A Short Introduction to Infinite Automata

Wolfgang Thomas

Lehrstuhl für Informatik VII, RWTH Aachen, 52056 Aachen, Germany
thomas@informatik.rwth-aachen.de

Abstract. Infinite automata are of interest not only in the verification of systems with infinite state spaces, but also as a natural (and so far underdeveloped) framework for the study of formal languages. In this survey, we discuss some basic types of infinite automata, which are based on the so-called prefix-recognizable, synchronized rational, and rational transition graphs, respectively. We present characterizations of these transition graphs (due to Muller/Schupp and to Caucal and students), mention results on their power to recognize languages, and discuss the status of central algorithmic problems (like reachability of given states, or decidability of the first-order theory).

1 Historical background

In the early days of computer science, the first purpose of "finite automata" was to serve as an abstract model of circuits. In this context, a "state" is given by an association of boolean values to the latches of a circuit, i.e., by a bit vector $(b_1, \ldots, b_n)$ if there are n latches. A set of states is then given by a boolean function in n variables and can be defined by a boolean formula $\varphi(x_1, \ldots, x_n)$, and a transition relation (say via a fixed input) by a boolean formula $\psi(x_1, \ldots, x_n, x'_1, \ldots, x'_n)$ of $2n$ boolean variables. Only later in the emergence of computer science, when programming languages had entered the stage and the purpose of string processing was associated with finite automata, a more abstract view of automata became dominant: Now states and inputs were collected in abstract finite sets, often denoted Q and Σ, the transition relation being simply a subset of $Q \times \Sigma \times Q$.

In the framework of model checking (for the automatic verification of state-based programs, cf. [CGP99]), a reverse development took place. At first, the abstract view was pursued: A system, such as a reactive program or a protocol, is modelled by an annotated transition graph, whose vertices are the system's states. The task of model checking a temporal property (given by a temporal logic formula) is to compute the set of those vertices where the given formula is satisfied. However, as noted by McMillan [McM93], the efficiency of model checking is improved considerably when the so-called "symbolic approach" is adopted. This approach involves a return "back to the roots" of automata theory: States of transition graphs are bit vectors, and sets of states, as well as transition relations, are given by boolean functions. An essential point in symbolic model-checking to achieve efficiency is a particular representation of boolean functions, namely by the "ordered binary decision diagrams" (OBDD's). These OBDD's are a form of minimized acyclic finite automata accepting sets of 0-1-words of bounded length.

At this point, and aiming at the treatment of infinite systems, it is natural to cancel the constraint of bounded length. When admitting words of arbitrary finite length as representations of individual states, the OBDD's should be replaced by usual finite automata, thereby dropping the requirement of acyclicity. State properties will thus be captured by regular languages. For the specification of transition relations, one has to invoke automata theoretic definitions of word relations. There is a large reservoir of options here, by considering finite-state tranducers or word rewriting systems of different kinds. Presently, a lot of research is devoted to the study of infinite transition systems using these ideas, in an attempt to generalize the methodology of model checking to systems with infinite state spaces. (The reader can trace this research fairly well by consulting the LNCS-Proceedings of the annual conferences *CAV (Computer-Aided Verification)* and *TACAS (Tools and Algorithms in the Construction and Analysis of Systems)*). But also from the language theoretical point of view, infinite transition systems are interesting since they provide an alternative to standard models of automata for the definition of non-regular languages.

In all these investigations, the classical finite automata are still present: They enter as a tool to define the infinite structures (infinite state spaces, infinite transition graphs). Thus their dynamics is conceptually no more associated with the "progress in time" (as in the original circuit model), but rather with the "exploration of static data" (when the scanned words are considered as representations of individual states of an infinite system). In this role, finite automata will also be useful in the development of an infinitary but algorithmically oriented model theory.

In this paper we survey some basic classes of infinite transition systems, giving characterization results, remarks on their power as acceptors of languages, and discussing the status of algorithmic problems. Our aim is to help the reader to enter this fastly developing field and to give some references where more results and also detailed proofs can be accessed. We do not treat in any depth the many connections with process theory, where different means of specifying infinite transition systems are employed and where the semantics is based on bisimulation equivalence rather than on accepted languages. (The reader may start that subject with the survey [BE97], where the essential concepts and algorithmic results are presented.)

2 Some basic classes of infinite transition graphs

We consider edge labelled transition graphs of the form $G = (V, (E_a)_{a \in A})$, where V is the set of vertices (sometimes considered as "states"), A the alphabet of edge labels, and E_a the set of a-labelled edges. Such a structure is extended to an automaton by the specification of initial and final vertices. In all cases considered below, where V is given as a regular language over some auxiliary alphabet Γ, the sets I, F of initial and final states will be assumed to be regular languages over Γ. For the classification of infinite automata it will thus suffice to look at their transition structure, i.e. at (the presentation of) the graph $(V, (E_a)_{a \in A})$.

2.1 Pushdown graphs and prefix-recognizable graphs

A graph $G = (V, (E_a)_{a \in A})$ is called *pushdown graph* if it is the transition graph of the reachable global states of some ϵ-free pushdown automaton. We consider pushdown automata in the format $\mathcal{P} = (Q, A, \Gamma, q_0, Z_0, \Delta)$, where Q is the finite set of control states, A the input alphabet, Γ the stack alphabet, q_0 the initial control state, $Z_0 \in \Gamma$ the initial stack symbol, and $\Delta \subseteq Q \times A \times \Gamma \times \Gamma^* \times Q$ the transition relation. As usual, a global state of the automaton is given by a control state and a stack content, i.e., by a word from $Q\Gamma^*$. The graph $G = (V, (E_a)_{a \in A})$ is now specified as follows:

- V is the set of configurations from $Q\Gamma^*$ which are reachable (via finitely many applications of transitions of Δ) from the initial global state $q_0 Z_0$.
- E_a is the set of all pairs $(p\gamma w, qvw)$ from V for which there is a transition (p, a, γ, v, q) in Δ.

Example 1. Consider the pushdown automaton over $A = \{a, b, c\}$ with the single state q_0, stack alphabet $\Gamma = \{X, Y\}$, and initial stack letter X, which recognizes the language $\{a^i b c^i \mid i \geq 0\}$ in the standard way (accepting by empty stack): By the transition (q_0, a, X, XY, q_0), it generates, starting from X, the stack content XY^i after reading i letters a, then upon reading b it cancels the top X from the stack by the transition $(q_0, b, X, \epsilon, q_0)$, and finally for an input letter c it deletes the top letter Y from the stack, using the transition $(q_0, c, Y, \epsilon, q_0)$. The infinite transition graph looks as follows:

$$\begin{tikzcd}
q_0X \arrow[r, "a"] & q_0XY \arrow[r, "a"] \arrow[d, "b"] & q_0XYY \arrow[r, "a"] \arrow[d, "b"] & q_0XYYY \arrow[r, "a"] \arrow[d, "b"] & \cdots \\
q_0 & q_0Y \arrow[l, "c"'] & q_0YY \arrow[l, "c"'] & q_0YYY \arrow[l, "c"'] & \cdots \arrow[l, "c"']
\end{tikzcd}$$

This view of pushdown automata is well-known; see e.g. [KS86, p.242].

By their definition, pushdown graphs are of bounded in- and out-degree. A more general class of graphs, which includes the case of vertices of infinite degree, has been introduced by Caucal [Cau96]. These graphs are introduced in terms of prefix rewriting systems in which "control states" (as they occur in pushdown automata) do no more enter 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 $u_1 w$ via letter a to the word $u_2 w$. The feature of infinite degree is introduced by allowing generalized rewriting rules of the form $U_1 \rightarrow_a U_2$ 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 $G = (V, (E_a)_{a \in A})$ is called *prefix-recognizable* if for some finite system $\mathcal{S}$ of such generalized prefix rewriting rules $U_1 \rightarrow_a U_2$ over an alphabet Γ, we have

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

Example 2. Consider the prefix rewriting system over $\Gamma = \{X\}$ with the rules $\epsilon \rightarrow_a X$ and $X^+ \rightarrow_b \epsilon$ (here we identify a singleton set with its element). The corresponding graph $G = (V, E_a, E_b)$ over $V = \{X\}^*$ has the edge relations $E_a = \{(X^i, X^{i+1}) \mid i \geq 0\}$ and $E_b = \{(X^i, X^j) \mid i > j\}$; so the graph G is isomorphic to the structure of the natural numbers with the successor relation and the $>$-relation.

It is known that the pushdown graphs coincide with the prefix-recognizable graphs of finite in- and out-degree. A third class of graphs, located strictly between the pushdown graphs and the prefix-recognizable graphs, are the HR-equational graphs (sometimes just called equational or regular graphs), introduced by Courcelle [Cou89]. These graphs are generated by deterministic hyperedge replacement graph-grammars. In [CK01] a restriction of the prefix-recognizable graphs is presented which characterizes the HR-equational graphs.

2.2 Rational and synchronized rational graphs

In many practical applications (of model checking), the restriction to prefix rewriting is too severe to allow the specification of interesting infinite-state systems. Exceptions are structures derived from recursive programs; in this case, prefix rewriting matches the structure of the state space to be described (see e.g. [ES01]).

From the classical literature on transducers ([EM65], [Eil74], [Ber79]), more powerful types of relations are known, in particular the rational and the synchronized rational relations. For simplicity we consider binary relations only, where moreover the alphabets of the two components coincide.

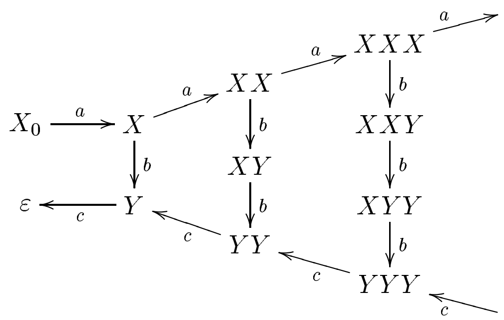
A relation $R \subseteq A^* \times A^*$ is *rational* if it can be defined by a regular expression starting from the atomic expressions $\emptyset$ (denoting the empty relation) and (u, v) for words u, v (denoting the relation $\{(u, v)\}$), by means of the operations union, concatenation (applied componentwise), and iteration of concatenation (Kleene star). An alternative characterization of these relations involves a model of non-deterministic automata which work one-way, but asynchronously, on the two components of an input $(w_1, w_2) \in A^* \times A^*$. To illustrate this model with a simple example, consider the suffix relation $\{(w_1, w_2) \mid w_1 \text{ is a suffix of } w_2\}$: A corresponding automaton 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 $G = (V, (E_a)_{a \in A})$ where V is a regular set of words over an auxiliarly alphabet Γ and where each E_a is a rational relation.

Example 3. ([Mor00]) Given an instance $(\overline{u}, \overline{v}) = ((u_1, \ldots, u_m), (v_1, \ldots, v_m))$ of the Post Correspondence Problem (PCP) over an alphabet Γ we specify a rational graph $G_{(\overline{u},\overline{v})} = (V, E)$. The vertex set V is Γ^*. 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).

In *synchronized rational relations* a more restricted processing of an input (w_1, w_2) is required: the two components are scanned strictly letter by letter. Thus one can assume that the automaton reads letters from $A \times A$ if $w_1, w_2 \in A^*$. 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. In this way a relation $R \subseteq A^* \times A^*$ induces a language L_R over the alphabet $(A \times A) \cup (A \times (A \cup \{\$\})) \cup ((A \cup \{\$\}) \times A)$. The relation R is called *synchronized rational* if the associated language L_R is regular. From this definition it is immediately clear that the synchronized rational 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 apply in the context of rational relations. For a comprehensive study of synchronized rational relations we recommend [FS93]. A graph $(V, (E_a)_{a \in A})$ is called synchronized rational if each edge relation E_a is synchronized rational.

Example 4. 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 $X^i Y^j$ 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 following synchronized rational graph:



If X_0 is taken as initial and ϵ as final state, the resulting automaton recognizes the context-sensitive language consisting of the words $a^i b^i c^i$ for $i \geq 1$.

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 a synchronized rational graph: 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 synchronized rational relations.

In the literature, the synchronized rational relations appear under several names; in [EM65] they are called regular, in [Büc60] sequential, and in the context of combinatorial group theory [EC*92] they are named automatic. Here one speaks of automatic groups (see [EC*92]), more generally of "automatic structures" (see [BG00]).

We give another example which illustrates the power of synchronized rational relations:

Example 6. Given a Turing machine M with state set Q and tape alphabet Γ, 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; from this it is easy to see that the relation E_M of word pairs which consist of successive M-configurations is synchronized rational. So the configuration graph $G_M = (V_M, E_M)$ is synchronized rational.

We now compare the four classes of graphs introduced so far.

Theorem 1. *The pushdown graphs, prefix-recognizable graphs, synchronized rational graphs, and rational graphs constitute, in this order, a strictly increasing inclusion chain of graph classes.*

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 synchronized rational ones. To verify that a prefix-recognizable graph is synchronized rational, we first proceed to an isomorphic graph which results by 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 synchronized rational: Consider a word pair (wu_1, wu_2) which results from the application of a suffix rewriting rule $U_1 \rightarrow_a U_2$, 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_2 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 2). To distinguish the other graph classes, one may use logical decidability results as presented in the next section. It will be shown there that the monadic second-order theory of a prefix-recognizable graph is decidable, which fails for the synchronized rational graphs, and that the first-order theory of a synchronized rational graph is decidable, which fails for the rational graphs.

2.3 External characterizations

The introduction of classes of transition graphs as developed above rests on an explicit representation of vertices and edges in terms of words, respectively pairs of words. There is an alternative way to characterize transition systems by structural properties. In the terminology of Caucal, this amounts to proceed from an "internal" to an "external" presentation. We present here three of these characterization results, due to Muller and Schupp (in their pioneering paper [MS85]) and to Caucal and his students ([Cau96], [Mor00], [Ris01]).

Let $G = (V, (E_a)_{a \in A})$ 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 [MS85], Muller and Schupp have 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 2. *A transition graph G of bounded degree is a pushdown graph iff the number of distinct isomorphism types of its ends is finite.*

A simple example is the full binary tree, which generates only a single isomorphism type by its ends (namely, the type of the full binary tree itself). In the subsequent characterizations, different structural properties of graphs are defined by different kinds of "reductions" to the structure of the binary tree. We consider the binary tree as the structure $T_2 = (\{0,1\}^*, E_0, E_1, \overline{E_0}, \overline{E_1})$, where E_0 (E_1) is the left (right) successor relation, while the edges of $\overline{E_0}, \overline{E_1}$ are the reverse ones of E_0, E_1, respectively. A finite path through the tree (which may move upwards as well as downwards, starting from some given vertex) can thus be described by a word over the alphabet $\{0, 1, \overline{0}, \overline{1}\}$. For example, starting from the vertex 10011, the word $\overline{1}\ \overline{1}\ 0\ 1$ describes the path leading up to 100 and then down to 10001.

We now can state Caucal's Theorem ([Cau96]) which says that the prefix-recognizable graphs can be obtained from the binary tree by a kind of regular modification, the edge sets E_a being described by regular sets of such path descriptions. (Thus prefix-recognizable graphs are rather "tree-like", which explains their limited use in modelling general transition systems.)

Theorem 3. *A transition graph is prefix-recognizable iff it can be obtained from the binary tree by an inverse regular substitution followed by a regular restriction.*

Let us explain the condition on the right-hand side, i.e. how it defines a graph $G = (V, (E_a)_{a \in A})$: By the regular restriction (to a regular set $V \subseteq \{0,1\}^*$ of vertices of the binary tree), we are provided with the vertex set. The regular substitution maps each letter $a \in A$ to a regular language $L_a \subseteq \{0, 1, \overline{0}, \overline{1}\}^*$. The inverse application of this substitution yields the edges as follows: There is an a-edge from u to v iff there is a word $w \in L_a$ describing a path from u to v.

Another way of stating the theorem is to say that the prefix-recognizable graphs are those which can be obtained as "regular interpretations" in the binary tree. As shown in [Bar97] and [Blu00], this coincides with a logical reduction, namely via monadic second-order definable interpretations.

A slight extension of the notion of regular substitution in the above theorem yields a corresponding characterization of the rational graphs, established by Morvan [Mor00]. Here one considers "special linear languages" over the alphabet $\{0, 1, \overline{0}, \overline{1}\}$, which are defined to be generated by linear grammars with productions $A \to \overline{u}Bv$, $A \to \epsilon$, assuming that $\overline{u} \in \{\overline{0}, \overline{1}\}^*$ and $v \in \{0, 1\}^*$. By a "special linear substitution", each letter a is associated to a special linear language $L_a \subseteq \{0, 1, \overline{0}, \overline{1}\}^*$.

Theorem 4. *A transition graph is rational iff it can be obtained from the binary tree by an inverse special linear substitution followed by a regular restriction.*

Let us illustrate the theorem for the example of the $(\mathbb{N} \times \mathbb{N})$-grid. As regular representation of the vertex set we take the "comb-structure" 0^*1^* consisting of the leftmost branch LB of the binary tree (of the vertices in 0^*) together with all rightmost branches starting on LB. Then the a-edges simply lead from a vertex of this comb-structure to its right successor, and hence we can set $L_a = \{1\}$. The unique b-edge from a vertex $u \in 0^*1^*$, say 0^m1^n, of the binary tree is obtained by going back from u to LB (via n $\overline{1}$-edges), proceeding from the vertex 0^m reached on LB one step to the left, and from there proceeding down the rightmost path again n times. So we have $L_b = \{\overline{1}^i01^i \mid i \geq 0\}$, which is a special linear language.

As shown by Rispal [Ris01], a suitable restriction of special linear substitutions yields also a characterization of the synchronized rational graphs.

2.4 Recognized languages

Since the pushdown graphs are derived from pushdown automata it may not seem surprising that their use as automata (with regular sets of initial and final states) allows the recognition of precisely the context-free languages. For the proof, the reader should consult the fundamental paper [MS85]. Caucal [Cau96] has shown that the expressive power (regarding recognition of languages) of the automata does not increase when proceeding to prefix-recognizable graphs:

Theorem 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).*

Recently, this track of research was continued by surprising results regarding the rational and the synchronized rational graphs. Morvan and Stirling prove in [MoS01] that the rational graphs allow to recognize precisely the context-sensitive languages, and Rispal [Ris01] shows that the synchronized rational graphs reach the same expressive power:

Theorem 6. *A language L is context-sensitive iff L is recognized by a synchronized rational 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).*

We give some comments on the claim that rational transition graphs can only recognize context-sensitive languages. For this, a Turing machine has to check (with linear space in the length of the input) whether in the given rational transition graph G there is an path from an initial to a final vertex labelled by the input word. The Turing machine guesses this path nondeterministically, reusing the same tape segment for the different words that constitute the path through G. The main problem is the fact that the length of the individual words visited on the path through G may increase exponentially in the length of the path (which is the length of the input word). This may happen, for example, if the transducer defining the edge relation of G has transitions that involve rewriting a letter X by XX; this would allow to double the length of the word in each transition by an edge of G. The solution, which cannot be described in detail here, is to check the correctness of the word rewriting online while traversing the two considered words simultaneously; this can be done in constant space.

The converse direction is still harder; in [MoS01] strong normal forms of context-sensitive grammars due to Penttonen ([Pen74]) are applied.

It seems that Theorem 6 points to a deeper significance of the class of context-sensitive languages (which so far played its main role as a rather special space complexity class). In particular, it turns out that the context-sensitive languages can be recognized in a one-way read-only mode by (rational or synchronized rational) infinite automata, in contrast to the standard description by two-way Turing machines with read- and write-moves. It remains to be analyzed whether nontrivial facts on the context-sensitive languages (like the closure under complementation) have new (and maybe even simpler?) proofs in the framework of infinite automata. An important open question asks whether rational (or synchronized rational) graphs with a deterministic transition structure suffice for the recognition of the context-sensitive languages.

Caucal [Cau01] has extended the two theorems above, giving a class of infinite automata that accept precisely the recursively enumerable languages.

In process theory, one compares the expressive power of transition systems using sharper distinctions than by their ability to recognize languages: In this context, the semantics of transition systems (with designated initial state) is given by their bisimulation classes. With respect to bisimulation equivalence, the pushdown graphs are no more equivalent to the prefix-recognizable graphs (there are prefix-recognizable graphs which are not bisimilar to any pusdown graph); a corresponding proper inclusion with respect to bisimulation equivalence applies to the synchronized rational graphs in comparison with the rational graphs.

3 Algorithmic Problems

The infinite transition graphs considered above are all finitely presented, namely by finite automata defining their state properties, and by finite word rewriting systems or transducers defining their transition relations. Thus, decision problems concerning these graphs have a clear algorithmic meaning: as input one takes the finite automata which serve as presentations of transition graphs.

In this section we touch only a few (however central) decision problems on infinite transition systems. First we shall discuss the "reachability problem" which asks for the existence of paths with given start and end vertices. Secondly, we treat the "model-checking problem" regarding first-order and monadic second-order formulas, or, in a more classical terminology, the question of decidability of the first-order and the monadic second-order theory of a transition graph. (Here we assume that the reader is familiar with the fundamentals of first-order and monadic second-order logic, see e.g. [Tho97].)

3.1 Problems on pushdown graphs and prefix-recognizable graphs

A main result of Büchi's paper [Büc65] on regular canonical systems (i.e., prefix rewriting systems) is the following theorem: The words which are generated from a fixed word by a 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 set T of vertices as "target set", and the problem is to find those vertices (i.e., words) from which a vertex of T is reachable by a finite number of pushdown transition steps. More precisely, one deals with regular target sets. Given a pushdown graph G, the aim is to find from a finite automaton recognizing the vertex set T another finite automaton recognizing the set $pre^*(T)$ of those vertices from which there is a path through G to T. (The notation indicates that we are dealing with the transitive closure of the edge predecessor relation, starting with T.)

Theorem 7. *Given a pushdown graph $G = (V, E)$ (with $V \subseteq Q\Gamma^*$, where Q is the set of control states and Γ the stack alphabet of the underlying pushdown automaton), and given a finite automaton recognizing a set $T \subseteq Q\Gamma^*$, one can compute a finite automaton recognizing $pre^*(T)$. In particular, it can be decided for any vertex v whether it belongs to $pre^*(T)$ or not.*

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. It is convenient to work with Q as the set of initial states of $\mathcal{A}$; so a stack content qw of the pushdown automaton is scanned by $\mathcal{A}$ starting from state q and then processing the letters of w. 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 also the stack content $p\gamma w$ shoud 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 γ to r. (For a detailed proof and complexity analysis of the construction see [EH*00]).

Let us embed the reachability problem into a more general question, that of model-checking monadic second-order properties. For this, we observe that reachability of a set T from a given vertex v can be expressed in the language of monadic second-order logic (MSO-logic). We indicate a corresponding formula $\varphi(x, Y)$ where x is a first-order variable (to be interpreted by single vertices v) and Y is a second-order variable (to be interpreted by target sets T). The formula $\varphi(x, Y)$ just has to express that x is in the smallest set containing Y and closed under taking "predecessor vertices". To express this more precisely, we assume a single edge relation E in the graph in question (which can be considered as the union of all edge relations E_a as considered before). The desired formula $\varphi(x, Y)$ says the following: "x belongs to each set Z containing Y, such that for any two vertices z, z' with $z' \in Z$ and $(z, z') \in E$ we have also $z \in Z$ ".

The MSO-definability of the reachability relation hints at a much more powerful decidability result than the previous theorem, which moreover extends to the class of prefix-recognizable graphs (cf. [Cau96]):

Theorem 8. *The monadic second-order theory of a prefix-recognizable graph is decidable.*

The proof works by a reduction to Rabin's Tree Theorem [Rab69] which states that the monadic second-order theory of the binary tree is decidable. The connection is provided by Theorem 3 above, which gives us a regular interpretation of any given prefix-recognizable graph G in the binary tree T_2. The interpretation of a graph G in T_2 by a regular substitution and a regular restriction is easily converted into an MSO-interpretation; this in turn allows to tranform any MSO-sentence φ into a new MSO-sentence φ' such that $G \models \varphi$ iff $T_2 \models \varphi'$. This provides a decision procedure for the MSO-theory of G, invoking the decidability of the MSO-theory of T_2.

3.2 Problems on rational and synchronized rational graphs

The decidability results above fail when we consider the much more extended class of synchronized rational graphs instead of the pushdown graphs or prefix-recognizable graphs.

Let us first consider the reachability problem. Here we use the fact that for a universal Turing machine M, say with a normalized halting configuration c_0, it is undecidable whether from a given start configuration c we can reach c_0. Hence, we know that for the synchronized rational graph G_M as introduced in Example 6 above, the reachability problem is undecidable.

Theorem 9. *There is a synchronized rational graph G such that the reachability problem over G (given two vertices, decide whether there is path from the first to the second) is undecidable.*

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

Let us also look at a more ambitious problem than reachability: decidability of the monadic second-order theory of a given graph. Here we get undecidability for synchronized rational graphs with much simpler transition structure than that of the graphs G_M of the previous theorem, for example for the infinite two-dimensional grid (shown to be synchronized rational in Example 5). Note that the reachability problem over the grid is decidable.

Theorem 10. *The monadic second-order theory of the infinite two-dimensional grid G_2 is undecidable.*

The idea is to code the work 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 transitions 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 Turing machine computation is thus represented by a sequence of colored rows, 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 (say with m states and n tape symbols) 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 φ_M such that

$$M \text{ halts on the empty tape iff } G_2 \models \varphi_M .$$

When we restrict the logical framework to first-order logic (FO-logic), then we are back to decidability:

Theorem 11. *The first-order theory of a synchronized rational graph is decidable.*

The proof of this theorem is based on an idea which was first proposed by Büchi in [Büc60], as a method to show the decidability of Presburger arithmetic (the first-order theory of the structure $(\mathbb{N}, +)$). In both cases, Presburger arithmetic and the FO-theory of a synchronized rational graph, one works with representations of the structure's elements (numbers, respectively vertices) by words. For

Presburger arithmetic one uses the representation by reversed binary numbers. An n-ary relation R is then described by a set of n-tuples of words. As in the definition of synchronized rational graphs, one codes R by a language L_R: For this, any n-tuple $(w_1, \dots, w_n)$ of words, say over the alphabet Γ, is presented as a single word over the alphabet $(\Gamma \cup \{\$\})^n$, reading the words $w_1, \dots, w_n$ simultaneously letter by letter and using the symbol \$ for filling up words at the end to achieve equal length. The words resulting from the n-tuples in R form the language L_R. The key step for the decidability proof is the following lemma: *If R is a first-order definable relation in a synchronized rational graph, then the language L_R is regular.* Given a synchronized graph $G = (V, (E_a)_{a \in A})$, this is easily proved by induction over the construction of first-order formulas: The case of relations defined by the atomic formulas $x = y$ and $(x, y) \in E_a$ is trivial by the assumption that G is synchronized rational. For the induction steps, regarding the connectives $\neg$, $\vee$, and the existential first-order quantifier, one uses the closure of the automata defined languages L_R under boolean operations and projection.

Let us finally show that the decidability of the first-order theory does not extend to rational graphs. We proceed in two steps. The first deals with a weaker result, concerning a uniform statement for the class of rational graphs, the second provides a single rational graph with an undecidable first-order theory.

Theorem 12. *([Mor00]) There is no algorithm which, given a presentation of a rational graph G and a first-order sentence φ, decides whether $G \models \varphi$.*

The proof is easy using the graphs $G_{(\overline{u},\overline{v})}$ as introduced in Example 3 above, where $(\overline{u}, \overline{v})$ is an instance of the Post Correspondence Problem. As explained in Example 3, $(\overline{u}, \overline{v})$ has a solution iff $G_{(\overline{u},\overline{v})} \models \exists x\ (x, x) \in E$. Thus a uniform algorithm to decide the first-order theory of any given rational graph would solve the Post Correspondence Problem.

In order to obtain a single rational graph with an undecidable first-order theory, we refine the construction. As for Theorem 9, we use a universal Turing machine M and the encoding of its undecidable halting problem (for different input words x) into a family of instances of the Post Correspondence Problem. For simplicity of exposition, we refer here to the standard construction of the undecidability of the PCP as one finds it in textbooks (see [HU79, Section 8.5]): A Turing machine M with input word x is converted into a PCP-instance $((u_1, \dots, u_m), (v_1, \dots, v_m))$ 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 $x = a_1 \dots a_n$, then u_1 is set to be the initial configuration word $c(x) := \# q_0 a_1 \dots a_n$ of M; furthermore we have always $v_1 = \#$, and $u_2, \dots, u_m, v_2, \dots, v_m$ only depend on M. Then the standard construction (of [HU79]) ensures the following:

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

iff the PCP-instance $((c(x), u_2, \dots u_m), (\#, v_2, \dots, v_m))$ 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 in analogy to Example 3: 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 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 complete 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 ϵ to a vertex which has an edge back to itself.

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

Theorem 13. *There is a rational graph with an undecidable first-order theory.*

4 Concluding Remarks

The theory of infinite automata is still in its beginnings. We mention two interesting tracks of research originating in the results outlined above.

For the purposes of model checking, it would be important to obtain classes of transition graphs which are more general (or at least more flexible for modelling concrete systems) than prefix-recognizable graphs but still have good algorithmic properties (e.g. a decidable reachability problem). For example, one can analyze further options of word rewriting for the definition of transition relations (see for example [Bou01]). Or one can pursue the approach developed by C. Löding in [Löd01]: He represents vertices of transition graphs by finite trees (rather than words) and uses ground rewriting as the mechanism to define the transitions.

In this way, interesting graphs are obtained which are not prefix-recognizable (the infinite two-dimensional grid being an example), but where the reachability problem remains decidable.

Another area is the study of language classes (beyond the regular languages) in terms of infinite automata. For example, the relation between nondeterministic and deterministic transition graphs as acceptors is not yet well understood, as well as their relation to the corresponding classical counterparts, pushdown automata and Turing machines. For the determinization problem, it may be useful to analyze the connection with uniformization results on rational relations. Also one may ask whether there are further language classes (besides the context-free and context-sensitive languages) which have appealing characterizations by finitely presented infinite automata. Finally, it would be interesting to see how much of the known general results on context-free, context-sensitive and other languages can be elegantly proved using the framework of infinite automata (instead of the standard approach by grammars, pushdown automata, and Turing machines).

5 Acknowledgment

Thanks a due to Didier Caucal, Thierry Cachat, Christof Löding, and Philipp Rohde for comments on a preliminary version of this paper.

References

[Bar97] K. Barthelmann, On equational simple graphs, Tech. Rep. 9, Universität Mainz, Institut für Informatik 1997.

[Blu00] A. Blumensath, Prefix-recognisable graphs and monadic second-order logic, Aachener Informatik-Berichte 2001-6, Dep. Of Computer Science, RWTH Aachen 2001.

[BG00] A. Blumensath, E. Grädel, Automatic structures, Proc. 15th IEEE Symp. Logic in Computer Science (2000), 51-62.

[Ber79] J. Berstel, *Transductions and Context-Free Languages*, Teubner, Stuttgart 1979.

[BE97] O. Burkart, J. Esparza, More infinite results, *Bull. EATCS* **62** (1997), 138-159.

[Bou01] A. Bouajjani, Languages, rewriting systems, and verification of infinite-state systems, in F. Orejas et al. (Eds.), Proc. ICALP 2001, LNCS 2076 (2001), Springer-Verlag 2001, pp. 24-39.

[Büc60] J. R. Büchi, Weak second-order arirhmetic and finite automata, *Z. Math. Logik Grundl. Math.* **6** (1960), 66-92.

[Büc65] J. R. Büchi, Regular canonical systems, *Arch. Math. Logik u. Grundlagenforschung* **6** (1964), 91-111.

[Cau96] D. Caucal, On transition graphs having a decidable monadic theory, in: F. Meyer auf der Heide, B. Monien (Eds.), Proc. 23rd ICALP, LNCS 1099 (1996), Springer-Verlag 1996, pp. 194-205.

[Cau01] D. Caucal, On transition graphs of Turing machines, in: M. Margenstern, Y. Rogozhin (Eds.), Proc. Conf. on Machines, Computation and Universality, LNCS 2055 (2001), Springer-Verlag 2001, pp. 177-189.

[CK01] D. Caucal, T. Knapik, An internal presentation of regular graphs by prefix-recognizable graphs, *Theory of Computing Systems* **34** (2001), 299-336.

[CGP99] E.M. Clarke, O. Grumberg, D. Peled, *Model Checking*, MIT Press 1999.

[Cou89] B. Courcelle, The monadic second-order logic of graphs II: Infinite graphs of bounded width, *Math. Systems Theory* **21** (1989), 187-221.

[EC*92] D.B.A. Epstein, J.W. Cannon, D.F. Holt, S. Levy, M.S. Paterson, W. Thurston, *Word Processing in Groups*, Jones and Barlett 1992.

[Eil74] S. Eilenberg, *Automata, Languages, and Machines*, Vol. A, Academic Press 1974.

[EM65] C.C. Elgot, J. Mezei, On relations defined by generalized finite automata, *IBM J. Res. Dev.* **9** (1965), 47-68.

[EH*00] J. Esparza, D. Hansel, P. Rossmanith, S. Schwoon, Efficient algorithms for model checking pushdown systems, Proc. CAV 2000, LNCS 1855 (2000), Springer-Verlag 2000, pp. 232-247.

[ES01] J. Esparza, S. Schwoon, A BDD-based model checker for recursive programs, in: G. Berry et al. (Eds.), CAV 2001, LNCS 2102 , Springer-Verlag 2001, pp. 324-336.

[FS93] C. Frougny, J. Sakarovitch, Synchronized rational relations of finite and infinite words, *Theor. Comput. Sci.* **108** (1993), 45-82.

[HU79] J.E. Hopcroft, J.D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley 1979.

[KS86] W. Kuich, A. Salomaa, *Semirings, Automata, Languages*, Springer 1986.

[Löd01] C. Löding, Model-checking infinite systems generated by ground tree rewriting, to appear in Proc. FOSSACS 2002, LNCS, Springer-Verlag 2002.

[McM93] K. L. McMillan, *Symbolic Model Checking*, Kluwer Academic Publishers 1993.

[Mor00] C. Morvan, On rational graphs, in: J. Tiuryn (Ed.), Proc. FOSSACS 2000, LNCS 1784 (2000), Springer-Verlag, pp. 252- 266.

[MoS01] C. Morvan, C. Stirling, Rational graphs trace context-sensitive languages, in: A. Pultr, J. Sgall (Eds.), Proc. MFCS 2001, LNCS, Springer-Verlag 2001, pp. 548-559

[MS85] D. Muller, P. Schupp, The theory of ends, pushdown automata, and second-order logic, *Theor. Comput. Sci.* **37** (1985), 51-75.

[Pen74] M. Penttonen, One-sided and two-sided context in formal grammars, *Inform. Contr.* **25** (1974), 371-392.

[Rab69] M.O. Rabin, Decidability of second-order theories and automata on infinite trees, *Trans. Amer. Math. Soc.* **141** (1969), 1-35.

[Ris01] C. Rispal, The synchronized graphs trace the context-sensitive languages, manuscript, IRISA, Rennes 2001.

[Tho97] W. Thomas, Languages, automata, and logic, in *Handbook of Formal Language Theory* (G. Rozenberg, A. Salomaa, Eds.), Vol 3, Springer-Verlag, Berlin 1997, pp. 389-455.