Courcelle and Walukiewicz [9])
*If the MSO-theory of $G$ is decidable and $v_0$ is an MSO-definable vertex of $G$, then the MSO-theory of $T_G(v_0)$ is decidable.*

The result holds also for a slightly more general construction ("tree iteration") which can also be applied to relational structures other than graphs. We cannot go into details here; a good presentation is given in [1].

## 6 Caucal's Hierarchy

MSO-interpretations and unfoldings are two operations which preserve decidability of MSO model-checking. Caucal [6] studied the structures generated by applying *both* operations, alternating between unfoldings and interpretations. He introduced the following hierarchy $(\mathcal{G}_n)$ of graphs, together with a hierarchy $(\mathcal{T}_n)$ of trees:

- $\mathcal{T}_0$ = the class of finite trees

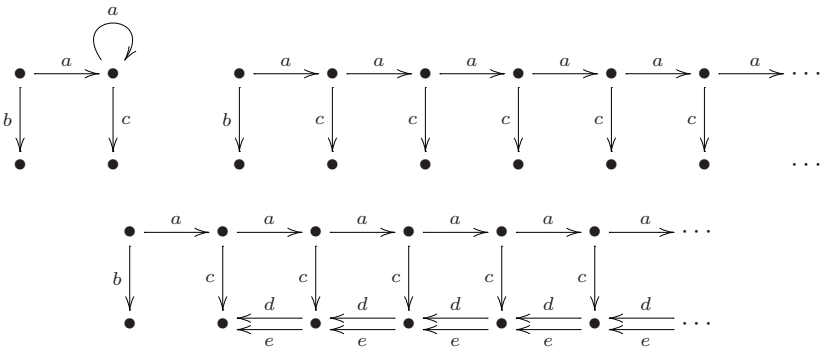



Figure 2: A graph, its unfolding, and a pushdown graph

- $\mathcal{G}_n$ = the class of graphs which are MSO-interpretable in a tree of $\mathcal{T}_n$
- $\mathcal{T}_{n+1}$ = the class of unfoldings of graphs in $G_n$

By the results of the preceding sections (and the fact that a finite structure has a decidable MSO-theory), each structure in the Caucal hierarchy has a decidable MSO-theory. By a hierarchy result of Damm on higher-order recursion schemes, the hierarchy is strictly increasing.

In Caucal's orginal paper [6], a different formalism of interpretation (via "inverse rational substitutions") is used instead of MSO-interpretations. We work with the latter to keep the presentation more uniform; the equivalence between the two approaches has been established by Carayol and Wöhrle [10].

Let us take a look at some structures which occur in this hierarchy (following [22]). It is clear that $\mathcal{G}_0$ is the class of finite graphs, while $\mathcal{T}_1$ contains the so-called regular trees (alternatively defined as the infinite trees which have only finitely many non-isomorphic subtrees). Figure 2 (upper half) shows a finite graph and its unfolding as a regular tree.

By an MSO-interpretation we can obtain the pushdown graph of Figure 2 in the class $\mathcal{G}_1$; the domain formula and the formulas defining $E_a, E_b, E_c$ are trivial, while

$$\psi_d(x,y) = \psi_e(x,y) = \quad \exists z \exists z' (E_a(z,z') \wedge E_c(z,y) \wedge E_c(z',x))$$

Let us apply the unfolding operation again, from the only vertex without incoming edges. We obtain the "algebraic tree" of Figure 3, belonging to $\mathcal{T}_2$ (for the moment one should ignore the dashed line).

As a next step, let us apply an MSO-interpretation to this tree which will produce a graph $(V, E, P)$ in the class $\mathcal{G}_2$ (where $E$ is the edge relation and $P$ a unary predicate). Referring to Figure 3, $V$ is the set of vertices which are located along the dashed line, $E$ contains the pairs which are successive vertices along the dashed line, and $P$ contains the special vertices drawn as non-filled circles. This structure is isomorphic to the structure $(\mathbb{N}, \mathrm{Succ}, P_2)$ with the successor relation Succ and predicate $P_2$ containing the powers of 2.

To prepare a corresponding MSO-interpretation, we use formulas such as $E_{d^*}(x,y)$ which expresses

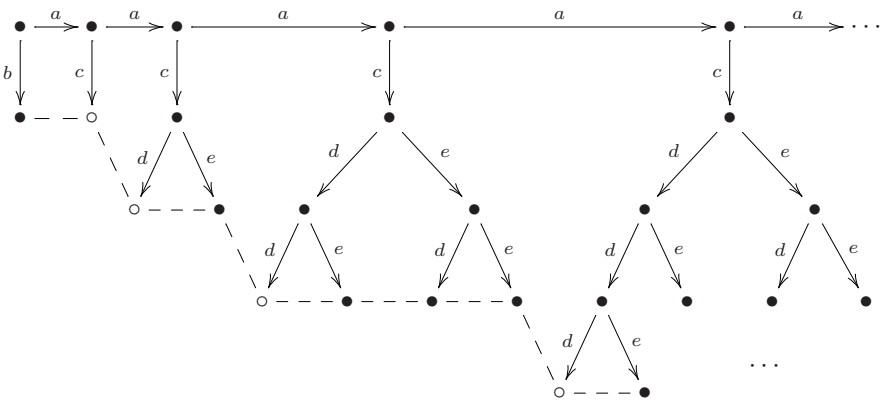



Figure 3: Unfolding of the pushdown graph of Figure 2

> "All sets which contain $x$ and are closed under $E_d$-successors contain $y$, and $y$ has no $E_d$-successor"

As domain formula we use

$$\varphi(x) = \exists z(E_b(z,x) \vee \exists y(E_c(z,y) \wedge E_{d^*}(y,x))).$$

The required edge relation $E$ is defined by $\psi(x,y) = \exists z \exists z'(\psi_1(x,y) \vee \psi_2(x,y) \vee \psi_3(x,y))$ where

- $\psi_1(x,y) \;=\; E_a(z,z') \wedge E_b(z,x) \wedge E_c(z',y)$
- $\psi_2(x,y) \;=\; E_a(z,z') \wedge E_{ce^*}(z,x) \wedge E_{cd^*}(z',y)$
- $\psi_3(x,y) \;=\; E_{de^*}(z,x) \wedge E_{ed^*}(z,y)$

Finally we define $P$ by the formula $\chi(x) = \exists z \exists z'(E_c(z,z') \wedge E_{d^*}(z',x))$.

We infer that the MSO-theory of $(\mathbb{N}, \mathrm{Succ}, P_2)$ is decidable, a result first proved by Elgot and Rabin in 1966 with a different approach.

Let us discuss another interesting structure of this kind, namely the structure $(\mathbb{N}, \mathrm{Succ}, \mathrm{Fac})$ where Fac is the set of factorial numbers. We start from a simpler pushdown graph than the one used above and consider its unfolding, which is the comb structure indicated by the thick arrows of the lower part of Figure 4.

We number the vertices of the horizontal line by $0, 1, 2 \ldots$ and call the vertices below them to be of "level 0", "level 1", "level 2" etc. Now we use the simple MSO-interpretation which takes all tree nodes as domain and introduces for $n \geq 0$ a new edge from any vertex of level $n+1$ to the first vertex of level $n$. This introduces the thin lines in Figure 4 as new edges (assumed to point backwards). It is easy to write down a defining MSO-formula. Note that the top vertex of each level plays a special role since it is the target of an edge labelled $b$, while the remaining ones are targets of edges labelled $c$.

Consider the tree obtained from this graph by unfolding. It has subtrees consisting of a single branch off level 0, 2 branches off level 1, $2 \cdot 3$ branches off level 2, and generally $(n+1)!$ branches off level $n$. Referring to the $c$-labelled edges these branches are arranged in a natural (and MSO-definable)

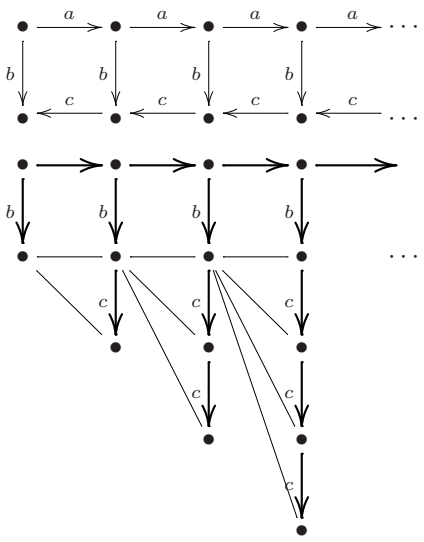
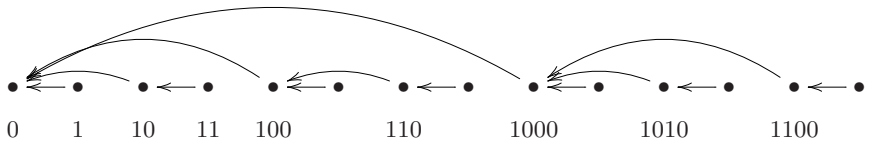



Figure 4: Preparing for the factorial predicate

order. To capture the structure $(\mathbb{N}, \mathrm{Succ}, \mathrm{Fac})$, we apply an interpretation which (for $n \geq 1$) cancels the branches starting at the $b$-edge target of level $n$ (and leaves only the branches off the targets of $c$-edges). As a result, $(n+1)! - n!$ branches off level $n$ remain for $n \geq 1$, while there is one branch off level 0. Numbering these remaining branches, the $n!$-th branch appears as first branch off level $n$. Note that we traverse this first branch off a given level by disallowing $c$-edges after the first $c$-edge. So a tree shape similar to Figure 3 emerges, now for the factorial predicate. Summing up, we have generated the structure $(\mathbb{N}, \mathrm{Succ}, \mathrm{Fac})$ as a graph in $\mathcal{G}_3$.

So far we have considered expansions of the successor structure of the natural numbers by unary predicates. We now treat the expansion by an interesting unary function (here identified with its graph, a binary relation). It is the *flip function*, which associates 0 to 0 and for each nonzero $n$ that number which arises from the binary expansion of $n$ by modifying the least significant 1-bit to 0. We give an illustration of the graph Flip of this function.




It is easy to see that the structure $(\mathbb{N}, \mathrm{Succ}, \mathrm{Flip})$ can be obtained from the algebraic tree of Figure 3 by an MSO-interpretation. A Flip-edge will connect vertex $u$ to the last leaf vertex $v$ which is reachable by a $d^*$-path from an ancestor of $u$; if such a path does not exist, an edge to the target of the $b$-edge (representing number 0) is taken.

The graphs in the Caucal hierarchy supply a vast universe of structures which has not been understood very well on the higher levels (say from level 3 onwards). Many interesting questions arise, for example the problem whether

one can compute the lowest level on which a given structure occurs. It is already known that the Caucal hierarchy does not fully exhaust the domain of graphs with a decidable MSO-theory.

# 7 Grid Structures and Tree Rewriting Graphs

The transition graphs of the Caucal hierarchy are still tightly connected with infinite trees – in fact, they can be generated for a given level $k$ from a single tree structure via MSO-interpretations. So these graphs are too restricted for many purposes of verification (excepting applications on the implementation of recursion).

A more flexible kind of model is generated when the idea of prefix-rewriting is generalized in a different direction, proceeding from word rewriting to tree rewriting (which we identify here with term rewriting). Instead of modifying the prefix of a word by applying a prefix-rewriting rule, we may rewrite a subtree of a given tree, precisely as it is done in ground term rewriting. So a rule $t \to t'$ allows to replace one occurrence of subtree $t$ by $t'$. To fix state properties, we refer to the well-known concept of regular sets of trees, defined by finite tree automata.

A *ground term rewriting graph (GTRG)* $G = (V, (E_a), (P_b))$ has a vertex set $V$ and subsets $P_b \subseteq V$ which are given by regular tree languages, and each edge relation $E_a$ is defined by a finite ground term rewriting system. Usually one restricts $V$ to contain only trees which are reachable from some regular set of initial trees via the edge relations $E_a$.

The concept is best introduced by an example. Consider the graph generated from the tree $f(c, d)$ by applying the rules $c \to g(c)$ and $d \to g(d)$ which produce the trees $f(g^i(c), g^j(d))$ in one-to-one correspondence with the elements $(i, j)$ of $\mathbb{N} \times \mathbb{N}$ (see Figure 5).

We thus see that the infinite $\mathbb{N} \times \mathbb{N}$-grid is a GTRG. Hence the MSO-theory of a GTRG can be undecidable.

However, for interesting properties the model-checking problem is still decidable. It is possible to combine the techniques of Section 2 (on automatic graphs) and of Section 3 (saturation algorithm), now applied over the domain of finite trees rather than words. Since the methodology does not change, we only state the result. (It should be noted, however, that in part (b) of the theorem the operator EFG ("there is a path with infinitely many vertices of a certain property") causes a lot of technical work which is far from straightforward.)

**Theorem 22.** (Dauchet, Tison [11], Löding [14])
*Over a ground tree rewriting graph, the model-checking problem is deciable for the logic FO(R) and for the branching-time logic with the following syntax:*
$T$ *(regular)* $\mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid EX_a\varphi \mid EF\varphi \mid EFG\varphi$.

It is possible – as for pushdown graphs – to generalize the rewriting rules without affecting the decidability results: Instead of allowing replacement of a single subtree by another one, one may use rules of the form $T \to T'$ for regular tree languages $T, T'$, meaning that an occurrence of subtree $t \in T$ can be replaced by any $t' \in T'$.

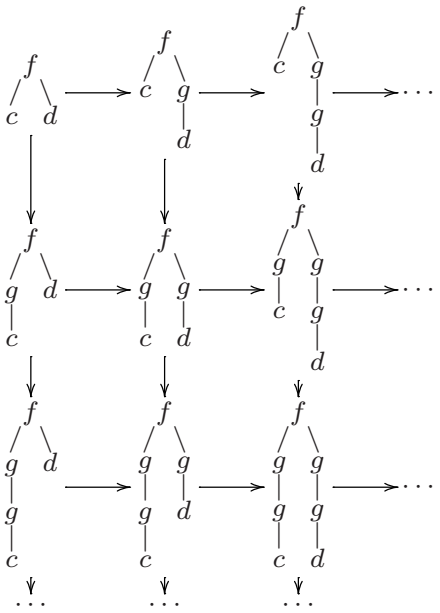


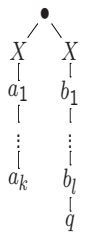
Figure 5: The grid as a ground tree rwriting graph

We now shall note that a slight extension of the logic of part (b) above leads to undecidability. This extension can best be explained in terms of branching time temporal operators in CTL-like notation: While the operators EF and EGF preserve decidability, this fails for the operator AF ("on each path there is a vertex with a certain property").

**Theorem 23.** (Löding [14])
*The following problem is undecidable:*

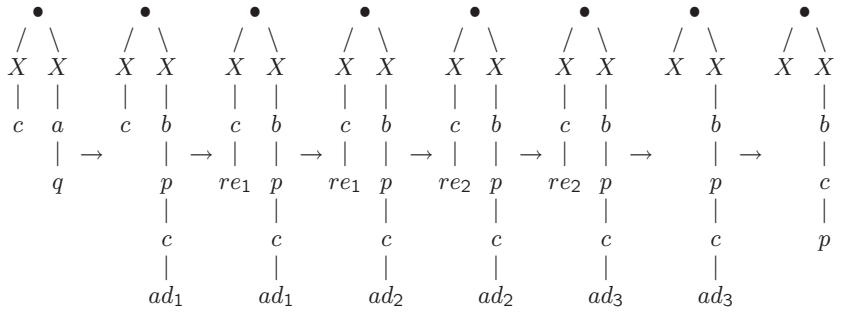
> *Given a ground tree rewriting graph $G$, a vertex $v$ and a regular set $T$ of vertices of $G$, does every path from $v$ through $G$ reach $T$?*

*Proof.* We give the main idea, which is typical for undecidability proofs where the essential logical operator to be exploited is universal (rather than existential, as needed in a direct coding of the halting problem). For a reduction of the halting problem for Turing machines we represent a Turing machine configuration $a_1 \dots a_k \; q \; b_l \dots b_1$ as a tree



Now we set up rewriting rules which simulate Turing machine computations. The main problem for a correct update of trees is the fact the left and right branch involve independent rewritings for a single Turing machine step. We enforce a correct match by the requirement that we implement a sequence of tree rewriting steps and require that all intermediate trees belong to a certain regular tree language $T$.

As an example consider the Turing machine instruction $q\ a\ b\ L\ p$. In the figure below a sequence of rewriting steps is presented which transform a tree code of the configuration $cqa$ into a tree coding the next configuration $pcb$. In the first two steps the new entry $c$ to the right-hand branch is guessed; the fact that $c$ was added is signalled by the marker $ad_1$. In the left-hand branch the symbol $re_1$ indicates that the symbol $c$ there is to be removed. The next steps introduce an increase of the $ad$- and $re$-indices (for technical reasons not explained here). A wrong guess (i.e., a mismatch between the symbol to be removed and the one to be added) can be described in terms of a regular tree language Err which contains the trees showing such a mismatch. The tree language Halt containing the codes of halting configurations is also regular.




The rewriting steps as indicated ensure that a Turing machine halts iff in the ground tree rewriting graph built up with the indicated rules, each path starting from the tree for the initial configuration meets either Err or Halt. □

# 8 Completing the Picture

## 8.1 Structural characterizations

We have discussed four basic types of infinite transition graphs: the rational, automatic, prefix-recognizable and the ground tree rewriting graphs. As specialization of the prefix-recognizable graphs we considered the pushdown graphs, and as a generalization the graphs of the Caucal hierarchy. For the definition of these structures, two approaches were pursued:

- the internal presentation in terms of automaton definable sets and relations of words, respectively trees,
- the external presentation by means of model transformations, starting from certain fundamental structures (in our case, the structures $T_2, T_2'$).

Some results at the end of Section 3 (on prefix-recognizable and automatic graphs) indicate that the two approaches can be merged. A match of internal and external presentations has also been achieved for the other types of graphs presented here: the rational graphs [15], the graphs of the Caucal hierarchy (see e.g. [10]), and the ground tree rewriting graphs [7]. The combination of both views (internal and external) is necessary for developing the algorithmic theory of infinite structures. Usually, the internal description is helpful in devising efficient algorithmic solutions, and the external presentation gives a convenient way of generating models without entering too much into "details of implementation". In classical mathematics, these two views are standard and complement each other. For example, if we specify a vector space by a basis (and the rule that linear combinations over the basis generate the elements of the space), we give an internal representation. If we take all linear maps over some vector space to construct a new vector space, we are building an external presentation.

The separation of these classes of graphs is not easy in general. We do not yet have many manageable "structural characterizations" which can be applied to graphs without reference to their presentations. A master example of such a characterization is a result of Muller and Schupp, concerning pushdown graphs.

Let $G = (V, (E_a)_{a\in\Sigma})$ be a graph of bounded degree and with designated "origin" vertex $v_0$. Let $V_n$ be the set of vertices whose distance to $v_0$ is at most $n$ (via paths formed from edges as well as reversed edges). Define $G_n$ to be the subgraph of $G$ induced by the vertex set $V \setminus V_n$, calling its vertices in $V_{n+1} \setminus V_n$ the "boundary vertices". The *ends* of $G$ are the connected components (using edges in both directions) of the graphs $G_n$ with $n \geq 0$. In [17], Muller and Schupp established a beautiful characterization of pushdown graphs in terms of the isomorphism types of their ends (where an end isomorphism is assumed to respect the vertex property of being a boundary vertex):

**Theorem 24.** (Muller, Schupp [17])
*A transition graph $G$ of bounded degree is a pushdown graph iff the number of distinct isomorphism types of its ends is finite.*

As an application, we see that the infinite $\mathbb{N} \times \mathbb{N}$-grid is not a pushdown graph. The ends $G_n$ exclude all vertices from the origin up to distance $n$. The vertices of distance precisely $n$ form a counter-diagonal from vertex $(0, n)$ to vertex $(n, 0)$. This counter-diagonal shows in particular that no two graphs $G_m, G_n$ for $m \neq n$ are isomorphic.

For the other graphs discussed in these lectures, such nice structural characterizations are still missing.

## 8.2 Recognized languages

Any infinite transition graph $G = (V, (E_a)_{a\in\Sigma}, I, F)$ with unary predicates $I, F \subseteq V$ (of "initial" and "final" vertices) may be used as an acceptor of words in the obvious way: A word is accepted if it occurs as a labelling of a path from a vertex in $I$ to a vertex in $F$.

If $V$ is finite, we obtain the usual model of nondeterministic finite automaton (here with several initial states), which yields the regular languages as corresponding class of languages. It is not surprising that the pushdown graphs (and: as it is easily verified, also the prefix-recognizable graphs) yield precisely the context-free languages:

**Theorem 25.** (Muller-Schupp [17], Caucal [5])
*A language $L$ is context-free iff $L$ is recognized by a pushdown graph (with regular sets of initial and final states) iff $L$ is recognized by a prefix-recognizable graph (with regular sets of initial and final states).*

This track of research was continued by surprising results regarding the rational and automatic graphs:

**Theorem 26.** (Morvan-Stirling [16], Rispal [19])
*A language $L$ is context-sensitive iff $L$ is recognized by an automatic graph (with regular sets of initial and final states) iff $L$ is recognized by a rational graph (with regular sets of initial and final states).*

The graphs of the Caucal hierarchy also correspond to known language classes which have been introduced in terms of "higher-order pushdown automata". For instance, the languages recognized by Caucal graphs of level 2 coincide with the "indexed languages" introduced in the 1960's by Aho. It is an open problem to provide a corresponding description for the languages recognized by ground tree rewriting graphs.

# 9 Retrospective and Outlook

These lectures gave an introduction to fundamental classes of infinite transition graphs, with some emphasis on the question which types of model-checking questions can be solved algorithmically.

Let us summarize the central ideas and proofs:

- the reduction of the Post Correspondence Problem and of the Halting Problem for Turing machines to simple questions about rational and automatic graphs,
- the decidability of the FO-theory of an automatic graph using an inductive construction of automata for definable relations,
- the reachability analysis for pushdown systems by the saturation algorithm,
- the method of interpretations, used to show that the MSO-theory of a pushdown graph is decidable, and the combination of interpretations and unfoldings for building up the Caucal hierarchy,
- the role of the infinite grid, as a structure with an undecidable MSO-theory but – as a ground tree rewriting graph – sharing still some decidability properties,
- the undecidability of properties over ground tree rewriting graphs that involve universal path quantification.

The subject of finitely presented infinite structures using automata theoretic ideas is fastly developing. Many tracks of research are open. We mention just a few.

- Study systematically *all* automatic / prefix recognizable presentations of a structure and investigate the efficiency of algorithms in dependence of the presentation.
- Consider more transformations for the generation of models, for example different kinds of products or variants of the unfolding operation (for example, instead of considering sequences to generate elements of the new model, one may consider sets).
- Proceed from graphs to more general structures (e.g., hypergraphs).
- Fill the gap between FO and MSO (by interesting intermediate logics), and similarly between automatic and pushdown graphs (by interesting intermediate graphs).
- Merge the theory of infinite transition systems with another source of infinity, namely arithmetical constraints (over $\mathbb{N}$ and $\mathbb{R}$).

# References


[1] D. Berwanger, A. Blumensath, The monadic theory of tree-like structures, *Automata, Logics, and Infinite Games*, Springer LNCS 2500 (2002), 285-302.

[2] A. Blumensath, E. Grädel, Automatic structures, in: *Proc. 15th LICS*, IEEE Comput. Soc. Press 2000, 51-62.

[3] J.R. Büchi, Weak second-order arithmetic and finite automata, *Z. Math. Logik Grundl. Math.* 6 (1960), 66-92.

[4] J.R. Büchi, Regular canonical systems, *Archiv Math. Grundlagenforschung* 6 (1964), 91-111.

[5] D. Caucal, On infinite transition graphs having a decidable monadic theory, in: *Proc. 23rd ICALP* (F. Meyer auf der Heide, B. Monien, Eds.), Springer LNCS 1099 (1996), 194-205 [Full version in: *Theor. Comput. Sci.* 290 (2003), 79-115].

[6] D. Caucal, On infinite graphs having a decidable monadic theory, in: *Proc. 27th MFCS* (K. Diks, W. Rytter, Eds.), Springer LNCS 2420 (2002), 165-176.

[7] Th. Colcombet, On families of graphs having a decidable first order theory with reachability, in: *Proc. 29th ICALP*, Springer LNCS 2380 (2002), 98-109.

[8] J.L. Coquidé, M. Dauchet, R. Gilleron, S. Vágvölgyi, Bottom-up tree pushdown automata: Classification and connection with rewrite systems, *Theor. Comput. Sci.* 127 (1994), 69-98.

[9] B. Courcelle, I. Walukiewicz, Monadic second-order logic, graph coverings and unfoldings of transition systems, *Ann. Pure Appl. Logic* 92 (1998), 51-65.

[10] A. Carayol, S. Wöhrle, The Caucal hierarchy of infinite graphs in terms of logic and higher-order pushdown automata, in: *Proc. 23rd FSTTCS*, Springer LNCS 2914 (2003), 112-123.

[11] M. Dauchet, S. Tison, The theory of ground rewrite systems is decidable, *Proc. 5th IEEE Symp. on Logic in Computer Science*, LICS'90, IEEE Computer Science Press 1990, 242-248.

[12] J. Esparza, D. Hansel, P. Rossmanith, S. Schwoon, Efficient algorithms for model-checking pushdown systems, *Proc. CAV 2000*, Springer LNCS 1855 (2000), 232-247.

[13] J.E. Hopcroft,. J.D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, Reading, Mass. 1979.

[14] C. Löding, Model-checking infinite systems generated by ground tree rewriting, *Proc. FoSSaCS 2002*, Springer LNCS 2303 (2002), 280-294.

[15] C. Morvan, On rational graphs, in *Proc. FoSSaCS 2000*, Springer LNCS 1784 (2000), 252-261.

[16] C. Morvan, C. Stirling, Rational graphs trace context-sensitive languages, *Proc. MFCS 2001*, Springer LNCS 2136 (2001), 548-559.

[17] D. Muller, P. Schupp, The theory of ends, pushdown automata, and second-order logic, *Theor. Comput. Sci.* 37 (1985), 51-75.

[18] M.O. Rabin, Decidability of second-order theories and automata on infinite trees, *Trans. Amer. Math. Soc.* 141 (1969), 1-35.

[19] C. Rispal, The synchronized graphs trace the context-sensitive languages, *Electr. Notes Theor. Comput. Sci.* 68 (2002).

[20] W. Thomas, Languages, Automata, and Logic, in: *Handbook of Formal Languages* (G. Rozenberg, A. Salomaa, eds.), Vol 3, Springer-Verlag 1997, pp. 389-455.

[21] W. Thomas, A short introduction to infinite automata, in: *Proc. 5th Conf. on Developments in Language Theory* Springer LNCS 2295, 130-144

[22] W. Thomas, Constructing infinite graphs with a decidable MSO-theory, in: *Proc. 28th MFCS*, Springer LNCS 2747 (2003), 113-124.