Verification of infinite state systems (preliminary version)

Angelo Montanari and Gabriele Puppis

Department of Mathematics and Computer Science
University of Udine, Italy
{montana,puppis}@dimi.uniud.it

1 Introduction

These lecture notes are devoted to the analysis of formal methods for the specification and verification of infinite state systems. More precisely, we aim at providing a description of the classes of infinite structures with a decidable model checking problem, that is, of those infinite structures whose verification problem can be solved by automatic procedures. We first provide background knowledge about logic, automata theory, and the model checking problem. Then, we describe the various approaches to the problem of automatically verifying relevant properties of transition systems having an infinite number of states that have been proposed in the literature. Throughout the exposition, we focus our attention on three main topics: (i) logical formalisms for the specification of relevant computational properties, (ii) finite representations of meaningful classes of transitions systems, and (iii) verification problems and their solutions. As for the first topic, we introduce some of the most widespread logics, e.g., first-order and monadic second-order, transitive closure and reachability logics, used to specify properties of infinite state systems and we compare their expressiveness. As for the second topic, we describe some frameworks to finitely represent transition systems with infinite state spaces, distinguishing between extrinsic and intrinsic representations. In particular, we analyze in detail alternative (equivalent) characterizations of meaningful classes of transition systems, including pushdown transition graphs, prefix-recognizable graphs, rational graphs, and graphs in the so-called Caucal hierarchy. As for the third topic, we introduce some paradigmatic problems in the area of automatic verification, e.g., reachability and model checking problems, and we present a variety of methods to cope with them, including logical, symbolic, transformational, and automaton-based methods.

2 Background

In this section, we revise some basic notation and terminology about words, languages, graphs, transition systems, and logics.

2.1 Words and languages

A *finite word* over an alphabet A is a mapping w from a set $\{1, ..., n\}$, where $n \in \mathbb{N}$, to A. The parameter n is said to be the length of the word w, shortly denoted $|w|$, and for every index $1 \leq i \leq n$, $w[i]$ denotes the i-th character of w. The symbol ε denotes the empty word (hence, $|\varepsilon| = 0$). An *infinite word* (or ω-word) over A is a mapping w from $\mathbb{N}$ to A and we assume that its length $|w|$ is ω. The concatenation operation $\cdot$ maps a pair of finite words u, v to the finite word $u \cdot v$ having length $|u| + |v|$ and such that $(u \cdot v)[i] = u[i]$ for every $i \in |u|$ and $(u \cdot v)[i + |u|] = v[i]$ for every $i \in |v|$.

A *language* (resp., an *ω-language*) over the alphabet A is a set of finite (resp., infinite) words over A. If L_1 and L_2 are two languages, then we denote by $L_1 \cdot L_2$ the language $\{u \cdot v : u \in L_1, v \in L_2\}$. Accordingly, if L is a language, then we denote by L^k the set of all finite words of the form $u_1 \cdot ... \cdot u_k$, where each u_i is a word from L (note that $L^0 = \{\varepsilon\}$). We further denote by L^* the language $\bigcup_{k \in \mathbb{N}} L^k$ and by L^ω the ω-language $\{v_1 \cdot v_2 \cdot ... : \forall\, i > 0.\ v_i \in L \setminus \{\varepsilon\}\}$.

2.2 Graphs and trees

We use the term *label* (resp., *color*) to mean a symbol, usually taken from a finite set A (resp., C), marking an edge (resp., a vertex) of a graph. An A-labeled (directed simple[1]) graph is a tuple $G = (V, (E_a)_{a \in A})$, where V, also denoted $\mathcal{D}om(G)$, is a set of vertices and $(E_a)_{a \in A}$ are binary relations defining the edges and their labels.

Given an edge $e = (v, v') \in E_a$, v is called the *source* of e and v' is called the *target* of e. A vertex v' is said to be *adjacent* to v if $(v, v') \in E_a$ or $(v', v) \in E_a$ for some $a \in A$ (namely, if v and v' are ends of the same edge). A vertex v is an *a-predecessor* of v' (or, equivalently, v' is an *a-successor* of v) in G if G contains an a-labeled edge from v to v'. We say that d is the *out-degree* of a vertex $v \in \mathcal{D}om(G)$ if there are exactly d successors of v in G. Furthermore, we say that the graph is *deterministic* if, for every $v \in \mathcal{D}om(G)$ and for every $a \in A$, there is at most one a-successor of v in G. Clearly, under the proviso that the set of labels is finite, the vertices in any deterministic graph have uniformly bounded out-degree, namely, there is k $(= |A|)$ such that for every vertex $v \in \mathcal{D}om(G)$, the out-degree of v does not exceed k.

Given a graph $G = (V, (E_a)_{a \in A})$ and a set $W \subseteq V$ of vertices, the *subgraph of G induced by W* is the graph $G|_W = (W, (E'_a)_{a \in A})$, where, for each $a \in A$, $E'_a = E_a \cap (W \times W)$. An *expanded graph* is a graph equipped with a tuple $\bar{V}$ of unary predicates, namely, a structure of the form $(V, (E_a)_{a \in A}, \bar{P})$, where $\bar{P} = (P_1, ..., P_m)$ and $P_i \subseteq V$ for all $1 \leq i \leq m$. Any expanded graph $(G, \bar{P})$ is canonically represented by a *C-colored* graph $G_{\bar{P}} = (G, \Omega)$, called the *canonical representation* of $(G, \bar{P})$, where $C = \mathscr{P}(\{1, ..., m\})$ and $\Omega : \mathcal{D}om(G) \rightarrow C$ is a coloring function that maps a vertex $v \in \mathcal{D}om(G)$ to the set of all indices $i \in \{1, ..., m\}$ such that $v \in P_i$. Hereafter, we shortly denote the color of a vertex v in $G_{\bar{P}}$ by $G_{\bar{P}}(v)$.

A finite *path* in G from v to v' is a finite sequence $\pi = e_1 e_2 ... e_n$ of edges such that (i) the source of e_1 is v, (ii) the target of e_n is v', and (iii) for all $1 \leq i \leq n$, the target of e_i is the source of e_{i+1}. The *length* $|\pi|$ of the path π is the number of its edges. We similarly define infinite paths, namely, infinite sequences of edges the form $\pi = e_1 e_2 e_3 ...$ (in such a case, we assume that $|\pi| = \omega$). Given a finite (resp., infinite) path π, we say that $w \in A^*$ (resp., $w \in A^\omega$) is the sequence of labels along π (or, equivalently, that π is labeled by w) if $|w| = |\pi|$ and, for all $1 \leq i \leq |\pi|$ (resp., for all $i \geq 1$), $w[i]$ is the label of the i-th edge $\pi[i]$ in π. Notice that, if the graph G is deterministic, then the path π is uniquely determined by the sequence of its labels.

Given two graphs G and G', an *isomorphism* from G to G' is a bijection f from $\mathcal{D}om(G)$ to $\mathcal{D}om(G')$ such that (v, v') is an a-labeled edge of G iff $(f(v), f(v'))$ is an a-labeled edge of G'. If G and G' are colored graphs, we further require that v is a c-colored vertex of G iff $f(v)$ is a c-colored vertex of G'. Intuitively, an isomorphism is simply a renaming of the vertices of a graph. If there exists an isomorphism from a (colored) graph G to a (colored) graph G', then we say that G and G' isomorphic. Note that the relation of isomorphism is an equivalence, namely, it is transitive, reflexive, and symmetrical. Unless strictly necessary, we shall not distinguish between isomorphic graphs. We now recall the notion of *bisimilarity*, which gives rise to another equivalence relation over graphs. Two graphs G and G' are said to be bisimilar if there is a relation $\sim \subseteq \mathcal{D}om(G) \times \mathcal{D}om(G')$, called bisimulation, such that for every pair of vertices $v \in \mathcal{D}om(G)$ and $v' \in \mathcal{D}om(G')$ and for every label $a \in A$, if $v \sim v'$, then for every a-successor w of v in G, there is an a-successor w' of v' in G' (and vice versa, for every a-successor w' of v', there is an a-successor w of v) such that $w \sim w'$. If G and G' are colored graphs, we further require that $v \sim v'$ implies that $G(v) = G'(v')$. Note that the bisimilarity relation is coarser than the isomorphism relation, since there exist some non-isomorphic bisimilar graphs, but isomorphic graphs are always bisimilar.

We shall also consider (rooted) *trees*, namely, graphs such that for every vertex v, there exists a unique path, called *access path*, from a distinguished source vertex (called *root*) to v. We say that v

[1] A directed graph is said to be *simple* if distinct edges cannot have the same source vertex, the same target vertex, and the same label.

is an *ancestor* of v' (or, equivalently, v' is a *descendant* of v) in a tree if there is a (possibly empty) path from v to v'. The *leaves* of a tree T are all and only the vertices $v \in \mathcal{D}om(T)$ such that for every $a \in A$, $v \cdot a \notin \mathcal{D}om(T)$; the other vertices are called *internal vertices*. The *frontier* of a tree T, denoted $\mathcal{F}r(T)$, is the set of all leaves of T.

Notice that the ancestor relation induces a partial left-linear order in the domain of the tree. Hence, we can identify each vertex in a tree (resp., in a deterministic tree) with its unique access path (resp., with the unique sequence of labels along its access path). In particular, we can view any *deterministic A-labeled C-colored tree* as a partial function T from A^* to C, whose domain $\mathcal{D}om(T)$ is a prefix-closed language over the set A of edge labels (a language L is prefix-closed if for every $u \in L$ and for every prefix v of u, $v \in L$). In this perspective, we shall denote the color of a vertex v in T by $T(v)$ and, whenever T is well understood, the a-successor of v in T by $v \cdot a$. Sometimes, if there is a well understood ordering of the labels in the set A, we can also use (unranked) terms to denote trees; for instance, if $A = \{a_1 < a_2 < a_3\}$, then $\emptyset$ denotes the empty tree and $c\langle d, d, \emptyset\rangle$ denotes the ternary tree consisting of a c-colored root and two d-colored leaves, which are targets of two edges labeled respectively by a_1 and a_2.

A *full* tree is a *deterministic* tree such that, whenever $(u, u \cdot a) \in E_a$ holds for some $a \in A$, then $(u, u \cdot a') \in E_{a'}$ holds for every $a' \in A$. An *infinite complete* tree is a *deterministic* tree T such that for every word $w \in A^*$, there is an access path from the root of T labeled with w. A tree T is said to be *regular* if T contains only finitely many non-isomorphic subtrees.

We now recall the notion of *unfolding* of a graph. Given a (colored) graph G and a source vertex $v_0 \in \mathcal{D}om(G)$, the unfolding of G from v_0, denoted $\mathcal{U}nf(G, v_0)$ is the (colored) tree whose domain consists of all and only the finite paths from v_0 to a vertex $v \in \mathcal{D}om(G)$, and where the a-labeled edges are all and only the pairs of paths (π, π') such that π' extends π with an a-labeled edge of G. Note that, if every vertex of G is reachable from v_0, then the unfolding $\mathcal{U}nf(G, v_0)$ is a tree bisimilar to G. Moreover, if G is deterministic, then $\mathcal{U}nf(G, v_0)$ is deterministic as well and this implies that $\mathcal{U}nf(G, v_0)$ is the unique tree (up to isomorphisms) which is bisimilar to G (to see this, simply notice that bisimilar deterministic trees are isomorphic). Finally, it is well known that the unfolding of a finite graph yields a regular tree having finite out-degree.

2.3 Transition systems and their configuration graphs

With the term *transition system* we refer to any (abstract) machine that has a set of possible configurations and some rules that tell the machine how to switch from one configuration to another configuration (possibly in a non-deterministic way). A *transition* of the system is defined as a single pair consisting of a source configuration and a target configuration. One can assign to each transition a *label*, which can be interpreted either as the event (from an hypothetical external environment) that made the system change configuration or as the action that the system undertook when it changed configuration. If transitions are equipped with labels, then the system is said to be a *labeled transition system*. Similarly, one can sometimes assign markings to the configurations of a transition system, thus making it possible to distinguish between different types of configurations (e.g., good configurations or dangerous ones). For the sake of clearness, in order to differentiate the markings on the configurations from those on the transitions, we call the former ones *colors*.

Formally, a (labeled) transition system can be defined as a tuple $\mathcal{M} = (Q, A, \delta)$, where Q is a (possibily infinite and uncountable) set of configurations, A is a set of labels (usually A is assumed to be finite), and $\delta \subseteq Q \times A \times Q$ is a transition relation. We use $p \xrightarrow[\mathcal{M}]{a} q$ to mean that the system $\mathcal{M}$ can go from the configuration $p \in Q$ to the configuration $q \in Q$ on event $a \in A$. A labeled transition system $\mathcal{M} = (Q, A, \delta)$ is often represented by an A-labeled graph $G = (Q, (E_a)_{a \in A})$, where the vertices represent system configurations and the edges E_a represent a-labeled transitions, namely, $(p, q) \in E_a$ iff $p \xrightarrow[\mathcal{M}]{a} q$. Such a graph G is called the *transition graph*, or *configuration graph*, of the system $\mathcal{M}$.

Labeled transition systems (in particular automata, which are special forms of transition systems) are often extended with ε-moves, namely, with transitions of the form $p \xrightarrow{\varepsilon} q$ which allow them to go from a configuration p to another configuration q with no event occurrence. When the transition systems are used as acceptors of languages, it is often the case (see, for instance, finite-state automata, Büchi automata, pushdown automata) that any given transition system $\mathcal{M}$ with ε-moves can be transformed into an ε-free transition system that recognizes the same language. Thus, we can say that, with respect to trace equivalence (i.e., the equivalence that pairs acceptors of the same language), ε-free systems are as expressive as systems with ε-moves. However, one can exploit ε-moves of transition systems to define richer graph structures. For any given labeled transition system $\mathcal{M}$ with ε-moves, we assume the following property: if $p \xrightarrow[\mathcal{M}]{\varepsilon} q$, then there exist no symbol $a \neq \varepsilon$ and no state q' such that $p \xrightarrow[\mathcal{M}]{a} q'$. This intuitively means that, at any given configuration, the transition system can either perform ε-moves only or perform A-labeled moves only. Our interest is in the collapsed transition graph of $\mathcal{M}$, where every source of an ε-labeled edge is identified with the corresponding target. More formally, given the $A \cup \{\varepsilon\}$-labeled transition graph G of $\mathcal{M}$, we define the *ε-closure* of G as the graph obtained from G by first adding an a-labeled edge between vertices p and q whenever there is an a-labeled path from p to q and secondly by removing all vertices with outgoing ε-transitions.

A transition system may have infinitely many possible configurations and in such a case it is commonly called an *infinite transition system* or an *infinite state system*. Infinite transition systems are well-suited to model (concrete) dynamical systems, such as computer programs and devices. In such a case, their configurations usually arise from a finite control part that operates on an infinite data domain. In order to effectively manipulate infinite transition systems, one should provide them with *finite presentations*, namely, finite objects representing them. We can distinguish between two kinds of presentations: *internal* and *external* ones. In the case of internal presentations, infinite state systems are defined as the transition graphs of suitable automata or rewriting systems. In the case of external presentations, infinite state systems are described (up to isomorphisms) as the graphs resulting from applications of suitable transformations, starting from well-known structures (e.g., the infinite binary complete tree). Several classes of infinite transitions systems enjoy a number of alternative characterizations based on both internal and external presentations.

Automata. In the following, we consider particular forms of transition systems, namely, (finite-state, Büchi, pushdown, Rabin, Muller, parity, ...) automata over finite and infinite words and over trees. Automata can be viewed as transition systems equipped with some initial configurations and some acceptance conditions. They are usually used as devices that recognize languages consisting of finite/infinite words/trees. In the following, we recall the basic definitions and some well-known results from automata theory (for a complete discussion on sequential and tree automata, we refer to [48, 70]).

Definition 1. *A* sequential automaton *is a tuple $\mathcal{A} = (S, A, \delta, \mathcal{I}, \mathcal{F})$, where*
- *S is a finite set of* states *(here the terms configuration and state are interchangable),*
- *A is a finite alphabet,*
- *$\delta \subseteq S \times A \times S$ is a transition relation,*
- *$\mathcal{I} \subseteq S$ is a set of* initial states*,*
- *$\mathcal{F} \subseteq S$ is a set of* final states*.*

A *run* of $\mathcal{A}$ over a finite (resp., infinite) word $w \in A^*$ (resp., $w \in A^\omega$) is a finite (resp., infinite) sequence of states ρ such that
- $|\rho| = |w| + 1$ ($|\rho| = |w| = \omega$ if w is infinite),
- for every $1 \leq i < |\rho|$, $(\rho[i], w[i], \rho[i+1]) \in \delta$.

Acceptance conditions can be defined for both finite and infinite words.

If w is a finite word and ρ is a run of $\mathcal{A}$ on w such that $\rho[1] \in \mathcal{I}$ and $\rho[|\rho|] \in \mathcal{F}$, then we say that ρ is a *successful run* on w. The finite word w is *accepted* by $\mathcal{A}$ iff there exists a successful run of $\mathcal{A}$

on w. The language *recognized* by $\mathcal{A}$, denoted $\mathscr{L}(\mathcal{A})$, is the set of all and only the finite words that are accepted by $\mathcal{A}$. The languages recognized by a finite-state sequential automaton are called *regular* (or *rational*) languages.

It can be easily showed (see for instance [48]) that regular languages are effectively closed under union, intersection, and complementation, namely, given two finite-state automata $\mathcal{A}_1$ and $\mathcal{A}_2$, one can compute a finite-state automaton $\mathcal{A}_1 \cup \mathcal{A}_2$ (resp., $\mathcal{A}_1 \cap \mathcal{A}_2$, $\bar{\mathcal{A}}_1$) that recognizes the language $\mathscr{L}(\mathcal{A}_1) \cup \mathscr{L}(\mathcal{A}_2)$ (resp., $\mathscr{L}(\mathcal{A}_1) \cap \mathscr{L}(\mathcal{A}_2)$, $A^* \setminus \mathscr{L}(\mathcal{A}_1)$).

When considering languages of infinite words, the automaton $\mathcal{A}$ is called (sequential) *Büchi automaton* and a different kind of acceptance condition is adopted (here we only consider Büchi acceptance conditions, even though different, but equivalent, acceptance conditions can be found in the literature). Given a run ρ of $\mathcal{A}$ on an infinite word w, we say that ρ is *successful* iff $\rho[1] \in \mathcal{I}$ and there is at least one final state $s \in \mathcal{F}$ that occurs infinitely often in ρ. In such a case w is *accepted* by $\mathcal{A}$. The ω-language *recognized* by a Büchi automaton $\mathcal{A}$, denoted $\mathscr{L}^\omega(\mathcal{A})$, is the set of all and only the infinite words that are accepted by $\mathcal{A}$. The Büchi-recognizable languages are also called *regular* (or *rational*) ω-languages.

In [6] Büchi showed that regular ω-languages are effectively closed under union, intersection, and complementation, namely, given two Büchi automata $\mathcal{A}_1$ and $\mathcal{A}_2$, one can compute a Büchi automaton $\mathcal{A}_1 \cup \mathcal{A}_2$ (resp., $\mathcal{A}_1 \cap \mathcal{A}_2$, $\bar{\mathcal{A}}_1$) that recognizes the ω-language $\mathscr{L}^\omega(\mathcal{A}_1) \cup \mathscr{L}^\omega(\mathcal{A}_2)$ (resp., $\mathscr{L}^\omega(\mathcal{A}_1) \cap \mathscr{L}^\omega(\mathcal{A}_2)$, $A^\omega \setminus \mathscr{L}^\omega(\mathcal{A}_1)$).

We now introduce tree automata, namely, automata recognizing sets of (possibly infinite) colored trees.

Definition 2. *A* tree automaton *is a tuple* $\mathcal{A} = (S, A, C, \delta, \mathcal{I}, Acc)$*, where*

- S *is a finite set of states,*
- A *is a finite set of edge labels,*
- C *is a finite set of vertex colors,*
- $\delta \subseteq S \times C \times S^A$ *is a transition relation,*
- $\mathcal{I} \subseteq S$ *is a set of* initial states*,*
- $Acc \subseteq S^\omega$ *is an* acceptance condition*, namely, a set consisting of infinite sequences of states.*

Given an *infinite complete* A-labeled C-colored tree T, a *run* of the automaton $\mathcal{A}$ on T is any infinite complete A-labeled S-colored tree R such that for every $v \in \mathcal{D}om(R)$,

$$(R(v), T(v), (R(v \cdot a))_{a \in A}) \in \delta$$

where $v \cdot a$ denotes the a-successor of v in R. We say that R is *successful*, and hence T is *accepted* by $\mathcal{A}$, if $R(\varepsilon) \in \mathcal{I}$ and for every infinite path π in R, $R|\pi$ (i.e., the sequence of colors of R along π) belongs to Acc. The language $\mathscr{L}(\mathcal{A})$ *recognized* by $\mathcal{A}$ is the set of all and only the (infinite complete) trees accepted by $\mathcal{A}$.

Like Büchi automata, one can use different, but equivalent, acceptance conditions, usually envisaging the occurrences of states in an infinite sequence. Given an infinite sequence $\alpha \in S^\omega$, we denote by $\mathcal{I}nf(\alpha)$ the set of all elements that occur infinitely often in α. We further denote by $\mathcal{I}mg(\alpha)$ the set of all elements that occur *at least once* in a (finite or infinite) sequence α. The following acceptance conditions are often used when dealing with tree automata:

- *Rabin acceptance conditions* are of the form

$$Acc = \{\alpha \in S^\omega \,:\, \exists\, 1 \leq i \leq n.\, \mathcal{I}nf(\alpha) \cap E_i = \emptyset \;\wedge\; \mathcal{I}nf(\alpha) \cap F_i \neq \emptyset\}$$

  where $(E_i, F_i)_{1 \leq i \leq n}$ is a finite set of acceptance pairs,
- *Rabin chain acceptance conditions* are Rabin acceptance conditions where the acceptance pairs satisfy

$$E_1 \subsetneq F_1 \subsetneq E_2 \subsetneq F_2 \subsetneq ... \subsetneq E_n \subsetneq F_n$$

- *parity acceptance conditions* are of the form

$$Acc = \{\alpha \in S^\omega \,:\, max(\mathcal{I}nf(\Omega(\alpha))) \text{ is even}\}$$

  where Ω is a function that maps states to integers and $\Omega(\alpha)$ is its natural extension to an infinite sequence α of states,
- *Muller acceptance conditions* are of the form

$$Acc = \{\alpha \in S^\omega \,:\, \mathcal{I}nf(\alpha) \in \mathcal{F}\}$$

  where $\mathcal{F} \subseteq \mathscr{P}(S)$ is a family of sets of states.

It should be noted that Rabin chain acceptance conditions and parity acceptance conditions are equivalent. Indeed, given a set of acceptance pairs $(E_i, F_i)_{1\leq i\leq n}$, with $E_i \subsetneq F_i$ for all $1 \leq i \leq n$ and $F_i \subsetneq E_{i+1}$ for all $1 \leq i < n$, one can set $\Omega(s) = 2i-1$ for all $s \in E_i \setminus F_{i-1}$ and $\Omega(s) = 2i$ for all $s \in F_i \setminus E_i$. Conversely, given a function $\Omega : S \to \{1, ..., 2n\}$ such that $\forall\, 1 \leq i \leq 2n.\ \exists\, s \in S.\ \Omega(s) = i$ (if this is not the case, one can extend the automaton with new states and possibly shift the values of Ω), one can set $E_i = \{s \in S : \Omega(s) \leq 2i-1\}$ and $F_i = \{s \in S : \Omega(s) \leq 2i\}$. In fact, it can be proved that Rabin tree automata, Rabin chain tree automata, parity tree automata, and Muller tree automata are all equivalent notions of automata in the sense that, given any such automaton, one can compute a Rabin, Rabin chain, parity, and Muller tree automaton recognizing the same language [45, 70, 61, 7].

In [64] Rabin generalizes the results of Büchi for infinite words to the case of infinite complete colored trees. He proved that the languages recognized by Rabin tree automata are effectively closed under union, intersection, and complementation.

We can further modify the notion of tree automaton in order to allow computations over *non-complete* trees. To do that, it is sufficient to extend the input alphabet C of an automaton $\mathcal{A}$ with a fresh symbol $\perp \notin C$ and assume that, whenever $v \notin \mathcal{D}om(T)$, $\mathcal{A}$ reads $\perp$ on the fictitious vertex v.

Finally, we define pushdown automata, which are sequential automata equipped with a stack (i.e., a 'first-in-last-out' list) and working over finite words.

Definition 3. *An ε-free* pushdown automaton *is a tuple $\mathcal{P} = (S, A, \Gamma, \delta, s_0, \gamma_0, \mathcal{F})$, where*
- *S is a finite set of* control states*,*
- *A is a finite alphabet of input letters,*
- *Γ is a finite alphabet of stack symbols,*
- *$\delta \subseteq S \times A \times \Gamma \times \Gamma^* \times S$ is a transition relation,*
- *$(s_0, \gamma_0) \in S \times \Gamma$ is an* initial configuration*,*
- *$\mathcal{F} \subseteq S$ is a set of final control states.*

A *configuration* of $\mathcal{P}$ is given by a pair consisting of a control state $s \in S$ and a stack content $w \in \Gamma^*$. The automaton $\mathcal{P}$ can read a symbol $a \in A$ and move from configuration (s, w) to configuration (s', w'), and we shortly write $(s, w) \xrightarrow[\mathcal{P}]{a} (s', w')$, iff there exist $\gamma \in \Gamma$ and $u, v \in \Gamma^*$ such that (i) $w = \gamma \cdot v$, (ii) $(s, a, \gamma, u, s') \in \delta$, and (iii) $w' = u \cdot v$ (notice that no transition is enabled when the stack is empty). A *run* of $\mathcal{P}$ over a finite word $w \in A^*$ is a finite sequence of configurations ρ such that
- $|\rho| = |w| + 1$,
- for every $1 \leq i < |\rho|$, $\rho[i] \xrightarrow[\mathcal{P}]{w[i]} \rho[i+1]$.

In analogy to finite-state automata, we use acceptance by final states (acceptance by empty stack can used as an alternative, but equivalent, choice): we say that the run ρ on w is successful and w is accepted by $\mathcal{P}$ if ρ ends in a configuration where the control state belongs to $\mathcal{F}$.

The languages recognized by pushdown automata are called *context-free* languages and they are effectively closed under union, but not under intersection and complementation (however, they are

closed under intersection with regular languages). We refer to [48] for other properties and equivalent characterizations of context-free languages in terms of grammars.

As regards the expressiveness (w.r.t. recognizable languages) of the deterministic versions of the automata introduced so far, it is known that, while deterministic sequential finite-state automata are expressively equivalent to non-deterministic sequential finite-state automata (cf. [48]), this is not the case for sequential Büchi automata, Rabin (Muller, etc.) tree automata, and pushdown automata (cf. [48, 70]).

In the sequel, we shall use the terms *finite-state system* and *pushdown system* to refer to finite-state automata and pushdown automata, respectively, devoid of their initial and final states. An *input-free* automaton is an automaton that does not read any symbol (it simply activates transitions).

Rewriting systems. Another interesting family of transition systems is given by the so-called rewriting systems. These are systems whose configurations are represented by terms (e.g., words or trees) and whose transitions are specified by suitable rules for term rewriting. A natural type of rewriting system is the word rewriting system, defined below.

Definition 4. *A* word rewriting system *is specified by a tuple* $\mathcal{R} = (\Gamma, A, P)$*, where*
- Γ *is a finite alphabet for the words that encode configurations,*
- A *is a finite alphabet for the input letters,*
- $P \subseteq \Gamma^* \times A \times \Gamma^*$ *is a finite set of rewrite rules of the form* (U, a, V)*.*

In order to finitely represent a rewriting system, the sets U and V in each rewriting rule are usually assumed to be *regular* sets. A rewriting of a word $w \in \Gamma^*$ can be constrained to occur at the beginning (i.e., *prefix-rewriting*), at the end (i.e., *suffix-rewriting*), or at a generic position (i.e., *infix-rewriting*). For instance, in a prefix-rewriting system $\mathcal{R} = (\Gamma, A, P)$ a word $u \cdot w$ is rewritten to $v \cdot w$ if there is a rule $(U, a, V) \in P$ satisfying $u \in U$ and $v \in V$. Similarly, in an infix-rewriting system $\mathcal{R} = (\Gamma, A, P)$ a word $w \cdot u \cdot w'$ is rewritten to $w \cdot v \cdot w'$ if there is a rule $(U, a, V) \in P$ satisfying $u \in U$ and $v \in V$. Another variant of rewriting system is the so-called *parallel-rewriting system*, where words are viewed as multi-sets and rewritings may occur for sub-sequences covering non-contiguous positions.

The configuration graph of a rewriting system $\mathcal{R} = (\Gamma, A, P)$ is defined as the graph $G = (V, (E_a)_{a \in A})$, where V consists of all finite words over Γ and each relation E_a consists of all pairs of words (u, v) such that u is rewritten to v by an a-labeled rule of $\mathcal{R}$. Transition graphs of prefix-rewriting systems are called *prefix-recognizable graphs* [9]. It should be clear that the notion of prefix-rewriting system generalizes that of pushdown system: indeed, a pushdown system $\mathcal{P} = (S, A, \Gamma, \delta, s_0, \gamma_0, \mathcal{F})$ is simply a prefix-rewriting system over the alphabet $S \cup \Gamma$ with rules of the form (U, a, V), where U is a singleton contained in $S \cdot \Gamma$ and V is a singleton contained in $S \cdot \Gamma^*$. In Section 4 we show that the prefix-recognizable graphs are exactly the ε-closures of transition graphs of pushdown automata with ε-moves.

Other interesting examples of rewriting systems are the ground tree rewriting systems [55], where trees, instead of words, are used.

2.4 Logics for specifying and verifying infinite state systems

A natural approach to verification problems for infinite state systems is to model a system as a directed graph (i.e., the transition graph), whose vertices (resp., edges) represent system configurations (resp., transitions). Properties of the system can then be expressed by logical formulas, which can be satisfied or not by the corresponding graph, thought of as a relational structure. Thus, verification problems for infinite state systems are often reduced to the model-checking problem (i.e., checking that a formula without free variables holds in a given structure) or to the satisfiability problem (i.e., checking that a formula with free variables is satisfiable in a given structure).

In the sequel, we review some logics which are commonly used to express properties of infinite state systems and discuss related decision problems. We start by giving some preliminary definitions. A *signature* is a tuple $\Sigma = (\mathcal{R}, k)$ which consists of a finite set $\mathcal{R}$ of relational symbols and a ranking function k from $\mathcal{R}$ to $\mathbb{N}^+$. A *relational structure* in the signature Σ, or simply a Σ-relational structure, is a tuple $S = (V, (r^S)_{r \in \mathcal{R}})$, where V, also denoted $\mathcal{D}om(S)$, is a (possibly infinite) set of elements and each $r^S \subseteq V^{k(r)}$ is an interpretation of the relational symbol $r \in \mathcal{R}$. Unless otherwise stated, we assume that a signature Σ contains at least the relational symbol $=$, which is interpreted, in any relational structure S, by the equality relation $\{(v, v) : v \in \mathcal{D}om(S)\}$. Moreover, by a slight abuse of notation, sometimes we shall not distinguish between the relational symbols in a signature Σ and the corresponding interpretations in a Σ-relational structure S. Finally, notice that transition systems and graphs structures can be thought of as special cases of relational structures. This allows us to use logics to specify and check properties of infinite state systems.

First-order logic. First-order (shortly, FO) logic is one the most simple classical logics for expressing properties of systems. Let fix a signature $\Sigma = (\mathcal{R}, k)$. FO-formulas in the signature Σ are built up starting from atoms of the form $r(x_1, ..., x_{k(r)})$, where $r \in \mathcal{R}$ and each x_i is a variable from a countable set $\mathcal{X}$. Atomic formulas can be combined by means of the Boolean connectives $\vee$ and $\neg$ and the existential quantifier $\exists$.

We say that a variable that occurs in a formula ψ is *free* if it is not bounded by any existential quantifier and we write $\psi(x_1, ..., x_n)$ to mean that the free variables in ψ are *only* $x_1, ..., x_n$.
In order to define the semantics of an FO-formula, we need to introduce assignments for first-order variables. Given a relational structure S in the signature Σ, an *assignment* for a FO-formula ψ is a function θ that maps each variable to a corresponding element in $\mathcal{D}om(S)$. We say that ψ *holds* in the expanded relational structure (S, θ), and we shortly write $(S, \theta) \vDash \psi$, if one of the following conditions holds:

- ψ is of the form $r(x_1, ..., x_n)$ and $(\theta(x_1), ..., \theta(x_n)) \in r_S$,
- ψ is of the form $\psi_1 \vee \psi_2$ and $(S, \theta) \vDash \psi_1$ or $(S, \theta) \vDash \psi_2$ holds,
- ψ is of the form $\neg\psi'$ and $(S, \theta) \nvDash \psi'$,
- ψ is of the form $\exists\, x.\ \psi'$ and there is $v \in \mathcal{D}om(S)$ such that $(S, \theta[v/x]) \vDash \psi'$, where $\theta[v/x]$ is the extension of θ with the assignment v for x.

In the following, we shall use $\psi_1 \wedge \psi_2$, $\psi_1 \rightarrow \psi_2$, and $\forall\, x.\ \psi$ as a shorthands for $\neg(\neg\psi_1 \vee \neg\psi_2)$, $\neg\psi_1 \vee \psi_2$, and $\neg\exists\, x.\ \neg\psi$, respectively.

Monadic second-order logic. Monadic second-order (shortly, MSO) logic is an extension of FO logic with second-order variables (namely, variables that are instanciated by sets of elements rather than single elements) and quantification over second-order variables. Formally, the atoms of MSO-formulas are of the form $x = y$, $x \in Y$, and $r(x_1, ..., x_{k(r)})$, for each relational symbol r in the signature Σ. Atomic formulas can then be combined by means of the Boolean connectives and the existential quantifications over first-order and second-order variables $\exists x$ and $\exists X$. In order to distinguish between first-order and second-order variables, we conventionally denote the former ones by lowercase letters and the latter ones by uppercase letters.
Let S be a relational structure S in the signature Σ and let θ be an assignment that maps each first-order variable (resp. second-order variable) to an element in $\mathcal{D}om(S)$ (resp. a subset of $\mathcal{D}om(S)$). We say that an MSO-formula ψ *holds* in (S, θ) and we shortly write $(S, \theta) \vDash \psi$, if one of the following conditions holds:

- ψ is of the form $x_i = x_j$ and $\theta(x_i) = \theta(x_j)$;
- ψ is of the form $x_i \in X_j$ and $\theta(x_i) \in \theta(X_j)$;
- ψ is of the form $r(x_{i_1}, ..., x_{i_{k(r)}})$ and $(\theta(x_{i_1}), ..., \theta(x_{i_{k(r)}})) \in r_S$;
- ψ is of the form $\psi_1 \vee \psi_2$ and $(S, \theta) \vDash \psi_1$ or $(S, \theta) \vDash \psi_2$ holds;
- ψ is of the form $\neg\psi'$ and $(S, \theta) \nvDash \psi'$;

- ψ is of the form $\exists\, x.\ \psi'$ and there is $v \in \mathcal{D}om(S)$ such that $(S, \theta[v/x]) \vDash \psi'$, where $\theta[v/x]$ is the extension of θ with the assignment v for x;
- ψ is of the form $\exists\, X.\ \psi'$ and there is $P \subseteq \mathcal{D}om(S)$ such that $(S, \theta[P/X]) \vDash \psi'$, where $\theta[P/X]$ is the extension of θ with the assignment P for X.

When writing MSO formulas, we shall use some natural shorthands like $X \subseteq Y$ for $\forall\, z.\ (z \in X \ \rightarrow\ z \in Y)$, $X = Y$ for $X \subseteq Y \ \wedge\ Y \subseteq X$, $X = \emptyset$ for $\forall\, Y.\ X \subseteq Y$, etc. Moreover, when considering MSO logic, one can restrict to an equivalent setup where only second-order variables are used. In such a case, first-order variables are represented by second-order variables which are restricted, via suitable formulas like $Sing(X) = \forall\, Y.\ (Y = \emptyset \ \vee\ X = Y \ \vee\ X \subseteq Y)$, to be instanciated by singletons. In this framework, any assignment θ for the free variables $X_1, ..., X_m$ can be encoded by the tuple of predicates $(\theta(X_1), ..., \theta(X_m))$ and any graph structure G expanded with the assignment θ can be viewed as a $\mathscr{P}(\{1, ..., m\})$-colored graph.

MSO logic is powerful enough to express several non-trivial properties of relational structures, like, for instance, planarity of graphs. A simple example is given by the reflexive and transitive closure r^* of a binary relation r, which can be expressed by the following MSO-formula:

$$\psi(x, y) = \forall\, X.\ (\forall\, z, w.\ (z \in X \ \wedge\ r(z, w) \ \rightarrow\ w \in X) \ \wedge\ \forall\, z.\ (r(x, z) \ \rightarrow\ z \in X)) \ \rightarrow\ y \in X.$$

Moreover, suitable fragments of MSO logics have been defined in the literature. For instance, when considering graphs structures, one can define the path (resp. the chain) fragment of MSO logic by restricting the second-order variables in a formula to be instanciated (via assignments) by paths (resp. subsets of paths). These frameworks are usually less expressive than plain MSO logic, but they still allow one to express interesting graph properties like reachability.

Transitive closure logic. Transitive closure logic (shortly, FO(TC) logic) is obtained by adding the transitive closure operator (TC) to FO logic: if $\psi(\bar{x}, \bar{y}, \bar{z})$ is an FO[TC]-formula, $\bar{x}$, $\bar{y}$, $\bar{z}$ are disjoint tuples of free variables, $\bar{x}$, $\bar{y}$ are of the same length $k > 0$, and $\bar{s}$, $\bar{t}$ are k-tuples of variables, then the formula

$$\phi(\bar{s}, \bar{t}, \bar{z}) = [TC_{\bar{x}, \bar{y}} \psi(\bar{x}, \bar{y}, \bar{z})](\bar{s}, \bar{t})$$

is an FO[TC]-formula as well. Notice that the variables $\bar{x}$ and $\bar{y}$ inside the formula $\phi(\bar{s}, \bar{t}, \bar{z})$ are bound by the TC-operator, while the variables $\bar{z}$ (if any) are free.
Given a relational structure S in the signature Σ and an assignment θ for the free variables $\bar{s}$, $\bar{t}$, $\bar{z}$, the formula $\phi(\bar{s}, \bar{t}, \bar{z})$ holds in (S, θ) iff the tuple $(\theta(\bar{s}), \theta(\bar{t}))$ belongs to $E^*(\theta(\bar{z}))$, where E is the function that map a tuple $\bar{w}$ of elements to the set $\{(\bar{a}, \bar{b}) \ :\ (S, \theta[\bar{a}/\bar{x}, \bar{b}/\bar{y}, \bar{w}/\bar{z}]) \vDash \psi\}$ and E^* is its reflexive and transitive closure, namely, $(\bar{u}, \bar{v}) \in E^*(\bar{w})$ iff there exists a finite (possibly empty) sequence $\bar{v}_0, ..., \bar{v}_n$ of tuples of elements such that $\bar{v}_0 = \bar{u}$, $\bar{v}_n = \bar{v}$, and $(\bar{v}_i, \bar{v}_{i+1}) \in E(\bar{w})$ for all $0 \leq i < n$.

It easy to prove that FO(TC) logic is even more expressive than MSO logic. As an example, consider the equi-level relation that holds between two vertices in the infinite binary complete tree iff they are at the same distance from the root. It is easy to see that such a relation is not expressible by any MSO-formula. However, it can be expressed by the following FO(TC)-formula:

$$\phi(s, t) = [TC_{\bar{x}, \bar{y}} \psi(\bar{x}, \bar{y})](\varepsilon, \varepsilon, s, t)$$

where ε denotes the root of the tree (it can be easily defined by a suitable FO-formula), $\bar{x} = (x_1, x_2)$, $\bar{y} = (y_1, y_2)$, and $\psi(x_1, x_2, y_1, y_2) = (succ_1(x_1, y_1) \ \vee\ succ_2(x_1, y_1)) \ \wedge\ (succ_1(x_2, y_2) \ \vee\ succ_2(x_2, y_2))$ (intuitively, the formula ψ states that y_1 is one level below x_1 and y_2 is one level below x_2).

We can define fragments of FO(TC) logic by constraining the number of variables that are bounded by the TC-operator. Precisely, we denote by $\mathrm{FO(TC)}_{(k)}$ the fragment of FO(TC) such that for every (sub-)formula $[TC_{\bar{x}, \bar{y}} \psi(\bar{x}, \bar{y}, \bar{z})](\bar{s}, \bar{t})$, we have $|\bar{x}| = |\bar{y}| \leq k$. As an example, in $\mathrm{FP(TC)}_{(1)}$ logic only binary relations (i.e., edges in a graph) can be defined using the TC-operator. This implies that

$\mathrm{FP(TC)}_{(1)}$ logic over graph structures is subsumed by MSO logic; indeed, the following equivalence holds for every graph G, for every assignment θ, and for every FO-formula $\psi(x, y, \bar{z})$:

$$\begin{aligned} & (G, \theta) \models [TC_{x,y}\psi(x, y, \bar{z})](s, t) \\ \text{iff} \quad & (G, \theta) \models \exists\, Z.\, (\theta(Z, s, t, \bar{z}) \ \wedge\ y \in Z \ \wedge\ \forall\, W.\, (\theta(W, s, t, \bar{z}) \ \rightarrow\ Z \subseteq W)) \end{aligned}$$

where $\theta(Z, s, t, \bar{z}) = s \in Z \ \wedge\ \forall\, x, y.\, (x \in Z \ \wedge\ \psi(x, y, \bar{z})) \ \rightarrow\ y \in Z$.

Reachability logics. A common feature of MSO logic and FO(TC) logic is that they both allow one to express reachability properties like 'vertex y is reachable from vertex x through a path with labels over $A' \subseteq A$'. Indeed, the latter property can be expressed by the following FO(TC)-formula:

$$\mathit{Reach}_{A'}(x, y) = [TC_{x,y}(x = y \ \vee\ \bigvee_{a \in A'} E_a(x, y))](x, y).$$

The reachability logic (shortly FO(R) logic) is the restriction of FO(TC) logic where transitive closure formulas are allowed to be of the form $\mathit{Reach}_{A'}(x, y)$ only. Such a logic is clearly less expressive than $\mathrm{FO(TC)}_{(1)}$ and MSO logics, but more expressive than pure FO logic. Sometimes, suitable extensions of FO(R) logic are used, such as, for instance, the FO(Reg) logic, which allows one to express properties like 'vertex y is reachable from vertex x through a path labeled by a word in L', where L is a regular language. In this case, the expressiveness of FO(Reg) logic is in between FO(R) and MSO logics.

Modal logics. Here we briefly describe a basic modal logic. We denote by A a finite set of labels and by $\mathcal{P}$ a countable set of propositional letters. The basic modal logic is an extension of propositional logic where formulas are build up starting from single propositional letters $P \in \mathcal{P}$, via the Boolean connectives $\vee, \neg$ and the modality $\langle a \rangle$, for each $a \in A$. Intuitively, the meaning of a formula $\langle A \rangle \psi$ is the following one: 'there is an a-labeled transition that brings the system in a configuration where ψ holds'. More formally, formulas are evaluated over a certain A-labeled transition system and with respect to a specified assignment θ for propositional letters and a specified initial configuration. We say that a formula ψ holds in the transition system $\mathcal{M} = (Q, A, \delta)$ with the assignment $\theta : \mathcal{P} \rightarrow \mathscr{P}(Q)$ and at the configuration $q \in Q$, and we denote it by $(\mathcal{M}, \theta, q) \models \psi$, if one of the following conditions holds:

- ψ is a single propositional letter P and $q \in \theta(P)$;
- ψ is of the form $\langle a \rangle \psi'$ and there is a configuration $q' \in Q$ such that $(q, a, q') \in \delta$ and $(\mathcal{M}, \theta, q') \models \psi'$;
- ψ is of the form $\psi_1 \vee \psi_2$ and $(\mathcal{M}, \theta, q) \models \psi_1$ or $(\mathcal{M}, \theta, q) \models \psi_2$ holds;
- ψ is of the form $\neg\psi'$ and $(\mathcal{M}, \theta, q) \not\models \psi'$.

We use $[a]\psi$ as a shorthand for $\neg\langle a \rangle \neg\psi$, meaning that 'every a-labeled transition (if any) brings the system in a configuration where ψ holds'. The expressive power of such a modal logic is quite weak: obviously any formula can only make statements about a given finite number of steps in the future. Indeed, any formula written in this basic modal logic can be easily translated into an equivalent FO-formula with one free variable. For instance, given a transition system $\mathcal{M} = (Q, A, \delta)$, an assignment θ, an initial configuration $q \in Q$, and a formula $\psi = \langle a \rangle P$, $(\mathcal{M}, \theta, q) \models \psi$ iff the FO-formula $\exists\, y.\, (E_a(x, y) \ \wedge\ p)$ holds in the transition graph $G = (V, (E_a)_{a \in A})$ of $\mathcal{M}$ expanded with the assignment $\theta(P)$ for for predicate P and the assignment q for the free variable x. It should be noted that different, and more expressive, variants of this basic modal logic (e.g., propositional dynamic logic, computation tree logic, etc.) have been introduced in the literature. As an example, the modal logic CTL* (see [37] for formal definitions) allows one to express properties like 'for every computation, ψ holds until a certain point in the future, where ϕ holds'.

The modal μ-calculus. We now focus on modal μ-calculus (sometimes called propositional μ-calculus), which is a modal logic that subsumes most other commonly used modal logics. For further

details we refer the reader to [54, 13]. The distinctive feature of μ-calculus is the use of monadic second-order variables bound by least and greatest fixed points of definable monotonic functions. An atomic formula in the modal μ-calculus can be a propositional letter $P \in \mathcal{P}$, its negation $\neg P$, or a monadic variable $X \in \mathcal{X}$. Atomic formulas can be then combined to form more complex formulas via the Boolean connectives $\vee$, $\wedge$ (note that negation is allowed on propositional letters only), the modalities $\langle a \rangle$ and $[a]$, with $a \in A$, and the fixpoint operators μX and νX, with $X \in \mathcal{X}$. As usual, formulas are evaluated over a A-labeled transition system $\mathcal{M} = (Q, A, \delta)$ expanded with an assignment $\theta : (\mathcal{P} \cup \mathcal{X}) \rightarrow \mathscr{P}(Q)$ for propositional letters and a distinguished initial configuration $q \in Q$. For the sake of simplicity, we define the semantics of a formula ψ by specifying the set $[\![\psi]\!]_{\mathcal{M},\theta}$ of all configurations where ψ holds. Formally, q belongs to $[\![\psi]\!]_{\mathcal{M},\theta}$ iff $(\mathcal{M}, \theta, q) \vDash \psi$. The set $[\![\psi]\!]_{\mathcal{M},\theta}$ is defined by induction on the structure of ψ as follows:

- $[\![P]\!]_{\mathcal{M},\theta} = \theta(P)$;
- $[\![\neg P]\!]_{\mathcal{M},\theta} = Q \setminus \theta(P)$;
- $[\![X]\!]_{\mathcal{M},\theta} = \theta(X)$;
- $[\![\langle a \rangle \psi]\!]_{\mathcal{M},\theta} = \{q \in Q \,:\, \exists\, q' \in Q.\ (q, a, q') \in \delta \ \wedge\ q' \in [\![\psi]\!]_{\mathcal{M},\theta}\}$;
- $[\![[a]\psi]\!]_{\mathcal{M},\theta} = \{q \in Q \,:\, \forall\, q' \in Q.\ (q, a, q') \notin \delta \ \vee\ q' \in [\![\psi]\!]_{\mathcal{M},\theta}\}$;
- $[\![\psi_1 \ \vee\ \psi_2]\!]_{\mathcal{M},\theta} = [\![\psi_1]\!]_{\mathcal{M},\theta} \cup [\![\psi_2]\!]_{\mathcal{M},\theta}$;
- $[\![\psi_1 \ \wedge\ \psi_2]\!]_{\mathcal{M},\theta} = [\![\psi_1]\!]_{\mathcal{M},\theta} \cap [\![\psi_2]\!]_{\mathcal{M},\theta}$;
- $[\![\mu X.\ \psi]\!]_{\mathcal{M},\theta}$ is the least fixpoint of the monotone[2] function that maps a set $P \subseteq Q$ to the set $[\![\psi]\!]_{\mathcal{M},\theta[P/X]}$, where $\theta[P/X]$ is the extension of θ with the assignment P for X;
- $[\![\nu X.\ \psi]\!]_{\mathcal{M},\theta}$ is the greatest fixpoint of the monotone function that maps a set $P \subseteq Q$ to the set $[\![\psi]\!]_{\mathcal{M},\theta[P/X]}$, where $\theta[P/X]$ is the extension of θ with the assignment P for X.

As regards the definition of the semantics of $\mu X.\ \psi$ (resp. $\nu X.\ \psi$), it should be noted that $[\![\mu X.\ \psi]\!]_{\mathcal{M},\theta}$ (resp. $[\![\nu X.\ \psi]\!]_{\mathcal{M},\theta}$) can be equivalently defined as the intersection (resp. the union) of all sets $P \subseteq Q$ such that $P \supseteq [\![\psi]\!]_{\mathcal{M},\theta[P/X]}$ (resp. $P \subseteq [\![\psi]\!]_{\mathcal{M},\theta[P/X]}$). Notice that the existence of these fixpoints is guaranteed by the well-known Tarsky's Theorem.

As regards the expressiveness of μ-calculus, it is worth to do the following remarks. First of all, as a modal logic, the μ-calculus distinguishes between graph structures up to bisimulation equivalence, that is, given two bisimilar structures $\mathcal{M}$ and $\mathcal{M}'$ (both expanded with a valuation for the propositional letters and the monadic variables and with a marking for the initial configuration), for every formula ψ in the μ-calculus, $\mathcal{M} \vDash \psi$ iff $\mathcal{M}' \vDash \psi$. In particular, this implies that a formula ψ holds in a graph structure $\mathcal{M}$ iff it holds in the unfolding of $\mathcal{M}$ from its initial vertex (recall that the unfolding of a graph is a tree bisimilar to the graph itself). Moreover, in [49], Janin and Walukiewicz proved that the μ-calculus is as expressive as the bisimulation invariant fragment of MSO logic, namely, every MSO-formula that does not distinguish between bisimilar models is equivalent to a suitable formula of the μ-calculus.

2.5 Decision problems

Here we introduce some fundamental decision problems involving infinite state systems and logics. In particular, we define the model checking and the satisfiability problems for logics, and we describe reachability problems. We only briefly mention other interesting decision problems, such as the family of equivalence-checking problems, which consist in testing the existence of isomorphisms, bisimulations, simulations, or trace equivalences between infinite state systems.

Model checking and satisfiability problems. Roughly speaking, a model checking problem consists in establishing, given (a finite representation of) a system and (a logical formula expressing) a desired property of the system, whether the system satisfies the given property.

[2] Note that monotonicity follows from the fact that the variable X always occurs positively in the formula ψ.

Let Σ be a signature, L a logic, and S a relational structure in the signature Σ (for instance S can be a transition graph possibly expanded with an assignment for free variables). The *model checking* problem for L and S is the problem of deciding, given a formula ψ in the logic L, whether $S \models \psi$. We call *L-theory of S* the set of all L-formulas that hold in the structure S. Thus, we say that the L-theory of S is recursive (or decidable) if the model checking problem for L and S is decidable. Relational structures can be part of the input instances of the model-checking problem. In such a case, we say that the L-theories of the relational structures belonging to a suitable class $\mathcal{S}$ are decidable if, for every structure $S \in \mathcal{S}$ and for every L-formula ψ, one can decide whether $S \models \psi$. The *satisfiability* problem for a logic L and a structure S, instead, consists in deciding whether *there exists an assignment* θ such that $(S, \theta) \models \psi$. Note that, in this case, no assignment is specified in the structure S. Despite of the different definitions, the model checking and the satisfiability problems are somehow related. For instance, it is easy to see that, in the case of FO and MSO logics, the latter problem is reducible to the former one by existentially closing formulas with free variables.

When trying to establish the decidability of L-theories for a class $\mathcal{S}$ of relational structures, one should carefully choose a trade-off between the expressiveness of the logic L and the number and the complexity of the structures in $\mathcal{S}$. For instance, let $\mathcal{S}_{Turing}$ be the class of the transition graphs of Turing machines, viewed as transition systems. It is well known that the halting problem for Turing machines is undecidable. Moreover, instances of the halting problem can be easily reduced to suitable reachability problems over the corresponding transition graphs. This means that if we take a logic L expressive enough to encode instances of reachability problems (e.g., $\mathrm{FO(TC)}_{(1)}$), then the class $\mathcal{S}_{Turing}$ cannot enjoy decidable L-theories (in fact, due to the existence of a universal Turing machine, there even exists a single graph in $\mathcal{S}_{Turing}$ with an undecidable L-theory).
As another negative example, one can prove that the $\mathrm{FO(TC)}_{(1)}$-theories of the transition graphs of infix rewriting systems are undecidable. Indeed, given a Turing machine $\mathcal{M}$ working with a set Q of control states and a set A of input letters, one can encode any configuration of $\mathcal{M}$ by a finite word over $Q \cup A$ as follows: if the tape contents is $a_1 \dots a_n \,\square\square\square \dots$, and $\mathcal{M}$ lies at state q with its head under the i-th letter, then we encode such a configuration by the finite word $a_1 \dots, a_i \, q \, a_{i+1} \dots a_n$. Since $\mathcal{M}$ can only move its head by one position at a time, the transitions of $\mathcal{M}$ can be encoded by suitable infix rewriting rules. This shows that the transition graphs of infix rewritings capture those of Turing machines and, from the previous arguments, they cannot enjoy decidable $\mathrm{FO(TC)}_{(1)}$-theories.

Despite the expressive power of MSO logic, a number of positive results involving the decidability of MSO-theories of linear and branching structures have been provided in the literature. Two fundamental results in this field are Büchi's and Rabin's Theorems [6, 45], which show the decidability of the model-checking problem for the linear order $(\mathbb{N}, <)$ and for the infinite binary complete tree T_2, respectively. Both results have been achieved by exploiting closures poperties of sequential Büchi automata (resp. Rabin tree automata) under union, complementation, and projection to reduce the model-checking problem to an acceptance problem for such automata. Precisely, given an MSO-formula $\psi(X_1, \dots, X_m)$, one can compute a Büchi automaton $\mathcal{A}$ (and, conversely, given a Büchi automaton $\mathcal{A}$, one can compute an MSO-formula $\psi(X_1, \dots, X_m)$) such that, for every assignment θ,

$$(\mathbb{N}, <, \theta) \models \psi \quad \text{iff} \quad w_\theta \in \mathscr{L}^\omega(\mathcal{A})$$

where w_θ is the characteristic word of $(\mathbb{N}, <, \theta)$, defined by $w_\theta[i+1] = \{j : 1 \leq j \leq m, i \in \theta(X_j)\}$ for every element $i \in \mathbb{N}$. We call *acceptance problem* of a word w the problem of deciding, for any given Büchi automaton $\mathcal{A}$, whether $\mathcal{A}$ accepts w. Similarly, the *emptiness problem* for Büchi automata is the problem of deciding whether a given Büchi automaton recognizes a non-empty language. The above correspondence between models of MSO-formulas and words accepted by Büchi automata, immediately implies that the MSO-theory of $(\mathbb{N}, <, \theta)$ is decidable iff the acceptance problem of w_θ for Büchi automata is decidable. Moreover, it easy to see that the satisfiability problem for MSO logic interpreted over $(\mathbb{N}, <)$ reduces to the emptiness problem for Büchi automata.
These results can be extended to branching structures, and in particular to the infinite binary complete tree T_2 and to Rabin tree automata. Precisely, Rabin's Theorem establishes a strong correspondence

between MSO-formulas satisfied by an expanded tree structure (T_2, θ) and Rabin tree automata accepting its canonical representation T_θ: for every MSO-formula $\psi(X_1, ..., X_m)$, one can compute a Rabin tree automaton $\mathcal{A}$ (and, conversely, for every Rabin tree automaton $\mathcal{A}$, one can compute an MSO-formula $\psi(X_1, ..., X_m)$) such that, for every assignment θ,

$$(T_2, \theta) \models \psi \quad \text{iff} \quad T_\theta \in \mathscr{L}(\mathcal{A})$$

where T_θ is the canonical representation of (T_2, θ), defined by $T_\theta(v) = \{j : 1 \leq j \leq m, v \in \theta(X_j)\}$ for every vertex v of T_2. In analogy to the linear case, it turns out that the MSO-theory of (T_2, θ) is decidable iff the acceptance problem of T_θ for Rabin tree automata is decidable. Such a result implies that the satisfiability problem for MSO logic interpreted over the infinite binary complete tree T_2 is decidable. Furthermore, the results can be generalized to non-complete trees by introducing a dummy symbol $\perp$, which is read by the tree automaton whenever a vertex is missing (see Section 2.3).

In Sections 4, 5, and 6 we deal with model checking problems for FO and MSO logics interpreted over more complex structures, possibly expanded with some predicates.

Reachability problems. In its most basic formulation, the reachability problem over a (labeled) graph $G = (V, (E_a)_{a \in A})$ consists in deciding, given two (possibly infinite) sets of vertices $I, F \subseteq V$, whether there exists a path π from a (source) vertex $i \in I$ to a (target) vertex $f \in F$. During reachability analysis, we clearly may need to represent infinite sets of vertices by means of suitable symbolic expressions. As an example, an infinite set $S \subseteq V$ of vertices could be represented by a logical formula ψ (holding true exactly at the vertices contained in S) or by an automaton $\mathcal{A}$ (accepting all and only the words that represent the vertices contained in S).

A number of variants of the plain reachability problem has been also introduced in the literature. Among all, we mention:

- recurrent reachability: given $I, F \subseteq V$, to decide whether there exists an infinite path π from a vertex $i \in I$ that meets F infinitely often;
- constrained reachability: given $I, C, F \subseteq V$, to decide whether there exists a path π from a vertex $i \in I$ that remains in C until it eventually reaches a vertex $f \in F$;
- universal reachability: given $I, F \subseteq V$, to decide whether any path π from any vertex $i \in I$ eventually reaches a vertex $f \in F$;
- universal recurrent reachability: given $I, F \subseteq V$, to decide whether any infinite path π from any vertex $i \in I$ meets F infinitely often;
- universal constrained reachability: given $I, C, F \subseteq V$, to decide whether any path π from any vertex $i \in I$ remains in C until it eventually reaches a vertex $f \in F$.

Reachability problems can be intuitively expressed by suitable CTL*-operators, as shown in Table 2.1.

Problem	**CTL*-operator**
plain reachability	EF
recurrent reachability	EGF
constrained reachability	EU
universal reachability	AF
universal recurrent reachability	AGF
universal constrained reachability	AU

Table 2.1. Overview of reachability problems.

As a matter of fact, several pragmatical problems in the context of verification of infinite state systems can be reduced to a suitable reachability problem; notably, safety and invariant properties

(e.g. "does the system never reach a dangerous state") can be verified by solving universal reachability problems, while liveness properties (e.g. "does the system infinitely often reaches a certain good state") can be formulated as recurrent reachability problems. It is worth to remark that the universal variants of the reachability problems are reducible to the corresponding existential variants, and vice versa (for example, (I, F) is a positive instance of the universal reachability problem iff $(I, V \setminus F)$ is a negative instance of the plain reachability problem). Moreover, it should be noted that the (universal) reachability problem is a special case of the (universal) recurrent and the (universal) constrained reachability problems.

In Section 7 we describe in detail some methods to solve reachability problems over suitable classes of transition graphs.

Equivalence-checking problems. In its most common formulation, the equivalence-checking problem for transition systems is the problem of deciding, given two (representations of) transition systems, whether these systems are equivalent up to a certain relation. According to such a definition, the equivalence-checking problem is actually a family of problems, whose different instances are obtained by specifying the relation that must hold between the transition systems. The relevance of this kind of problems comes from the fact that the first of the two given systems can be viewed as a concrete implementation while the second one can be viewed as a specification (i.e., the intended behavior) of the system; in such a way, establishing the equivalence of the two systems ensures that their behavior is comparable and hence the implementation is correct. Possible notions of system equivalence are isomorphism, bisimilarity, trace equivalence[3]; moreover, also the simularity relation, which is not a graph equivalence in the proper sense, is often used as a parameter to measure similarities of systems behaviors. As happens for model-checking and reachability problems, the decidability of equivalence-checking problems depend on the considered class of structures. For instance, it turns out that the bisimilarity-checking problem for transition graphs of Petri nets and the isomorphism-checking problem for automatic graphs are undecidable. In these lecture notes, we shall not consider equivalence-checking problems, even though they have been recognized to play a fundamental role in the verification of infinite state systems.

3 Basic techniques

In this section we introduce some basic techniques that have been proposed in the literature to solve decision problems for suitable logics and meaningful classes of infinite state systems.

We first focus our attention on transformations of infinite state systems specified by logical interpretations (and their possible variants). Then, we consider more powerful transformations, precisely, tree iterations and unfoldings. Such transformations allows one to transfer definability, decidability, and complexity results among different logical theories. In particular, one can show that MSO-definable interpretations and unfoldings preserve the decidability of MSO-theories and thus they can be used to define interesting classes of decidable relational structures, starting from some basic ones. Moreover, these transformations can be used either in isolation or in combination. For instance, in [20], Caucal exploits MSO-definable interpretations and unfoldings to generate, starting from the infinite binary complete tree, a hierarchy of infinite graphs whose MSO-theories are decidable.

We preliminary recall the following definition, based on [31]. Given two signatures Σ and Σ' and two logical languages L and L' over Σ and Σ', respectively, we say that a transformation f from Σ-relational structures to Σ'-relational structures is *L-to-L'-compatible* if, given any L'-sentence ψ, one can effectively build an L-sentence $\overleftarrow{\psi}$ such that, for every Σ-relational structure S,

$$S \models \overleftarrow{\psi} \quad \text{iff} \quad f(S) \models \psi.$$

[3] Two graphs are said to be trace equivalent if the languages accepted by reading the labels along paths between specified pairs of vertices coincide.

On the grounds of such a definition, it is clear that an L-to-L'-compatible transformation f allows one to transfer the decidability of L-theories of a class $\mathcal{S}$ of Σ-relational structures to the L'-theories of the class $\mathcal{S}' = \{f(S) : S \in \mathcal{S}\}$ of Σ'-relational structures. If both L and L' are FO (resp. MSO, FO(TC), etc.) logics (over possibly different signatures), we simply call the L-to-L' compatible transformations FO-compatible (resp. MSO-comptatible, FO(TC)-compatible, etc.).

3.1 Interpretations and transductions

Let fix two signatures $\Sigma = (\mathcal{R}, k)$ and $\Sigma' = (\mathcal{R}', k')$, a Σ-relational structure $S = (V, (r^S)_{r\in\mathcal{R}})$, and a logic L in the signature Σ that uses (at least) first-order variables. For the sake of brevity, given a formula $\psi(x_1, ..., x_m)$, we write $S \vDash \psi[v_1, ..., v_m]$ to mean that ψ holds in S under the assignment v_i for x_i, for each $1 \leq i \leq m$. Formally, an *L-definable interpretation* of Σ' into Σ is a tuple

$$\mathcal{I} = \big(\phi(x), (\psi_r(x_1, ..., x_{k'(r)}))_{r\in\mathcal{R}'}\big)$$

consisting of one L-formula $\phi(x)$ plus one L-formula $\psi_r(x_1, ..., x_{k'(r)})$ for each relational symbol r of Σ'. The interpretation $\mathcal{I}$ can be viewed as a transformation that maps a Σ-relational structure S to a Σ'-relational structure S', which is defined as follows. The domain of S' is the set of all elements $v \in V$ such that $S \vDash \phi[s]$. For each relational symbol $r \in \mathcal{R}'$, $r^{S'}$ is the set of all and only the $k'(r)$-tuples of elements $(v_1, ..., v_{k'(r)})$ such that $S \vDash \psi_r[v_1, ..., v_{k'(r)}]$. Intuitively, interpretations allows one to describe a new relational structure within a given one, by exploiting suitable logical formulas with free variables. Each relation $r^{S'}$ that results from the evaluation of an L-formula $\psi_r(x_1, ..., x_{k'(r)})$ over S is said to be *L-definable* in S. For instance, the unary predicate $P = \{2n : n \in \mathbb{N}\}$ of even natural numbers is MSO-definable in $(\mathbb{N}, <)$.

We now give two application examples of the notion of logical interpretation. In the first example, we consider the set $\mathbb{N}$ of natural numbers expanded with their usual ordering $<$ and the predicate *even* of even numbers. We denote by L the relational structure $(\mathbb{N}, <, even)$ and we define a new relational structure $K = (\mathbb{Z}, \prec, neg)$, where $\prec$ denotes the standard ordering of the integers and $neg = \{(n, -n) : n \in \mathbb{Z}\}$, by means of a suitable FO-definable interpretation of L. Intuitively, positive elements of K are represented by even numbers in L, while negative elements are represented by odd numbers in L. We define the following FO-formulas:

$$\begin{aligned}
\psi_{\prec}(x, y) = {} & (even(x) \ \wedge \ even(y) \ \wedge \ x < y) \vee \\
& (\neg even(x) \ \wedge \ \neg even(y) \ \wedge \ y < x) \vee \\
& (y < x \ \wedge \ \forall\, z.\, z < x \ \rightarrow \ z = y) \\
\psi_{neg}(x, y) = {} & (x = y \ \wedge \ \neg\exists\, z.\, z < x) \vee \\
& (even(x) \ \wedge \ y < x \ \wedge \ \neg\exists\, z.\, y < z < x) \vee \\
& (\neg even(x) \ \wedge \ x < y \ \wedge \ \neg\exists\, z.\, x < z < y) \\
\phi(x) = {} & \texttt{true}.
\end{aligned}$$

Clearly, the interpretation $\mathcal{I} = \big(\phi(x), \psi_{\prec}(x, y), \psi_{neg}\big)$ maps L to (an isomorphic copy of) K.
In this second example, we show that infinite ternary complete tree T_3 can be obtained from the infinite binary complete tree T_2 via a suitable MSO-definable interpretation. We denote by E_1 and E_2 the left- and right-child relations of T_2 and we define the following MSO-formulas

$$\begin{aligned}
\psi_{F_1}(x, y) &= E_1(x, y) \\
\psi_{F_2}(x, y) &= \exists\, z.\, E_2(x, z) \ \wedge \ E_1(z, y) \\
\psi_{F_2}(x, y) &= \exists\, z.\, E_2(x, z) \ \wedge \ E_2(z, y) \\
\psi_{F}(x, y) &= \psi_{F_1}(x, y) \ \vee \ \psi_{F_2}(x, y) \ \vee \ \psi_{F_3}(x, y) \\
\phi(x) &= \exists\, X.\, x \in X \ \wedge \ \exists\, y.\, (y \in X \ \wedge \ \neg\exists\, z.\, \psi_F(z, y)) \wedge \\
&\qquad \forall\, y, z.\, ((y \in X \ \wedge \ \psi_F(y, z)) \ \rightarrow \ z \in X).
\end{aligned}$$

Intuitively, the formulas ψ_{F_i}, with $i \in \{1,2,3\}$, define the edge relations of T_3 inside T_2, the formula ψ_F defines the successor relation F of T_3, and the formula ϕ defines the domain of T_3, which is obtained as the closure of the root under the successor relation F. Thus, it is clear that the infinite ternary complete tree T_3 can be thought of as the image of T_2 under the MSO-definable interpretation $\mathcal{I} = (\phi(x), (\psi_{F_i}(x,y))_{i\in\{1,2,3\}}))$ (see Figure 3.1).

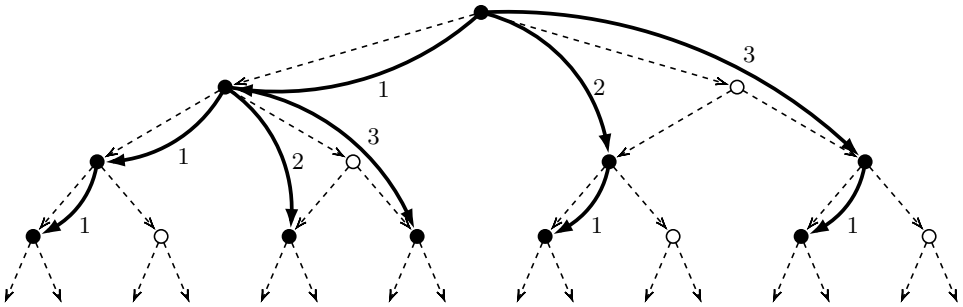

Fig. 3.1. The infinite ternary complete tree defined inside the binary one.

As regards decidability issues, it is easy to see that L-definable interpretations preserve the decidability of L-theories, namely, for every L-interpretation $\mathcal{I}$, if the L-theory of the relational structure S is decidable, then the L-theory of the image of S under $\mathcal{I}$ is decidable as well. Such a result follows from the L-to-L-compatibility of L-definable interpretations. Indeed, given an L-formula ψ in the signature Σ' and given a Σ-relational structure S, one can compute an equivalent formula $\overleftarrow{\psi}$ in the signature Σ by

i) using $\phi(x)$ (resp. $\forall\, x.\, x \in X \;\rightarrow\; \phi(x)$) to constrain each first-order variable x (resp. second-order variable X) to be instanciated by an element in the resulting structure $\mathcal{D}om(\mathcal{I}(S))$ (resp. a subset of $\mathcal{D}om(\mathcal{I}(S))$);
ii) substituting each atom of the form $r(x_1, ..., x_{k'(r)})$ with the formula $\psi_r(x_1, ..., x_{k'(r)})$.

As a matter of fact, from Büchi and Rabin theorems, we already know that the structure $L = (\mathbb{N}, <, \mathit{even})$ and the infinite binary complete tree T_2 enjoy decidable MSO-theories (notice that the predicate *even* can be defined by a suitable MSO-formula in $(\mathbb{N}, <)$). Thus, from the previous observations and examples, we immediately have that the structure $K = (\mathbb{Z}, \prec, \mathit{neg})$ and the infinite ternary complete tree T_3 enjoy decidable MSO-theories as well.

A noticeable limitation of logical interpretations (as defined above), is that, for every structure S and for every interpretation $\mathcal{I}$, $\mathcal{I}(S)$ is always a sub-structure of S. In order to overcome this clumsy situation, one can preliminarily introduce n copies of the input structure S and then apply interpretation to their disjoint union. The resulting transformation is the so-called transduction. Transductions were originally introduced in [27] for MSO-logics and transition graphs; here we adapt the definition to the case of a generic logic L and a generic relational structure S. Let $\Sigma = (\mathcal{R}, k)$ and $\Sigma' = (\mathcal{R}', k')$ be two signatures and let $[n]$ denote the set $\{1, ..., n\}$. An *L-definable transduction* from Σ to Σ' is a tuple

$$\mathcal{T} = \big(n, (\phi_i(x))_{i\in[n]}, (\psi_{r,\bar{i}}(x_1, ..., x_{k'(r)}))_{r\in\mathcal{R}', \bar{i}\in[n]^{k'(r)}}\big)$$

where $\phi_i(x)$ and $\psi_{r,\bar{i}}(x_1, ..., x_{k'(r)})$ are L-formulas. The application of $\mathcal{T}$ to a Σ-relational structure S results in an Σ'-relational structure S', which is defined as follows. The domain of S' is the set of all pairs (v, i) such that $v \in \mathcal{D}om(S)$, $1 \leq i \leq n$, and $S \vDash \phi_i[v]$. For each relational symbol $r \in \mathcal{R}'$, $r^{S'}$ is the set of all and only the $k'(r)$-tuples of elements $((v_1, i_1), ..., (v_{k'(r)}, i_{k'(r)}))$ such that $S \vDash \psi_{r,i_1,...,i_{k'(r)}}[v_1, ..., v_{k'(r)}]$. Roughly speaking, any L-definable transduction can be thought of as

an n-fold copy operation followed by an L-definable interpretation. Like L-definable interpretations, L-definable transductions preserve the decidability of L-theories.

As an example, one can easily obtain the relational structure $K = (\mathbb{Z}, \prec, neg)$ from $(\mathbb{N}, <)$ (rather than $(\mathbb{N}, <, even)$) via a stuiable FO-definable transduction.

The interested reader could also see [23], where an automaton-based characterization of bisimilarity-preserving MSO-definable transductions is given. Precisely, a bisimilarity-preserving transduction is defined as a transduction $\mathcal{T}$ such that

- $\mathcal{T}(G)$ is a deterministic rooted graph whenever G is a deterministic rooted graph,
- $\mathcal{T}(G)$ is bisimilar to $\mathcal{T}(G')$ whenever the graph G is bisimilar to the graph G'.

In [23], it has been showed that bisimilarity-preserving MSO-definable transductions can be computed by suitable forms of determistic tree transducers equipped with a rational look-ahead facility (see [38] for formal definitions).

3.2 Inverse mappings and restrictions

Inverse mappings are an alternative way to define new relational structures inside given ones. They can be applied when the input and the output signatures consist of binary relational symbols only. Thus, inverse mappings can be viewed as suitable transformations of (labeled) transition graphs. Let A and B be two alphabets for the edge labels of the input and the output graphs, respectively. We define a disjoint copy $\bar{A}$ of the alphabet A and we denote its elements with $\bar{a}$ for each $a \in A$. A *mapping* is a function h that maps a label b from B to a language L_b of finite words over $A \cup \bar{A}$. Given an A-labeled graph $G = (V, (E_a)_{a \in A})$, we expand it with the backward edge relation $E_{\bar{a}} = \{(v, v') : (v', v) \in E_a\}$ for each $a \in A$, and we allow paths to traverse edges in both directions. Note that the sequence of labels along any path in the (expanded) graph G is a word over the alphabet $A \cup \bar{A}$. A mapping $h : B \rightarrow \mathscr{P}((A \cup \bar{A})^*)$ can be applied to an A-labeled graph $G = (V, (E_a)_{a \in A})$ by its inverse, thus obtaining the B-labeled graph $h^{-1}(G) = (V, (E'_b)_{b \in B})$, where E'_b is the set of all pairs (v, v') of vertices of G such that there is a path from v to v' (traversing edges in both directions) labeled by a word $w \in h(b)$.

Mappings of the form $h : B \rightarrow \mathscr{P}((A \cup \bar{A})^*)$ can be classified on the grounds of the languages $h(b)$. For instance, if $h(b)$ is a finite language for every $b \in B$, then the mapping h is said to be *finite*. Similarly, if $h(b)$ is a rational (resp. contex-free) language for every $b \in B$, then h is said to be *rational* (resp. *context-free*). It should be also noted that inverse rational mappings are a special case of MSO-definable interpretations. Indeed, the existence of a path in a graph labeled by a word in a rational language can be expressed by a suitable MSO-formula. Thus, given any rational mapping $h : B \rightarrow \mathscr{P}((A \cup \bar{A})^*)$, one can build an MSO-formula $\psi_b(x, y)$, for each $b \in B$, that defines exactly the b-labeled edges of $h^{-1}(G)$ inside G, for every A-labeled graph G.

One can further introduce suitable operations that restrict the domain of a graph structure. For instance, vertices can be selected by specifying their access paths from a designated source vertex in a given graph. Let $G = (V, (E_a)_{a \in A})$ be a generic A-labeled graph, L be a language over $A \cup \bar{A}$, and $v_0 \in V$ a distinguished source vertex of G. The *L-restriction of G from v_0* is the graph $G|_{v_0, L} = (V', (E_a)_{a \in A})$, where

- V' is the set of all vertices $v \in V$ for which there is a path in the (expanded) graph G from v_0 to v labeled by a word $w \in L$,
- for each $a \in A$, $E'_a = E_a \cap (V' \times V')$.

In analogy to inverse mappings, we define *rational* (resp. *context-free*, etc.) restrictions as restrictions specified by rational (resp. context-free, etc.) languages. Notice that finite restrictions produce only finite graphs and thus they are of no interest. It is clear that rational restrictions from MSO-definable vertices are special cases of MSO-definable interpretations. Moreover, rational restrictions are usually applied to deterministic labeled trees with their roots as source vertices. In these cases, we can use

the shorter notation $T|_L$, instead of $T|_{v_0,L}$, which denotes the L-restriction from the root v_0 of a tree T.

We now give an example of inverse finite mapping and rational restriction. We apply these two operations to the infinite binary complete tree T_2. First, let assume that the edge labels of T_2 are taken from the set $A = \{1, 2\}$ and let $B = \{a, b, c\}$. Then, we define the finite mapping $h : B \to \mathscr{P}((A \cup \bar{A})^*)$ such that $h(a) = \{1\}$, $h(b) = \{2\}$, and $h(c) = \{\bar{2}\bar{1}2\}$ and the rational restriction $L = \{1\}^* \cdot \{\varepsilon, 2\}$. The construction of the resulting graph $G = h^{-1}(T_2)|_L$ is depicted in Figure 3.2. Since inverse finite mappings and rational restrictions are special cases of MSO-definable interpretations and since T_2 enjoys a decidable MSO-theory, we immediately have that G enjoys a decidable MSO-theory as well.

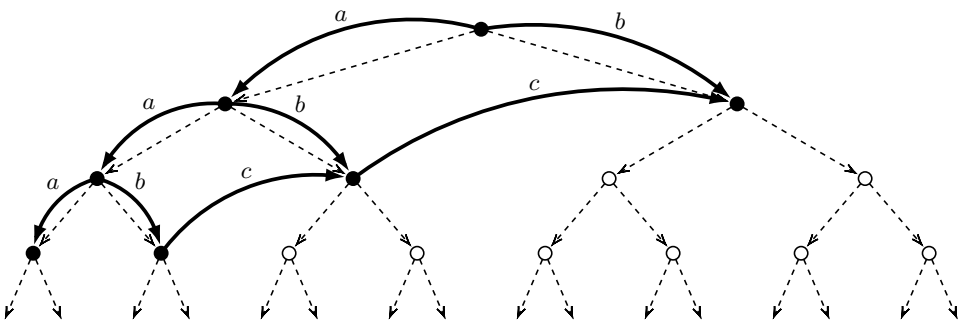

Fig. 3.2. A graph resulting from an inverse rational mapping followed by a rational restriction.

3.3 Tree iterations and unfoldings

The operation of unfolding (or unraveling), together with its variants, attracted a lot of interest in the field of infinite state system verification, since the celebrated Muchnik's Theorem was stated in [67]. The unfolding of a graph, which we formally defined in Section 2.2, is the tree whose vertices are all and only the finite paths originating from a specified source vertex and where the edge relations are implicitly given by path prolongations. According to [28], the unfolding of a graph representing a transition system is usually viewed as the *behavior* of the system. A noticeable implication of Muchnik's Theorem, whose first complete proof is due to Walukiewicz [74], is that the transformation that maps a rooted graph to its unfolding is MSO-compatible. In other words, one can translate a given (MSO-definable) property of the behavior of a system into an equi-satisfiable property of the system itself. In [28], Courcelle proved a weaker form of Muchnik's Theorem, from which it followed that the unfolding of a *deterministic* graph is MSO-compatible. Subsequently, in [73, 74], the result has been extended to a more powerful transformation (i.e., the tree iteration of a graph), thus producing the first formal proof of Muchnik's Theorem. Here, we recall the basic definitions and results.

Definition 5. *Given a graph $G = (V, (E_a)_{a \in A})$, the* tree iteration *of G is the relational structure $G^* = (V', (E'_a)_{a \in A}, son, clone)$, where*

- *V' is the set V^* of all finite (possibly empty) sequences of vertices of G,*
- *E'_a is the set of all pairs of the form $(\alpha \cdot u, \alpha \cdot v)$, with $\alpha \in V^*$ and $(u, v) \in E_a$,*
- *son is the set of all pairs of the form $(\alpha, \alpha \cdot u)$, with $\alpha \in V^*$ and $u \in V$,*
- *$clone$ is the set of all elements of the form $\alpha \cdot u \cdot u$, with $\alpha \in V^*$ and $u \in V$.*

Note that the unary predicate *clone* can be thought of as a coloring of the graph $(V', (E'_a)_{a \in A}, son)$, and hence G^* as a colored graph. Figure 3.3 depicts the graph $G = (\{l, r\}, \{(l, r)\})$ and its tree iteration G^* (dashed arrows represent instances of the relation *son* and black vertices represent elements of the predicate *clone*).

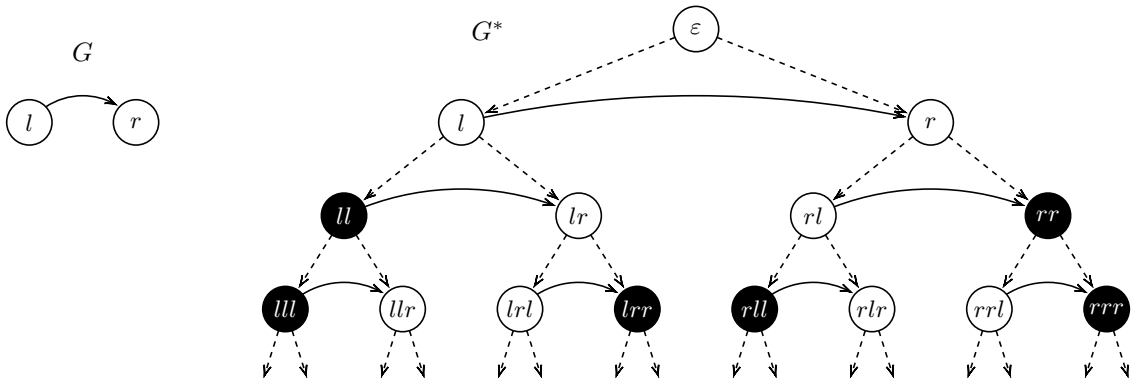

Fig. 3.3. The tree iteration of a graph.

Theorem 1 (Semenov-Muchnik [67]). *The transformation that maps a graph G to its tree iteration G^* is MSO-compatible.*

Proof. A complete proof can be found in [74]. □

Corollary 1. *The transformation that maps a rooted graph G to its unfolding $\mathcal{U}nf(G)$ is MSO-compatible.*

Proof. We prove that the unfolding of a rooted graph G can be obtained via an MSO-definable interpretation of the tree iteration of G_A, where G_A is a suitable n-fold copy of G. Let $G = (V, (E_a)_{a\in A})$ and let $\mathcal{T}$ be the MSO-definable transduction $\big((\phi_a(x))_{a\in A}, (\psi_{a,b,c}(x,y))_{a,b,c\in A}\big)$, where

- $\phi_a(x) = \texttt{true}$ for all $a \in A$,
- $\psi_{a,b,c}(x,y) = E_a(x,y)$ if $c = a$, otherwise $\psi_{a,b,c}(x,y) = \texttt{false}$.

We define $G_A = \mathcal{T}(G)$. Intuitively, G_A is obtained by making a copy (v,a) of each vertex v of G, for each label $a \in A$, and then by connecting a vertex (u,b) to a vertex (u,c) with an a-labeled edge iff $(u,v) \in E_a$ and $c = a$. It is easy to see that the unfolding of G from a vertex v_0 is isomorphic to the unfolding of G_A from the vertex (v_0,a), for any choice of $a \in A$. Moreover, the target vertex of any a-labeled edge in G_A can only be of the form (v,a), with $v \in V$. This implies that each path in G_A originating from a vertex (v_0,a_0) is uniquely determined by the sequence of its vertices (recall that graphs are assumed to be simple, namely, distinct edges have different labels or they have different source or target vertices). We can exploit such a property to define the unfolding of G_A from a vertex (v_0,a_0) inside the tree iteration G_A^* of G_A.

Let $G_A^* = (V^*(E'_a)_{a\in A}, \mathit{son}, \mathit{clone})$. Given two paths in G_A represented by the sequences $\alpha, \beta \in V'$, we have that β is an extension of α with an a-labeled edge (and hence (α,β) is a candidate a-labeled edge in $\mathcal{U}nf(G_A,(v_0,a_0))$) iff there is a sequence $\gamma \in V^*$ satisfying

i) $\gamma \in \mathit{clone}$,
ii) $(\alpha,\gamma) \in \mathit{son}$,
iii) $(\gamma,\beta) \in E'_a$.

The above conditions can be easily expressed by suitable MSO-formulas over the structure G_A^*. Moreover, one can build another MSO-formula over G_A^* that constrains a sequence α to represent a real path in G_A that originates from (v_0,a_0) (simply apply the closure of (v_0,a_0) under the previously defined relations). Thus, there is an MSO-interpretation that generates (an isomorphic copy) of $\mathcal{U}nf(G,v_0)$ starting from G_A^*. In particular, from Theorem 1 and from the MSO-compatibility of MSO-definable transductions and interpretations, we know that the unfolding operation is MSO-compatible. □

It should be also noted that Theorem 1 generalizes Rabin's Theorem, since the decidability of the MSO-theory of the infinite binary complete tree T_2 follows from the MSO-compatibility of the

unfolding operation and from the decidability of the MSO-theories of finite graphs (notice that T_2 can be obtained as the unfolding of the finite graph $G = (V, E_1, E_2)$, where $V = \{v\}$ and $E_1 = E_2 = \{(v, v)\}$). On the other hands, the proof of Muchnik's Theorem strongly relies on the closure of Rabin tree automata under union and complementation.

3.4 Caucal hierarchy

By alternating MSO-compatible transformations starting from some basic graph structures, it is possible to generate a number of interesting structures enjoying decidable MSO-theories. For instance, one can start with finite graphs, which obviously enjoy decidable MSO-theories, and then apply unfoldings to them, thus obtaining (decidable) regular trees of finite out-degree. In succession, one can apply inverse rational mappings followed by rational restrictions, thus obtaining other (decidable) graphs (the so-called prefix-recognizable graphs [19, 21]), and so on. Such an alternation of unfoldings, inverse rational mappings, and rational restrictions yields a hierarchy of decidable graphs and trees, called *Caucal hierarchy*, or pushdown hierarchy [20]. We formally define such a hierarchy as follows:

$$\begin{aligned}
\mathcal{G}_0 &= \{G : G \text{ is a finite graph}\} \\
\mathcal{T}_n &= \{\mathcal{U}nf(G, v_0) : G \in \mathcal{G}_n, v_0 \text{ is an MSO-definable vertex of } G\} \\
\mathcal{G}_{n+1} &= \{h^{-1}(T)|_L : h \text{ is a rational mapping}, L \text{ is a rational language}\}
\end{aligned}$$

From the MSO-compatibility of unfoldings, inverse rational mappings, and rational restrictions, it immediately follows that every graph in the Caucal hierarchy has a decidable MSO-theory.

Equivalent characterizations of such a hierarchy have been provided. For instance, in [72] the same hierarchy is obtained as an alternation MSO-definable interpretations and unfoldings from MSO-definable vertices and in [14] it is obtained from MSO-definable transductions and the treegraph operation (i.e., a suitable variant of the tree iteration). Moreover, always in [14], it is proved that, for every $n > 0$, the level n Caucal graphs coincide with the transition graphs of level n higher-order pushdown automata, which are automata working on level n stacks (a level 1, or simple, stack is a finite word, a level $n + 1$ stack is a stack of level n stacks, see also [33, 34, 35] for further details). From the above characterizations and from the strictness of the hierarchy of the languages recognized by higher-order pushdown automata (see [32, 40]), it follows that the Caucal hierarchy is strictly increasing as well.

Moreover, in [20] Caucal introduces a sub-hierarchy consisting of deterministic trees only. He extends the notions of unfolding, rational mapping, and rational restriction to deal with colored (ranked) trees and he defines the so-called *term hierarchy* by repeatedly applying rational markings (i.e., a variant of the notion of rational restriction), inverse *deterministic* rational mappings, and unfoldings, and then by restricting the resulting structures to be deterministic colored (ranked) trees. He also shows that the term hierarchy coincides with the *safe higher-order program schemes* [33, 53, 52] and with the hierarchy of trees obtained by iterating the evaluation of first-order substitutions starting from regular trees [30].

3.5 Reductions

In the beginning of Section 3, we introduced the notion of L-to-L'-compatible transformation. Roughly speaking, such a notion gives an *effective and uniform* method to reduce the L'-theory of the structure $f(S)$ to the L-theory of the structure S, where f is a specified transformation and S is an input structure for f. However, the notion of reducibility between logical theories can be applied in a more general setting, where transformations are not involved. Precisely, given two signatures Σ and Σ', two logics L and L', and two relational structures S and S', respectively in the signature Σ and in the signature Σ', we say that an L-formula ψ is *equivalent* to an L'-formula ψ' whenever $S \models \psi$ iff $S' \models \psi'$

(if ψ has some free variables, then we assume that S is a structure expanded with an assignment and the same for ψ' and S'). We then say that the L'-theory of S' is *reducible* to the L-theory of S if there is an effective translation of L'-formulas interpreted over S' into equivalent L-formulas interpreted over S. Note that, in such a case, if the L-theory of S is decidable, then the L'-theory of S' is decidable as well. It should be also noted that, differently from the cases of interpretations and transductions, we do not enforce any restriction on the form of the reduced formulas; in particular, the number and the types of the free variables of L'-formulas may not coincide with the number and the types of the free variables of reduced L-formulas.

We now give an example of reduction, taken from [69]. In particular, we reduce the chain fragment of the MSO-theory (shortly, the CMSO-theory) of the infinite binary complete tree T_2 expanded with the equi-level predicate to the MSO-theory of the semi-infinite line $(\mathbb{N}, <)$. We assume that the vertices of T_2 are the finite words over the set $A = \{1, 2\}$ and the edge relations are of the form $E_1 = \{(v, v \cdot 1) : v \in A^*\}$ and $E_2 = \{(v, v \cdot 2) : v \in A^*\}$. We further denote by L the equi-level predicate of T_2, which consists of all pairs of words v, v' such that $|v| = |v'|$. Recall that CMSO logic restricts second-order variables to be instanciated by chains, namely, subsets of paths consisting of $\{1, 2\}$-labeled edges only. Without loss of generality, we can assume that quantifications in CMSO logic are restricted to *non-empty* chains. We now show how to encode a generic non-empty chain C with two subsets Z_C, W_C of $\mathbb{N}$. Let C be a non-empty chain. We say that $P \subseteq A^*$ is a *cover* of C if P is a maximal path including C. We denote by P_C the *leftmost* cover of C, that is, the (unique) cover P_C such that, if v is the longest word in C, then the path P_C consists only of 1-labeled edges starting from the vertex v downward. We then define Z_C and W_C in such a way that, for every $i \in \mathbb{Z}$,

i) $n \in Z_C$ iff P_C contains a word of the form $v \cdot 2$, with $|v| = n$ (the set Z_C encodes the $\{2\}$-labeled edges along the path P_C),
ii) $n \in W_C$ iff C contains a word of length n (W_C encodes the levels which intersected by the chain C).

Notice that the pair (Z_C, W_C) determines in an unambiguous way the non-empty chain C. Furthermore, we can map the above construction in the logic. Precisely, let ψ be a CMSO-formula. For every chain variable X in ψ', we introduce two set variables Z_X and W_X (to be instanciated by subsets of $\mathbb{N}$). Then, we translate ψ' to an equivalent MSO-formula $\overrightarrow{\psi}$ as follows:

- if ψ is of the form $X \subseteq Y$, then we set $\overrightarrow{\psi}$ to be $W_X \subseteq W_Y \wedge \big(Z_X = Z_Y \vee \exists\, w \in W_X.\ (\forall\, w' \in W_X.\ w \geq w' \wedge \forall\, z \leq w.\ z \in Z_X \leftrightarrow z \in Z_Y)\big)$ (read 'W_X is included in W_Y and either $Z_X = Z_Y$ holds or W_X has a maximum element w and $Z_X \cap \{z \leq w\} = Z_Y \cap \{z \leq w\}$');
- if ψ is of the form $E_1(X, Y)$, then we set $\overrightarrow{\psi}$ to be $\exists\, w.\ Z_X = Z_Y \wedge W_X = \{w\} \wedge W_Y = \{w + 1\}$;
- if ψ is of the form $E_2(X, Y)$, then we set $\overrightarrow{\psi}$ to be $\exists\, w.\ Z_X \cup \{w + 1\} = Z_Y \wedge W_X = \{w\} \wedge W_Y = \{w + 1\}$;
- if ψ is of the form $L(X, Y)$, then we set $\overrightarrow{\psi}$ to be $\exists\, w.\ W_X = W_Y = \{w\}$;
- if ψ is of the form $\psi_1 \vee \psi_2$, then we set $\overrightarrow{\psi}$ to be $\overrightarrow{\psi}_1 \vee \overrightarrow{\psi}_2$;
- if ψ is of the form $\neg\psi'$, then we set $\overrightarrow{\psi}$ to be $\neg\overrightarrow{\psi}'$;
- if ψ is of the form $\exists\, X.\ \psi'$, then $\overrightarrow{\psi}$ is $\exists\, Z_X.\ \exists\, W_X.\ \overrightarrow{\psi}' \wedge W_X \neq \emptyset \wedge \forall\, w \in W_X.\ (\forall\, w' \in W_X.\ w \geq w') \rightarrow (\forall\, z \in Z_X.\ w \geq z)$ (read 'there are Z_X and $W_X \neq \emptyset$ satisfying $\overrightarrow{\psi}'$ and, whenever W_X has a maximum element w, then w is greater than or equal to every element of Z_X', meaning that the cover corresponding to the encoding of the chain X is the leftmost one).

It is routine to check that for every CMSO-sentence ψ, ψ holds in (A^*, E_1, E_2, L) iff $\overrightarrow{\psi}$ holds in $(\mathbb{N}, <)$. From the decidability of the MSO-theory of $(\mathbb{N}, <)$, it follows that the CMSO-theory of (A^*, E_1, E_2, L) is decidable as well.

In the literature (see, for instance, [21, 63]) one can find several examples of reductions between different logics.

4 Context-free and prefix-recognizable graphs

In this section, we introduce the context-free graphs and the prefix-recognizable graphs and we study their properties with a special emphasis on equivalent (internal and external) presentations and on the decidability of the corresponding MSO-theories.

Context-free graphs and prefix-recognizable graphs are naturally represented as transition graphs of pushdown automata and prefix-rewriting systems, respectively. However, alternative presentations based on deterministic graph grammars, more precisely, least solutions of hyperedge-replacement equations [24] and vertex-replacement equations [25], have been provided in the literature. Both approaches exploit suitable rewriting rules that transform a given hypergraph (i.e., a graph containing edges with arbitrary arities) to another hypergraph. Intuitively, an hyperedge-replacement operation consists in replacing a non-terminal hyperedge e in a hypergraph G by another hypergraph H and then gluing the sources of H with the former attachment points of e in G. A vertex-replacement operation, instead, consists in replacing a vertex v (and its incident hyperedges) in a hypergraph G by another hypergraph H and adding new hyperedges that connect H to the former neighbours of v in G. It is worth to mention that the class of finite graphs generated by repeated applications of hyperedge-replacement operations is strictly contained in the class of finite graphs generated by repeated applications of vertex-replacement operations. However, in [42, 43] it has been showed that both approaches are equally expressive for sets of simple hypergraphs of bounded in- and out-degree. Moreover, infinite (hyper)graphs can be obtained as the least solutions of deterministic systems of hyperedge-/vertex-replacement equations, or, equivalently, as the limits of hyperedge-/vertrex-replacement operations specified by deterministic graph grammars. The reader can find comparisons of the hyperedge- and vertex-replacement approaches in [29, 41].

In the following section, we formally define hyperedge-replacement grammars. We then prove that the context-free graphs can be equivalently represented as the graphs generated by hyperedge-replacement grammars. As for prefix-recognizable graphs, we only mention, without proving, that the prefix-recognizable graphs coincide (up to isomorphisms) with the graphs generated by vertex-replacement grammars.

Hyperedge-replacement grammars. Let fix a finite set A of edge labels and a finite set N of (ranked) non-terminal symbols. To each symbol $\gamma \in N$ we associate an arity k_γ. Given a non-terminal symbol $\gamma \in N$, a γ-labeled *hyperedge* is a tuple consisting of exactly k_γ vertices $v_1, ..., v_{k_\gamma}$. An $A \cup N$-labeled *hypergraph* is a structure of the form $G = (V, (E_a)_{a \in A \cup N})$, where V is a set of vertices and E_a is either a set of edges or a set of hyperedges, depending on whether $a \in A$ or $a \in N$. Intuitively, hypergraphs are natural generalizations of graphs, where edges can have arbitrary arities.

A (deterministic) *graph grammar* is a tuple $\mathcal{G} = (H_\gamma)_{\gamma \in N}$, where, for each $\gamma \in N$, H_γ is a finite hypergraph. The graph grammar $\mathcal{G}$ is used as a description of rewriting rules for non-terminal (i.e., N-labeled) hyperedges. More precisely, each hypergraph H_γ has hyperedges labeled over $A \cup N$ and it is equipped with a coloring function that associates to each vertex a color from the set $\{\bot\} \cup \{1, ..., k_\gamma\}$ and that satisfies the following condition: for each $1 \leq i \leq k_\gamma$, there is exactly one vertex $v \in \mathcal{D}om(H_\gamma)$ that is colored with i, every other vertex is colored with $\bot$. The coloring associated to each graph H_γ makes it possible to replace a γ-labeled hyperedge $(v_1, ..., v_{k_\gamma})$ in a hypergraph G with a copy of the graph H_γ by 'gluing' each vertex v_i with the corresponding copy of the i-colored vertrex in H_γ.

Before giving formal definitions of graph rewriting rules, we give an intuitive example of graph grammar. In Figure 4.4, we represented a graph grammar consisting of two rules for the non-terminal symbols γ_0 and γ_1 with arity 2 and 3, respectively. In the bottom part of the same figure, we depicted (a portion of) the graph resulting from repeated applications of rewrite rules starting from an initial γ_0-labeled hyperedge.

Below, we formally define hypergraph rewriting rules.

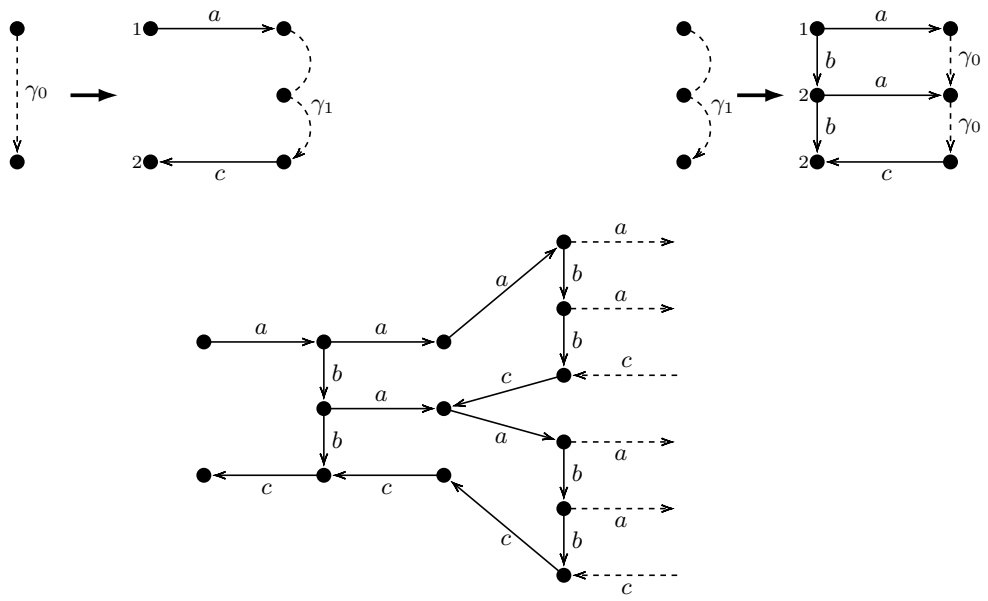

Fig. 4.4. A graph grammar and the generated graph.

Definition 6. *Given a graph grammar $\mathcal{G} = (H_\gamma)_{\gamma \in N}$, two $A \cup N$-labeled hypergraphs G and G', and a γ-labeled hyperedge $(v_1, ..., v_{k_\gamma})$ from G, with $\gamma \in N$, we say that* G' is a rewriting of G on $(v_1, ..., v_{k_\gamma})$ according to $\mathcal{G}$, *and we write $G \xrightarrow[\mathcal{G}]{(v_1,...,v_{k_\gamma})} G'$,if there is an isomorphic copy H'_γ of H_γ such that:*

- $\mathcal{D}om(G) \cap \mathcal{D}om(H'_\gamma) = \{v_1, ..., v_{k_\gamma}\}$,
- *for every $1 \leq i \leq k_\gamma$, v_i is the (unique) i-colored vertex of H'_γ,*
- $\mathcal{D}om(G') = \mathcal{D}om(G) \cup \mathcal{D}om(H'_\gamma)$,
- *for each $a \in A \cup N$, the a-labeled hyperedges of G' are the a-labeled hyperedges of G, except $(v_1, .., v_{k_\gamma})$ if $a = \gamma$,plus all a-labeled hyperedges of H_γ.*

Note that, even though $\mathcal{G}$ is a deterministic graph grammar, for each non-terminal symbol $\gamma \in N$, we can have different choices on which γ-labeled hyperedge has to be replaced (there may exist distinct hyperedges labeled with the same symbol γ). However, the rewriting relation is *confluent* in the following sense: if $(v_1, ..., v_{k_\gamma})$ and $(w_1, ..., w_{k_{\gamma'}})$ are two hyperedges of G, then there is G_1 such that $G \xrightarrow[\mathcal{G}]{(v_1,...,v_{k_\gamma})} G_1 \xrightarrow[\mathcal{G}]{(w_1,...,w_{k_{\gamma'}})} G'$ iff there is G_2 such that $G \xrightarrow[\mathcal{G}]{(w_1,...,w_{k_{\gamma'}})} G_2 \xrightarrow[\mathcal{G}]{(v_1,...,v_{k_\gamma})} G'$. Thus, it makes sense to define the *parallel rewriting of a hypergraph* G, denoted $G \underset{\mathcal{G}}{\Rightarrow} G'$, as the rewriting of all non-terminal hyperedges (in any arbitrary order) of G. Such a notion is needed to guarantee that every hyperedge in a given hypergraph is eventually replaced according to the rules in $\mathcal{G}$. Moreover, we have the following two properties:

i) if $G \underset{\mathcal{G}}{\Rightarrow} G'$ and $\underset{\mathcal{G}}{\Rightarrow} G''$ hold, then G' and G'' are isomorphic graphs,
ii) if $\preceq$ is the preordering relation between graphs such that $G \preceq G'$ iff every vertex and every A-labeled edge of G also belongs to G', then, whenever $G \underset{\mathcal{G}}{\Rightarrow} G'$ holds, $G \preceq G'$ follows.

These properties allows us to think of $\underset{\mathcal{G}}{\Rightarrow}$ as a monotone function, provided that we identify graphs whose restrictions to A-labeled edges are isomorphic. Given an initial non-terminal symbol $\gamma_0 \in N$, called *axiom*, we can denote by $\mathcal{G}^\omega(\gamma_0)$ the least fixpoint of n-fold applications of $\underset{\mathcal{G}}{\Rightarrow}$ starting from a

hypergraph consisting of a single γ_0-labeled hyperedge (note that such an A-labeled graph is unique up to isomorphisms). We call the graph $\mathcal{G}^\omega(\gamma_0)$ the *pattern graph* generated by $\mathcal{G}$ from γ_0.

4.1 Context-free graphs

Roughly speaking, a context-free graph is a connected component of an ε-free pushdown automaton. Hereafter, a *connected component* of a graph G is a maximal sub-graph G' such that, for every pair of vertices v, v' in G', v' is reachable from v by a path that can traverse edges in both directions (note that this is not the standard notion of connected component for directed graphs).

Definition 7. *Given a pushdown automaton $\mathcal{P} = (S, A, \Gamma, \delta, s_0, \gamma_0)$ (devoid of its final states), the* context-free graph *(or* accessible pushdown transition graph*) corresponding to $\mathcal{P}$ is the graph $G = (V, (E_a)_{a \in A})$, where*

- *for each $a \in A$, E_a is the set of all pairs $((s, w), (s', w'))$ such that there are $\gamma \in \Gamma$ and $u, v \in \Gamma^*$ satisfying (i) $w = \gamma \cdot v$, (ii) $(s, a, \gamma, u, s') \in \delta$, and (iii) $w' = u \cdot v$,*
- *V is the maximal set of configurations of $\mathcal{P}$ that contains the initial configuration $(s_0, \gamma_0$ and such that, for every pair of vertices $v, v' \in V$, there is a sequence $v_1, v_2, ..., v_n$ such that $v_1 = v$, $v_n = v'$, and, for all $1 \leq i < n$, v_i and v_{i+1} are adjacent vertices.*

From the above definition, it should be clear that the context-free graphs are exactly the connected components of the pushdown transition graphs, namely, the transition graphs of ε-free pushdown systems (recall that a pushdown system is a pushdown automaton devoid of its initial configuration and its final states).

As an example, consider the pushdown automaton $\mathcal{P} = (S, A, \Gamma, \delta, s_0, \gamma_0)$, where $S = \{s_0, s_1\}$, $A = \{a, b, c\}$, $\Gamma = \{\gamma_0, \gamma\}$, and δ consists of the following transitions: $(s_0, a, \gamma_0, \gamma\gamma_0, s_0)$, $(s_0, a, \gamma, \gamma\gamma, s_0)$, $(s_0, b, \gamma, \varepsilon, s_0)$, $(s_0, c, \gamma_0, \gamma_0, s_1)$, $(s_0, c, \gamma, \gamma, s_1)$, $(s_1, d, \gamma, \varepsilon, s_1)$ (see the left part of Figure 4.5). The (initial fragment of the) context-free graph of $\mathcal{P}$ is reported in the right part of Figure 4.5.

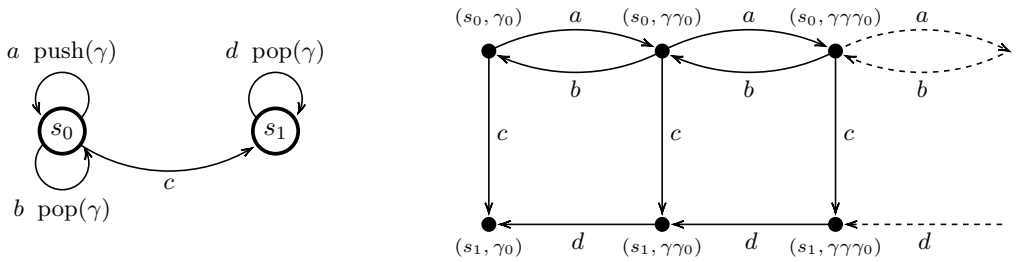

Fig. 4.5. A pushdown automaton and its context-free graph.

External presentations of context-free graphs. Context-free graphs are naturally represented by their pushdown automata. However, alternative presentations of context-free graphs have been proposed in the literature (see [24, 17, 18]). As an example, context-free graphs can be equivalently represented as the pattern graphs of special forms of graph grammars. In the following, we define special forms of pushdown automata and graph grammars.

Definition 8. *A pushdown automaton $\mathcal{P} = (S, A, \Gamma, \delta, s_0, \gamma_0)$ is said to be in* normal form *iff for every $(s, a, \gamma, w, s') \in \delta$, either $|w| \leq 1$ or $w = \gamma'\gamma$ for some $\gamma' \in \Gamma$.*

Intuitively, a pushdown automaton in normal form can change state at any time but it can only perform a push or a pop operation or change the symbol at the top of the stack. It is not difficult to show (see, for instance, [18]) that for every pushdown automaton $\mathcal{P}$, there is a pushdown automaton $\mathcal{P}'$ in normal form such that the corresponding context-free graphs are isomorphic. Roughly speaking, the construction of such a pushdown automaton $\mathcal{P}'$ is done by substituting the stack alphabet Γ of $\mathcal{P}$ with a new stack alphabet $\bigcup_{1 \leq i \leq n} \Gamma^i$ that encodes blocks of the former stack symbols having bounded length; then, the transitions of $\mathcal{P}'$ simply mimic the transitions of $\mathcal{P}$ by pushing/popping at most one block-symbol at a time.

In the following definition, given a graph grammar $\mathcal{G} = (H_\gamma)_{\gamma \in N}$ and a vertex v of some hypergraph H_γ, we say that v is an *input vertex* (resp. an *output vertex*) if v is colored with an index $i \in \{1, ..., k_\gamma\}$ (resp. v is target of an non-terminal hyperedge).

Definition 9. *A graph grammar $\mathcal{G} = (H_\gamma)_{\gamma \in N}$ is said to be* uniform *if for every $\gamma \in N$ the following conditions hold:*

- *non-terminal hyperedges in H_γ have distinct vertices,*
- *there is no input vertex in H_γ which is target of a non-terminal hyperedge (namely, the set of input vertices and the set of output vertices are disjoint),*
- *every output vertex is also a target of a terminal edge,*
- *every terminal edge in H_γ has at least one target which is an input vertex,*
- *for evrey $\gamma \in N$, the pattern graph of $\mathcal{G}$ from γ is a connected graph.*

Intuitively, the above conditions enforce that the pattern graphs generated by a uniform graph grammar are connected and have bounded in- and out-degree (note that the removal of any of such conditions may yield unconnected pattern graphs or pattern graphs with unbounded degree). It can be proved (see [18]) that for every graph grammar $\mathcal{G}$ and for every axiom $\gamma \in N$, if $\mathcal{G}^\omega(e)$ is a connected graph of finite in- and out-degree, then there exists a uniform graph grammar $\mathcal{H}$ such that $\mathcal{H}^\omega(e)$ is isomorphic to $\mathcal{G}^\omega(e)$.

The characterization of context-free graphs in terms of pattern graphs of uniform graph grammars has been first proved (in a rather different, but equivalent, framework) by Muller and Schupp [62]. Later in [17, 18] Caucal proved the same result, but in a constructive way. Below, we sketch the proof idea, by first showing that the context-free graphs are included in the pattern graphs of uniform graph grammars (up to isomorphisms), and then the converse inclusion.

Theorem 2 (Muller and Schupp [62]). *For every pushdown automaton $\mathcal{P}$ in normal form, there is a (uniform) graph grammar $\mathcal{G}$ and an axiom $\gamma_0 \in N$ such that $\mathcal{G}^\omega(\gamma_0)$ is isomorphic to the context-free graph of $\mathcal{P}$.*

Proof (sketch). We can assume that $\mathcal{P} = (S, A, \Gamma, \delta, s_0, \gamma_0)$ is in normal form and we denote by G the context-free graph of $\mathcal{P}$, where each vertex is identified with a word of the form $s \cdot w$, with $s \in S$ and $w \in \Gamma^*$. We build a graph grammar $\mathcal{G}$ that generates G by vertices of increasing length. Given a vertex u in G, we denote by G_u the connected component of G that contains u and only the vertices of length greater than or equal to $|u|$; we call the sub-graph G_u the *end* of G at vertex u (such a notion reminds the one used in [62]). We further denote by F_u the (finite) set of all vertices of G_u that have minimum length, namely, all vertices $v \in \mathcal{D}om(G_u)$ such that $|v| = |u|$; the set F_u is called *the frontier* of G_u. It is not difficult to see (cf., for instance, [18]) that, since $\mathcal{P}$ is in normal form, all vertices of G_u have a common suffix suf_u of length $|u| - 2$ (we assume that $suf_u = \varepsilon$ if $|u| < 2$). Note that $suf_u \in \Gamma^*$ for every $u \in \mathcal{D}om(G)$. Now, we define a suitable equivalence between the vertices of G: we say that two vertices $u, v \in \mathcal{D}om(G)$ are equivalent, denoted $u \equiv v$, iff the two sets $\{w \in S \cdot \Gamma^* : w \cdot suf_u \in F_u\}$ and $\{w \in S \cdot \Gamma^* : w \cdot suf_v \in F_v\}$ coincide (intuitively, the two frontiers F_u and F_v coincide after the removal of the corresponding suffixes). Such an equivalence is commonly known as the end-isomorphism equivalence. It is easy to see that if $u \equiv v$, then G_u and G_v are isomorphic graphs. Moreover, the equivalence $\equiv$ has finite index. This means that there are *only finitely many non-isomorphic ends of G*.

We now denote by U any complete set of representatives for the $\equiv$-classes. Then, for each word $u \in U$, we fix (i) an arbitrary ordering on the frontier vertices of F_u, thus writing $F_u = (f_1^{(u)}, ..., f_{h_u}^{(u)})$, and (ii) a bijective mapping ι from U to a finite set N (disjoint from A) of non-terminal symbols, thus writing $N = \{\iota(u) : u \in U\}$. We let the arity $k_{\iota(u)}$ of a non-terminal symbol $\iota(u)$ be exactly the number h_u of the frontier vertices in F_u. Then, for each representative $u \in U$ and for each label $a \in A$, we define $E_a^{(u)}$ as the set of all a-labeled edges of G_u that have at least one vertex of length exactly $|u|$ as target. We also define, for each representative $u \in U$ and for each non-terminal symbol $\gamma \in N$, the set $E_\gamma^{(u)}$ of all γ-labeled hyperedges of the form $(x_1 \cdot suf_w, ..., x_{k_\gamma} \cdot suf_w)$, where w is a vertex of G_u of length $|u|+1$, v is the representative of $[w]_\equiv$ in U, $\gamma = \iota(v)$, and $x_i \cdot suf_v = f_i^{(v)}$ for all $1 \leq i \leq k_\gamma = h_v$. We finally define the graph grammar $\mathcal{G}$ that associates to each non-terminal symbol $\gamma = \iota(u)$, with $u \in U$, the hypergraph H_γ consisting of the edge relations $(E_a^{(u)})_{a \in A \cup N}$ (the vertices are implicitly defined as the targets of these edges and their colors range over $\{\bot, 1, ..., k_\gamma\}$ and are determined by the ordering of the frontier vertices in F_u). It is routine to check that the pattern graph $\mathcal{G}^\omega(\iota(v_0))$, where v_0 is the representative of the initial configuration $s_0 \cdot \gamma_0$ in U, is isomorphic to the context-free graph G of $\mathcal{P}$. Since $\mathcal{G}$ generates a connected graph of finite degree starting from axiom $\iota(v_0)$, $\mathcal{G}$ can be transformed into an equivalent uniform graph grammar $\mathcal{G}'$. □

As an example, consider the pushdown automaton $\mathcal{P}$ depicted in Figure 4.5 and its context-free graph G. Note that $\mathcal{P}$ is already in normal form. Let $\equiv$ be the equivalence between the vertices of G induced by end-isomorphisms. It is easy to see that there are exactly 2 equivalence classes, namely, $\{s_0\gamma_0, s_1\gamma_0\}$ and $\{s_0\gamma^n\gamma_0, s_1\gamma^n\gamma_0 : n > 0\}$. Thus, we can define the set $U = \{s_0\gamma_0, s_0\gamma\gamma_0\}$ of representatives of $\equiv$-equivalence classes and let $\iota : U \to N$ be any bijection from U to the finite set of non-terminal symbols $N = \{\lambda_0, \lambda_1\}$. Finally, it turns out that the graph grammar $\mathcal{G} = (H_\lambda)_{\lambda \in N}$ represented in Figure 4.6 generates from axiom λ_0 a pattern graph isomorphic to G.

Fig. 4.6. Extraction of a graph grammar from a pushdown automaton in normal form.

Theorem 3 (Muller and Schupp [62]). *For every graph grammar $\mathcal{G}$ and for every axiom $\gamma_0 \in N$, if $\mathcal{G}^\omega(\gamma_0)$ is a connected graph of finite degree, then there is a pushdown automaton $\mathcal{P}$ whose context-free graph is isomorphic to $\mathcal{G}^\omega(\gamma_0)$.*

Proof (sketch). From the previous observations, we can assume that $\mathcal{G} = (H_\gamma)_{\gamma \in N}$ is a uniform graph grammar. Moreover, by removing useless rules and non-terminal symbols, by properly renaming vertices, and by possibly adding new rules, we can assume, without loss of generality, that that $\mathcal{G}$ and γ_0 satisfy the following properties:

i) for every axiom $\gamma \in N$, the pattern graph $\mathcal{G}^\omega(e)$ is non-empty,
ii) every non-terminal $\gamma \in N$ labels at least one hyperedge in some hypergraph generated by $\mathcal{G}$ starting from axiom γ_0,
iii) all graphs in $\mathcal{G}$ have separate output vertices, namely, every output vertex in a graph H_γ is target of a unique hyperedge,
iv) hyperedges in each graph H_γ have distinct labels, namely, for every pair of non-terminal symbols $\gamma, \gamma' \in N$, H_γ contains at most one γ'-labeled hyperedge,

v) all graphs in $\mathcal{G}$ have separate vertices, namely, for every pair of non-terminal symbols $\gamma, \gamma' \in N$, $\mathcal{D}om(H_\gamma) \cap \mathcal{D}om(H_{\gamma'}) = \emptyset$,

vi) $k_{\gamma_0} = 1$, namely, the axiom γ_0 has arity 1.

We denote by V the set of all vertices of hypergraphs of $\mathcal{G}$, namely, $V = \bigcup_{\gamma \in N} \mathcal{D}om(H_\gamma)$. Now, we fix a non-terminal symbol $\gamma \in N$ and we define the function τ_γ from $\mathcal{D}om(H_\gamma)$ to $V \cup (V \cdot N)$ as follows. For every vertex v of H_γ, if v is is target of A-labeled edges only, then we set $\tau_\gamma(v) = v$; otherwise, if v is the i-th target of a (unique) γ'-labeled hyperedge $(v_1, ..., v_{k_\gamma})$ (namely, $v = v_i$), then we set $\tau_\gamma(v) = v_i \cdot \gamma'$. Note that, since the graphs of $\mathcal{G}$ have separate outputs, the function τ_γ is well defined. Moreover, note that the images of τ_γ can be viewed as words over the alphabet $\Gamma = V \cup N$.

We now define a prefix rewriting system $\mathcal{R} = (\Gamma, A, P)$, where P consists of all rules of the forms $(\tau_\gamma(v) \cdot \gamma, a, \tau_\gamma(v') \cdot \gamma)$, where $\gamma \in N$ and (v, v') is an a-labeled edge in H_γ. We further let v_0 be the (unique) vertex of H_{γ_0} which is colored by 1 (i.e., the vertex that corresponds to the target of a γ_0-labeled hyperedge during an application of a rewriting rule). The connected component of the transition graph of $\mathcal{R}$ that includes the word $v_0 \cdot \gamma_0$ is easily shown to be isomorphic to $\mathcal{G}^\omega(\gamma_0)$. Finally, it is not difficult to transform the prefix rewriting system $\mathcal{R}$ into a pushdown automaton $\mathcal{P}$ whose context-free graph is isomorphic to $\mathcal{G}^\omega(\gamma_0)$. □

We now show that context-free graphs can be obtained from infinite complete trees via *inverse finite mappings* and *rational restrictions*.

Theorem 4 (Caucal [19]). *Let $\mathcal{P} = (S, A, \Gamma, \delta, s_0, \gamma_0)$ be a pushdown automaton. The context-free graph of $\mathcal{P}$ can be obtained from an infinite $|S| + |\Gamma|$-ary complete tree by applying an inverse finite mapping followed by a rational restriction.*

Proof. Let $\mathcal{P} = (S, A, \Gamma, \delta, s_0, \gamma_0)$ be a pushdown automaton, let $G = (V, (E_a)_{a \in A})$ be its context-free graph, and let $T_{S \cup \Gamma}$ be the infinite $S \cup \Gamma$-labeled complete tree. As usual, we identify the vertices of the tree $T_{S \cup \Gamma}$ by words over $A \cup \Gamma$ (recall that ε denotes the root of the tree). We then encode the configurations of $\mathcal{P}$ by words over $S \cup \Gamma$. In such a case, it is convenient to represent a configuration $(s, w) \in S \times \Gamma^*$ in its reversed order, namely, by the word $w^{\mathtt{rev}} \cdot s$, where $w^{\mathtt{rev}}$ denotes the reversal of the stack content w. In such a way, $\mathcal{P}$-configurations can be viewed as a vertices of $T_{S \cup \Gamma}$ and the transitions of $\mathcal{P}$ can modeled by the suffix-rewriting system $\mathcal{R} = (S \cup \Gamma, A, P)$, where

$$P = \{(\{\gamma \cdot s\}, a, \{u^{\mathtt{rev}} \cdot s'\}) \,:\, (s, a, \gamma, u, s' \in \delta\}.$$

Note that the context-free graph G coincides (up to isomorphisms) with the transition graph G' of $\mathcal{R}$ restricted to those vertices which are reachable from $\gamma_0 s_0$ via paths that traverse edges in both directions. Moreover, it easy to see that the transition graph G' of $\mathcal{R}$ can be obtained from $T_{S \cup \Gamma}$ by applying the *inverse finite mapping* h^{-1}, where, for each $a \in A$,

$$h(a) = \{\bar{s} \cdot \bar{\gamma} \cdot u^{\mathtt{rev}} \cdot s' \,:\, (\{\gamma \cdot s\}, a, \{u^{\mathtt{rev}} \cdot s'\}) \in P\}.$$

It remains to show that the context-free graph G can be obtained from G' by restricting its domain to a suitable *regular* language. To do that, we define the relation $\rightsquigarrow$ between words over $S \cup \Gamma \cup \bar{S} \cup \bar{\Gamma}$ such that $w \rightsquigarrow w'$ if w' can be obtained from w by erasing one substring of the form $x \cdot \bar{x}$, for $x \in S \cup \Gamma$. We then denote by $\rightsquigarrow^*$ the reflexive and transitive closure of $\rightsquigarrow$ and, given any language L over $S \cup \Gamma \cup \bar{S} \cup \bar{\Gamma}$, we define $\rightsquigarrow^*(L) = \{w' \in (S \cup \Gamma \cup \bar{S} \cup \bar{\Gamma})^* \,:\, \exists\, w \in L.\, w \rightsquigarrow^* w'\}$. By exploiting finite-state automata, it is easy to prove that, if L is a regular language, then $\rightsquigarrow^*(L)$ is a regular language as well. In particular, if we let

$$L = \bigcup_{a \in A} \big(h(a) \cup \overline{h(a)}^{\mathtt{rev}}\big)$$

(here $\overline{h(a)}$ denotes the regular language $\{\bar{w} \,:\, w \in h(a)\}$), then the language

$$L' = \rightsquigarrow^* (\{\gamma_0 s_0\} \cdot L^*) \cap (\Gamma^* \cdot S)$$

is regular as well. It is easy to check that the graph G is isomorphic to the L'-restriction of G'. □

Recall that inverse finite mappings are special forms of MSO-definable interpretations and hence they are MSO-compatible. Therefore, as a corollary of Theorem 4, we obtain that every context-free graph has a decidable MSO-theory. Here, we give an alternative proof of such a corollary by directly exploiting MSO-definable interpretations (see also [72]).

Corollary 2 (Muller and Schupp [62]). *The class of context-free graphs enjoy decidable MSO-theories.*

Proof. Let $\mathcal{P} = (S, A, \Gamma, \delta, s_0, \gamma_0)$ be a pushdown automaton and $G = (V, (E_a)_{a \in A})$ its context-free graph. We provide an MSO-definable interpretation of G into the infinite $S \cup \Gamma$-labeled complete tree $T_{S \cup \Gamma}$: the decidability of the MSO-theory of G will follows immediately from Rabin's Theorem and from the MSO-compatibility of MSO-definable interpretations. By definition, every configuration of $\mathcal{P}$ is a pair of the form (s, w), with $s \in S$ and $w \in \Gamma^*$. We represent the configuration (s, w) by the (unique) vertex in the tree $T_{S \cup \Gamma}$ that is reachable from the root via a path labeled by $w^{\mathtt{rev}} \cdot s$. An MSO-definable interpretation of G in $T_{S \cup \Gamma}$ can be defined as follows. Every edge relation E_a can be defined by a suitable MSO-formula $\psi_a(x, y)$ that states that there exist a vertex w in $T_{S \cup \Gamma}$ and a transition (s, a, γ, u, s') of $\mathcal{P}$ such that

i) w is accessible from the root via a path labeled over Γ,
ii) x denotes the target of a $\gamma \cdot s$-labeled path from w,
iii) y denotes the target of an $u^{\mathtt{rev}} \cdot s'$-labeled path from w.

The set of vertices of G consists of all and only the configurations reachable from the initial one (s_0, γ_0) via paths that traverse edges in both directions. Thus, it can be defined by a suitable MSO-formula $\phi(x)$ which states that for every set X, if X contains the word $\gamma_0 s_0$ and it is closed under the relations E_a and $E_a^{-1} = \{(v, u) : (u, v) \in E_a\}$, for all $a \in A$, then X contains x. □

Below, we prove a converse inclusion which implies that the graphs obtained by applying inverse finite mappings to infinite complete trees and then restricting to accessible vertices are exactly the context-free graphs.

Theorem 5 (Caucal [19]). *Let T_A be the infinite A-labeled complete tree, v_0 a vertex of T_A, and $h : B \to \mathscr{P}((A \cup \bar{A})^*)$ a finite mapping. The graph obtained by restricting $h^{-1}(T_A)$ to the vertices that are accessible from v_0 via paths that traverse edges in both directions is a context-free graph.*

Proof (sketch). Since the inverse finite mapping h^{-1} is applied to the deterministic tree T_A, we can assume, without loss of generality, that every word in $h(b)$, for any $b \in B$, belongs to the language $\bar{A}^* \cdot A^*$. Indeed, if this is not the case, then we can substitute h with the equivalent finite mapping h' such that, for every $b \in B$,

$$h'(b) = \rightsquigarrow^* (h(b)) \cap (\bar{A}^* \cdot A^*)$$

(recall that $\rightsquigarrow$ is the relation such that $w \rightsquigarrow w'$ iff w' is obtained from w by erasing one substring of the form $a \cdot \bar{a}$, for $a \in A$. Under such an assumption, we can define the prefix-rewriting system $\mathcal{R} = (A, B, P)$, where $P = \{(\{u\}, b, \{v\}) : b \in B, \bar{u}^{\mathtt{rev}} \cdot v \in h(b)\}$. It is immediate to see that the transition graph of $\mathcal{R}$ coincides with $h^{-1}(T_A)$. Moreover, one can easily translate $\mathcal{R}$ into a pushdown automaton $\mathcal{P}$ whose context-free graph is isomorphic to the restriction of $h^{-1}(T_A)$ to the vertices accessible from v_0 via paths that traverse edges in both directions. □

4.2 Prefix-recognizable graphs

In the previous section we defined the context-free graphs as the transitions graphs of accessible configurations of pushdown automata. We then proved that context-free graphs can be obtained from infinite complete trees via inverse finite mappings and rational restrictions. In Section 2.3, we also remarked that pushdown systems are a proper subclass of prefix-rewriting systems. This naturally leads to the definition of prefix-recognizable graphs [19, 21], which are rational restrictions of the

transition graphs of prefix-rewriting systems [4]. It turns out that prefix-recognizable graphs strictly include the context-free graphs of Muller and Schupp and the regular graph of Courcelle (i.e., pattern graphs of unrestricted hyperedge-replacement graph grammars). Moreover, prefix-recognizable graph can be obtained from the infinite binary complete tree via inverse *rational* mappings and rational restrictions and they turn out to be the largest class of graphs that enjoy decidable MSO-theories provable by MSO-definable interpretations in the infinite binary tree.

Definition 10. *Given a prefix-rewriting system $\mathcal{R} = (\Gamma, A, P)$ and a regular language $L \subseteq \Gamma^*$, the corresponding* prefix-recognizable graph *is the graph $G = (L, (E_a)_{a \in A})$, where, for each $a \in A$, E_a is the set of all pairs $(u \cdot w, v \cdot w) \in L \times L$ for which there is a rule $(U, a, V) \in P$ satisfying $u \in U$ and $v \in V$.*

Unlike pushdown systems, prefix-rewriting systems are devoid of control states and they allow one to rewrite a word (on 'the top of the stack') rather than a single letter. Moreover, any rule (U, a, V) gives rise to an in general infinite set of rewrite triples (u, a, v), with $u \in U$ and $v \in V$ (recall that both U and V are possibly infinite languages recognized by finite-state automata). It is immediate to see that prefix-recognizable graphs may present vertices with infinite in- and out-degree. Consider, for instance, the prefix-rewriting system $\mathcal{R} = (\Gamma, A, P)$, where $\Gamma = \{X\}$, $A = \{<\}$, and P consists of a single rule $(\{\varepsilon\}, <, \{X\}^+)$; the prefix-recognizable graph of $\mathcal{R}$, depicted in Figure 4.7, is an isomorphic copy of $(\mathbb{N}, <)$ and it has vertices with infinite out-degree and unbounded in-degree.

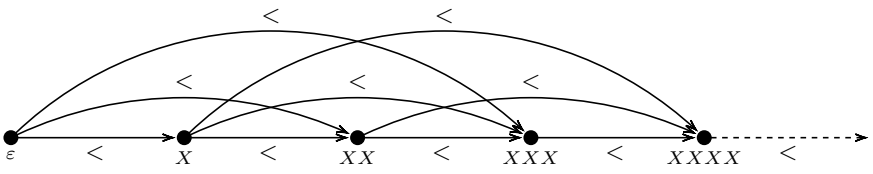

Fig. 4.7. A prefix-recognizable graph.

External presentations of prefix-recognizable graphs. We start by showing, in analogy to Theorem 4, that prefix-recognizable graphs are obtained from infinite complete trees (hence from the infinite binary complete tree) via inverse rational mappings.

Theorem 6 (Caucal [19]). *Let $\mathcal{R} = (\Gamma, A, P)$ be a prefix-rewriting system and let $L \subseteq \Gamma^*$ be a regular language. The prefix-recognizable graph of $\mathcal{R}$ and L can be obtained from an infinite $|\Gamma|$-ary complete tree by applying an inverse rational mapping and a rational restriction.*

Proof. The proof is a straightforward generalization of Theorem 4. Let $\mathcal{R} = (\Gamma, A, P)$ be a prefix-rewriting system, $L \subseteq \Gamma^*$ a regular language, and $G = (L, (E_a)_{a \in A})$ the corresponding prefix-recognizable graph. We denote by T_Γ be the infinite Γ-labeled complete tree and we identify its vertices by words over Γ. We represent a configuration w of $\mathcal{R}$ by the reversed word $w^{\mathtt{rev}} \in L^{\mathtt{rev}}$. It is immediate to see that the prefix-recognizable graph G can be obtained from T_Γ by applying the inverse rational mapping h^{-1}, where, for each $a \in A$,

$$h(a) = \{\bar{u} \cdot v^{\mathtt{rev}} \,:\, \exists\, (U, a, V) \in P.\ u \in U, v \in V\}$$

followed by the $L^{\mathtt{rev}}$-restriction (note that $L^{\mathtt{rev}}$ is a regular language). □

[4] Actually, in their original formulation, the prefix-recognizable graphs were defined as the graphs resulting from the infinite binary complete tree via inverse rational mappings and rational restrictions and then they were showed to coincide with the rational restrictions of the transition graphs of prefix-rewriting systems.

Since inverse rational mappings are special cases of MSO-definable interpretations, Theorem 6 immediately implies every prefix-recognizable graph has a decidable MSO-theory. Here, we give an alternative proof of such a result by directly exploiting MSO-definable interpretations.

Corollary 3 (Muller and Schupp [62]). *The class of prefix-recognizable graphs enjoy decidable MSO-theories.*

Proof. The proof generalizes that of Corollary 2. Let $\mathcal{R} = (\Gamma, A, P)$ be a prefix-rewriting system, $L \subseteq \Gamma^*$ a regular language, and $G = (L, (E_a)_{a \in A})$ the corresponding prefix-recognizable graph. We show how to obtain G from the infinite Γ-labeled complete tree T_Γ via an MSO-definable interpretation. The edge relation induced by a rule $(U, a, V) \in P$ can be defined by a suitable MSO-formula $\psi_{U,a,V}(x, y)$ that states that there exist a vertex w in T_Γ and two words $u, v \in \Gamma^*$ such that
i) $u \in U^{\mathtt{rev}}$,
ii) $v \in V^{\mathtt{rev}}$,
iii) x denotes the target of an u-labeled path from w,
iv) y denotes the target of a v-labeled path from w
(note that the first two conditions can be expressed in terms of existence of successful runs of the automata that recognize the regular languages $U^{\mathtt{rev}}$ and $V^{\mathtt{rev}}$). The disjunction of the formulas $\psi_{U,a,V}(x, y)$, one for each a-labeled rule (U, a, V) of the prefix-rewriting system, is a formula that defines the edge relation E_a of G. Finally, the MSO-formula $\phi(x)$ defining the domain L of G is obtained by constraining the variable x to be instanciated by those vertices in T_Γ that are accessible from the root via paths labeled by words in L (this is done by expressing the existence of successful runs for the automaton recognizing the regular language L). □

The converse of Theorem 6 also holds.

Theorem 7 (Caucal [19]). *Let T_A be the infinite A-labeled complete tree, let $h : B \to \mathscr{P}((A \cup \bar{A})^*)$ be a rational mapping, and let $L \subseteq A^*$ be a regular language. The graph $h^{-1}(T_A)|_L$ is a prefix-recognizable graph.*

Proof (sketch). The proof is almost identical to that of Theorem 5. Since the inverse rational mapping h^{-1} is applied to the deterministic tree T_A, we can assume, without loss of generality, that every word in $h(b)$, for any $b \in B$, belongs to the language $\bar{A}^* \cdot A^*$. Under such an assumption, we can define the prefix-rewriting system $\mathcal{R} = (A, B, P)$, where $P = \{(f(h(b)), b, g(h(b))^{\mathtt{rev}}) : b \in B\}$ and f (resp. g) is the natural extension to the languages of the function that erases all symbols from A (resp. from $\bar{A}$) in a given word (notice that both f and g preserve regularity of languages). It is immediate to see that the $L^{\mathtt{rev}}$-restriction of the transition graph of $\mathcal{R}$ coincides with $h^{-1}(T_A)|_L$. □

As for graph grammars, in [39, 26] it has been proved that the graphs obtained via vertex-replacement operations can be also obtained from the infinite binary tree via special forms of MSO-definable interpretations, which are very similar to inverse rational mappings followed by rational restrictions. On the grounds of the previous results, this suggests the fact, proved in [4], that the prefix-recognizable graphs are exactly the graphs specified by deterministic graph grammars written with vertex-replacement operations. Since hyperedge- and vertex-replacement operations are equally expressive for sets of simple hypergraphs of bounded in- and out-degree [42, 43], this also shows that the connected components of prefix-recognizable graphs of bounded degree are exactly the context-free graphs.

Finally, it is possible to show that the prefix-recognizable graphs are exactly the rational restrictions of ε-closures of transition graphs of pushdown systems with ε-moves.

Theorem 8 (Stirling [68]). *The prefix-recognizable graphs are isomorphic to the rational restrictions of ε-closures of transition graphs of pushdown systems with ε-moves.*

Proof (sketch). We first prove that any rational restriction of the ε-closure of the transition graph of a pushdown system $\mathcal{P} = (S, A \cup \{\varepsilon\}, \Gamma, \delta)$ is a prefix-recognizable graph. Let G be the transition graph of $\mathcal{P}$ and let G' be the $A \cup \{\#\}$-labeled graph obtained from G by replacing every ε-labeled edge by a corresponding $\#$-labeled edges, where $\#$ is a fresh symbol. Since $\mathcal{P}$ is a special case of prefix-rewriting system, G' can be obtained from the infinite binary complete tree T_2 via an inverse rational mapping h^{-1}. Moreover, the ε-closure of G is definable in G' via another inverse rational mapping j^{-1} such that $j(a) = \{a\} \cdot \{\#\}^*$ for all $a \in A$. Since, the functional composition $j^{-1} \circ h^{-1}$ of two inverse rational mappings is an inverse rational mapping as well, this shows that, for every regular language $L \subseteq \Gamma^*$, $G' = (j^{-1} \circ h^{-1})(T_2)|_L$ is a prefix-recognizable graph.
As for the converse inclusion, let $\mathcal{R} = (\Gamma, A, P)$ be a prefix-rewriting system, where P consists of the following rewriting rules (U_1, a_1, V_1), ..., (U_n, a_n, V_n), and let $L \subseteq \Gamma^*$ be a regular language. For each $1 \leq i \leq n$, we denote by $\mathcal{A}_i = (S_i, \Gamma, \delta_i, \mathcal{I}_i, \mathcal{F}_i)$ and by $\mathcal{A}'_i = (S'_i, \Gamma, \delta'_i, \mathcal{I}'_i, \mathcal{F}'_i)$ the finite-state automata recognizing U_i and $(V_i)^{\mathtt{rev}}$ respectively (note that both are regular languages). Without loss of generality, we can assume that these automata have pairwise disjoint sets of states. We denote by $q_\#$ a fresh state not belonging to any set S_i or S'_i. Then, we define the pushdown system $\mathcal{P} = (Q \cup \{q_\#\}, A \cup \{\varepsilon\}, \Gamma, \delta)$ such that

- $Q = \bigcup_{1 \leq i \leq n} (S_i \cup S'_i)$,
- δ consists of transitions of the following forms

$$\begin{array}{ll}
(q_\#, \varepsilon, \gamma, \gamma, s) & \text{if } s \in \mathcal{I}_i \text{ and } \gamma \in \Gamma \\
(s, \varepsilon, \gamma, \varepsilon, s') & \text{if } (s, \gamma, s') \in \delta_i \\
(s, \varepsilon, \gamma, \gamma, s') & \text{if } s \in \mathcal{F}_i, s' \in \mathcal{I}'_i, \text{ and } \gamma \in \Gamma \\
(s, \varepsilon, \gamma, \gamma'\gamma, s') & \text{if } (s, \gamma', s') \in \delta'_i \text{ and } \gamma \in \Gamma \\
(s, a_i, \gamma, \gamma, q_\#) & \text{if } s \in \mathcal{I}'_i \text{ and } \gamma \in \Gamma.
\end{array}$$

It is routine to check that the L'-restriction, where $L' = \{q_\#\} \cdot L$, of the ε-closure of the transition graph of $\mathcal{P}$ is isomorphic to the prefix-recognizable graph of $\mathcal{P}$ and L. □

5 The contraction method

In this section we study the model-checking problem for MSO logics interpreted over the semi-infinite line (i.e., $(\mathbb{N}, <)$) and over branching structures (e.g., the infinite complete binary tree) expanded with *unary* predicates. The model-checking problem is tackled by reducing it to the acceptance problem for sequential Büchi automata and Rabin tree automata, respectively.

In the following, we start by considering expansions of $(\mathbb{N}, <)$ by unary predicates. As for expansions of $(\mathbb{N}, <)$ by *binary* predicates, it is worth to remark that undecidability arises already in very simple cases, like for instance the relation $\{(n, 2n) : n \in \mathbb{N}\}$). Moreover, according to Seese's conjecture [66], positive results for the decidability of the MSO-theories of this kind of structures can always be obtained by using MSO-definable interpretations over tree structures with decidable MSO-theories. This last remark gives a motivation for studying, in Section 5.2, the decidability of the model-checking problem for MSO logics interpreted over tree structures.

5.1 The case of the semi-infinite line

We first reduce the model-checking problem for MSO logic interpreted over an expanded structure $(\mathbb{N}, <, P)$, where $P \subseteq \mathbb{N}$, to the acceptance problem for sequential Büchi automata. Precisely, given an MSO-formula $\psi(X)$, we know from Büchi's Theorem [6] how to compute a Büchi automaton $\mathcal{A}$ such that $(\mathbb{N}, <) \models \psi[P]$ iff $\mathcal{A}$ accepts the characteristic word $w_P \in \{0, 1\}^\omega$ associated to P. The problem of deciding whether a given automaton $\mathcal{A}$ accepts a fixed word w is called *acceptance problem* and it is denoted by Acc_w.

We just saw that the model-checking problem for an expanded structure $(\mathbb{N}, <, P)$ is reducible to the problem Acc_{w_P}. Moreover, the latter problem is easily seen to be decidable if w_P is an *ultimately periodic word*. In [36] Elgot and Rabin found several interesting predicates P for which w_P is not ultimately periodic and Acc_{w_P} is still decidable, notably the predicate $\{n! : n \in \mathbb{N}\}$ of the factorial numbers and the predicate $\{n^k : n \in \mathbb{N}\}$ of the k-th powers. Their approach was based on a reduction of Acc_w to the case of ultimately periodic words. Given P as one of the predicates mentioned above, they showed that, for any Büchi automaton $\mathcal{A}$, contiguous repetitions of the symbol 0 in w_P can be *contracted* in such a way that an ultimately periodic word w' is obtained which is accepted by $\mathcal{A}$ iff w_P is accepted by $\mathcal{A}$. In [15, 16], Carton and Thomas generalized such a method to the class of the so-called residually (or profinitely) ultimately periodic predicates, which includes morphic predicates (e.g., Thue-Morse word predicate, Fibonacci numbers) as well as non-morphic ones (e.g., factorial numbers). Later in [65], Rabinovich showed that the class of residually ultimately periodic predicates is the largest class of unary predicates P for which the MSO-theory of $(\mathbb{N}, <, P)$ is decidable. In the following, we describe in detail Carton and Thomas's approach.

We start with some preliminary definitions. A (possibly infinite) set S endowed with an internal binary operation $\cdot$ is said to be a *semigroup* if $\cdot$ is associative, namely, for every $a, b, c \in S$, $a \cdot (b \cdot c) = (a \cdot b) \cdot c$. A *monoid* is a semigroup $(S, \cdot)$ that contains the identity element (i.e., the element 1_S such that, for every $a \in S$, $a \cdot 1_S = 1_S \cdot a = a$). As an example, the set A^* of all finite words over the alphabet A equipped with the concatenation operator $\cdot$ is a monoid (its identity element is the empty word ε). A *morphism* from a semigroup (resp., a monoid) $(S, \cdot)$ into a semigroup (resp., monoid) $(S', \odot)$ is any function μ from S to S' such that, for every $a, b \in S$, $\mu(a \cdot b) = \mu(a) \odot \mu(b)$ (resp., $\mu(a \cdot b) = \mu(a) \odot \mu(b)$ and $\mu(1_S) = 1_{S'}$). Note that a morphism from a semigroup/monoid $(S, \cdot)$ into a semigroup/monoid $(S', \cdot)$ is completely determined by the images of the individual elements of S.

Definition 11. *An infinite sequence $(u_n)_{n \geq 0}$ of finite words on alphabet A is said to be* residually ultimately periodic *if for any morphism μ from the semigroup $(A^*, \cdot)$ into a finite semigroup $(S, \odot)$, the sequence $(\mu(u_n))_{n \geq 0}$ is ultimately periodic.*

This property is said to be *effective* iff one can compute, for any given morphism μ from the semigroup $(A^*, \cdot)$ into a finite semigroup $(S, \odot)$, two integers p and q such that for every $n \geq p$, $\mu(u_n) = \mu(u_{n+q})$. An infinite word w is called *effectively residually ultimately periodic* if we can provide a factorization of w of the form $u_0 u_1 u_2 ...$ such that $(u_n)_{n \geq 0}$ is effectively residually ultimately periodic.

As an example, the sequence $(a^{n!})_{n \geq 0}$, where a is a fixed symbol, is effectively residually ultimately periodic. Indeed, the sequence can be easily shown to be effectively residually ultimately *constant*, since it is well known that for any element a of a finite semigroup $(S, \cdot)$ and any integer $n > |S|$, $a^{n!}$ is equal to a fixed element, usually denoted a^ω. A slight variant of the previous example shows that the sequence $(u_n)_{n \geq 0}$, where $u_n = 0^{nn!-1} \cdot 1$ is also effectively residually ultimately constant. Since $(n+1)! - n! - 1 = nn! - 1$, the word $w = u_0 u_1 u_2 ...$ is the characteristic word of the factorial predicate $P = \{n! : n \in \mathbb{N}\}$ and it is also effectively residually ultimately periodic.

Indistinguishability of words. We can exploit the standard relation $\sim_{\mathcal{A}}$ of *indistinguishability* for a sequential Büchi automaton $\mathcal{A}$ to reduce an instance of the acceptance problem over a non-ultimately periodic word to an equivalent instance over an ultimately periodic word.

The relation $\sim_{\mathcal{A}}$ groups finite words on which $\mathcal{A}$ behaves in a similar way. Precisely, it is defined as follows: for every pair of words $u, v \in A^*$, $u \sim_{\mathcal{A}} v$ if for every pair of states p, q of $\mathcal{A}$, there is a run of $\mathcal{A}$ from p to q reading u (and visiting a final state) iff there is a run of $\mathcal{A}$ from p to q reading v (and visiting a final state).

The relation $\sim_{\mathcal{A}}$ is an equivalence of finite index (it has at most 3^{n^2} different equivalence classes, where n is the number of states of $\mathcal{A}$) and it is compatible with the operation of concatenation, namely, if $u_0, u_1, u_2, ...$ and $v_0, v_1, v_2, ...$ are two sequences of words such that $u_i \sim_{\mathcal{A}} v_i$ for all $i \in \mathbb{N}$, then $w = u_0 u_1 u_2 ... \sim_{\mathcal{A}} v_0 v_1 v_2 ... = w'$. Intuitively, this means that, given any Büchi automaton $\mathcal{A}$ and

any infinite word w, we can replace a substring u of w by a $\sim_{\mathcal{A}}$-equivalent one u' and obtain an infinite word w' which is indistinguishable from w, namely, $w \in \mathscr{L}(\mathcal{A})$ iff $w' \in \mathscr{L}(\mathcal{A})$. By repeatedly applying substitutions of this form, one can reduce the acceptance problem for a non-ultimately periodic word to the acceptance problem for an ultimately periodic one. As a consequence, the acceptance problem for effectively residually ultimately periodic words (e.g., the characteristic word of the factorial predicate) turns out to be decidable. Such a result is stated and proved formally in the sequel.

Proposition 1. *Given a sequential Büchi automaton $\mathcal{A}$ one can compute another automaton $\mathcal{A}'$ such that for any infinite word $w = u_0u_1u_2...$ $w \in \mathscr{L}(\mathcal{A})$ iff $w' \in \mathscr{L}(\mathcal{A}')$, where $w' = [u_0]_{\sim_A}[u_1]_{\sim_A}[u_2]_{\sim_A}...$ and each $[u_i]_{\sim_A}$ is the $\sim_{\mathcal{A}}$-equivalence class of u_i (since $\sim_{\mathcal{A}}$ has finite index, w' can be thought of as an infinite word).*

Proof. Let $\mathcal{A} = (S, A, \delta, I, F)$ be a sequential Büchi automaton. The automaton $\mathcal{A}' = (S', A', \delta', I', F')$ has two distinct copies, $\bar{q}$ and $\widetilde{q}$, of each state q of $\mathcal{A}$, namely, $S' = \bar{S} \cup \widetilde{S}$. The input alphabet of $\mathcal{A}'$ is the set of all $\sim_{\mathcal{A}}$-equivalence classes, which we know to be finite. Then, for any $\sim_{\mathcal{A}}$-equivalence class C, we let

1. $(\bar{p}, C, \bar{q}) \in \delta'$ and $(\widetilde{p}, C, \bar{q}) \in \delta'$ iff $\mathcal{A}$ can go from p to q by reading any word in C and visiting at least one final state;
2. $(\bar{p}, C, \widetilde{q}) \in \delta'$ and $(\widetilde{p}, C, \widetilde{q}) \in \delta'$ iff $\mathcal{A}$ can go from p to q by reading any word in C and visiting no final state.

The initial states of $\mathcal{A}'$ are all and only the states $\widetilde{q}$ with $q \in I$. The final states of $\mathcal{A}'$ are all and only the states $\bar{q}$ with $q \in F$. It can be easily verified that $\mathcal{A}'$ accepts $w' = [u_0]_{\sim_A}[u_1]_{\sim_A}[u_2]_{\sim_A}...$ iff $\mathcal{A}$ accepts $w = u_0u_1u_2....$ □

Corollary 4. *The acceptance problem for effectively residually ultimately periodic words is decidable.*

Proof. The claim follows almost trivially from previous observations. Let $w = u_0u_1u_2...$ be an infinite word, where $(u_n)_{n\geq 0}$ is an effectively residually ultimately periodic sequence. Now, let $\mathcal{A}$ be a generic sequential Büchi automaton and let A' be the set of all $\sim_{\mathcal{A}}$-equivalence classes. Let $\mu : A^* \to A'$ be the function that maps a finite word u to its $\sim_{\mathcal{A}}$-equivalence class $[u]_{\sim_{\mathcal{A}}}$. Notice that the concatenation operator $\cdot$ can be naturally extended to elements of A' by exploiting the compatibility of $\sim_{\mathcal{A}}$ (namely, $[u]_{\sim_{\mathcal{A}}} \cdot [v]_{\sim_{\mathcal{A}}} = [u \cdot v]_{\sim_{\mathcal{A}}}$). Thus, μ turns out to be a morphism from the semigroup $(A^*, \cdot)$ into the finite semigroup $(A', \cdot)$. By definition of residually ultimately periodic sequence, $(\mu(u_n))_{n\geq 0}$ is an ultimately periodic sequence and we are able to compute two indices p and q such that $\mu(u_n) = \mu(u_{n+q})$, for all $n \geq p$. By Proposition 1, one can obtain a Büchi automaton $\mathcal{A}'$ that accepts the ultimately periodic word $w' = (\mu(u_0)...\mu(u_{p-1})) \cdot (\mu(u_p)...\mu(u_{p+q-1}))^\omega$ iff $\mathcal{A}$ accepts w. □

Closure properties of residually ultimately periodic functions In the previous part, we obtained decidability results for expanded structures of the form $(\mathbb{N}, <, P)$ by exploiting noticeable properties of effectively residually ultimately periodic words. Here we aim at providing a large class of predicates of the form $P = \{f(n)\}_{n\geq 0}$, where f is a strictly increasing function on natural numbers, whose characteristic words are effectively residually ultimately periodic. Such a class of predicates is defined in terms of strong closure properties w.r.t. natural arithmetic operations.

We start by defining ultimately periodic functions w.r.t. finite monoids and we prove some relevant properties of them. Such a notion of function is found, with different but equivalent definitions, in many areas of computer science (e.g., group theory, combinatorics of words, and automata theory). Roughly speaking, ultimately periodic functions w.r.t. finite monoids are functions over the natural numbers that manifest a strong repeating pattern whenever projected into any finite monoid. Examples of such functions are i^2, 2^i, $2^i - i^2$, i^i, $i!$, and the exponential tower $2^{2^{\cdot^{\cdot^{\cdot^2}}}}$.

Henceforth, for some given $p \geq 0$, $q > 0$, and $i \geq 0$, we denote by $[i]_{p,q}$ either the value i or the value $((i - p) \bmod q) + p$, depending on whether $i < p$ or $i \geq p$ holds.

Definition 12. *A function $f : \mathbb{N} \to \mathbb{N}$ is said to be* ultimately periodic w.r.t. finite monoids *if, given $k \geq 0$ and $r > 0$, one can compute $p \geq 0$ and $q > 0$ such that, for every $i \geq 0$,*

$$\big[f(i)\big]_{k,r} = \big[f([i]_{p,q})\big]_{k,r}. \tag{1}$$

The famility of 'ultimately periodic function w.r.t. finite monoids' owes its name to the fact that every function f satisfying Equation 1 can be characterized in terms of the periodicity of the sequences of the form $(a^{f(i)})_{i\geq 0}$, where a is an element of a finite (multiplicative) monoid $(S, \cdot)$ (a^0 is assumed to be the identity element of the monoid). This is formally stated in the following Proposition 2. As a matter of fact, such a characterization connects functions satisfying Definition 12 to sequences of words satisfying Definition 11. Indeed, since $(A^*, \cdot)$ is a monoid, given any ultimately periodic function w.r.t. finite monoids $f : \mathbb{N} \to \mathbb{N}$ and any symbol $a \in A$, the sequence $(u_n)_{n\geq 0}$ of words, where $u_n = a^{f(n)}$, turns out to be effectively residually ultimately periodic.

Proposition 2. *A function $f : \mathbb{N} \to \mathbb{N}$ is ultimately periodic w.r.t. finite monoids iff, given any finite (multiplicative) monoid[5] $(S, \cdot)$ and given any element $e \in S$, one can compute $p \geq 0$ and $q > 0$ such that, for all $i \geq 0$,*

$$e^{f(i)} = e^{f([i]_{p,q})},$$

namely, the sequence $(e^{f(i)})_{i\geq 0}$ is (effectively) ultimately periodic.

Proof. Let assume that f satisfies Equation 1 and fix an arbitrary pair of integers $k \geq 0$ and $r > 0$. Then, there are $p \geq 0$ and $q > 0$, computable from k and r, such that $\big[f(i)\big]_{k,r} = \big[f([i]_{p,q})\big]_{k,r}$ holds for all $i \geq 0$. Now, given any finite monoid $(S, \cdot)$ and any element $e \in S$, we know, from the Pigeonhole Principle, that there exist two positive integers $k \geq 0$ and $r > 0$, computable from $(S, \cdot)$ and e, such that $e^j = e^{[j]_{k,r}}$ holds for every $j \geq 0$. Thus, we have $e^{f(i)} = e^{[f(i)]_{k,r}} = e^{[f([i]_{p,q})]_{k,r}} = e^{f([i]_{p,q})}$.
For the converse implication, suppose that, given $f : \mathbb{N} \to \mathbb{N}$, for every finite monoid $(S, \cdot)$ and for every element $e \in M$, we can compute $p \geq 0$ and $q > 0$ such that, for all $i \geq 0$, $e^{f(i)} = e^{f([i]_{p,q})}$ holds. Let fix two integers $k \geq 0$ and $r > 0$. We can chose (compute) a finite monoid $(S, \cdot)$ and an element $e \in S$ such that r turns out to be the *least* integer such that e^r is the identity of S. Note that, by construction, for any pair of integers i and j, $a^i = a^j$ implies $i \bmod r = j \bmod r$, whence $[i]_{k,r} = [j]_{k,r}$. From the hypothesis on f, we can compute $p \geq 0$ and $q > 0$ such that, for every $i \geq 0$, $e^{f(i)} = e^{f([i]_{p,q})}$ holds. Hence, for every $i \geq 0$, we have $[f(i)]_{k,r} = [f([i]_{p,q})]_{k,r}$. □

Below, we describe a number of ways to obtain residually ultimately periodic functions, starting from a set of basic functions. Hereafter, we use $i = j \pmod m$ as a shorthand for $i \bmod m = j \bmod m$. We say that a function $f : \mathbb{N} \to \mathbb{N}$ has *unbounded infimum* if $lim\ inf_{i \to \infty} f(i) = \infty$. In such a case, we understand that, for any given $l \in \mathbb{N}$, we can compute i_0 such that $f(i) \geq l$ holds, for all $i \geq i_0$.

Proposition 3. *Let f and g be two ultimately periodic functions w.r.t. finite monoids. The following functions are also ultimately periodic w.r.t. finite monoids:*

1. **(Sum)** *$h = f + g$, defined by $h(i) = f(i) + g(i)$;*
2. **(Product)** *$h = f * g$, defined by $h(i) = f(i) * g(i)$;*
3. **(Difference)** *$h = f - g$, defined by $h(i) = f(i) - g(i)$, provided that h has unbounded infimum;*
4. **(Quotient)** *$h = \lfloor \frac{f}{d} \rfloor$, defined by $h(i) = \lfloor \frac{f(i)}{d} \rfloor$, where $d > 0$;*
5. **(Exponentiation)** *$h = f^g$, defined by $h(i) = (f(i))^{g(i)}$, provided that h has unbounded infimum;*
6. **(Exponential tower)** *h defined by $h(0) = 1$ and $h(i + 1) = b^{h(i)}$, where $b > 0$;*
7. **(Fibonacci numbers)** *h defined by $h(0) = h(1) = 1$ and $h(i + 2) = h(i) + h(i + 1)$;*
8. **(Generalized sum)** *h defined by $h(i) = \sum_{j=0}^{i-1} f(j)$;*
9. **(Generalized product)** *h defined by $h(i) = \prod_{j=0}^{i-1} f(j)$;*

[5] It should be noted that the claim of the proposition holds even if the term 'monoid' is replaced by 'semigroup', provided that $f(n) > 0$ holds for every $n \in \mathbb{N}$.

10. **(Substitution)** $h = f \circ g$, *defined by* $h(i) = g(f(i))$.

Proof. As for cases *1.* and *2.*, it suffices to note that the operator $[\ ]_{l,r}$ respects sums and products, namely, $[i+j]_{l,r} = [[i]_{l,r} + [j]_{l,r}]_{l,r}$ and $[i * j]_{l,r} = [[i]_{l,r} * [j]_{l,r}]_{l,r}$, for every $i, j \in \mathbb{N}$.
Similarly, one can show that $[\ ]_{l,r}$ respects differences, namely, $[i-j]_{l,r} = [[i]_{l,r} - [j]_{l,r}]_{l,r}$, provided that $i - j \geq l$. Thus, case *3.* follows immediately if we assume that $h = f - g$ has unbounded infimum.
Case *4.* is proved by noticing that, for every $l \geq 0$ and $r > 0$, $[h(i)]_{l,r}$ is either 0 or $[[h(i-d)]_{l,r} + 1]_{l,r}$, depending on whether $i < d$ or $i \geq d$. Thus, by defining $(h_1(i), ..., h_d(i)) = ([h(i)]_{l,r}, ..., [h(i+d-1)]_{l,r})$, we obtain

$$(h_1(i+1), ..., h_d(i+1)) = \begin{cases} (0, ..., 0) & \text{if } i = 0, \\ (h_2(i), ..., h_d(i), [h_1(i)+1]_{l,r}) & \text{if } i > 0. \end{cases}$$

Since each value $h_j(i)$ ranges over the finite domain $\{0, ..., l+r-1\}$, we can apply the Pigeonhole Principle and claim that there are two (computable) integers $p \geq 0$ and $q > 0$ such that $h_j(i) = h_j(i+q)$, for every $i \geq p$ and $j \in [d]$. This proves that $[h(i)]_{l,r} = [h([i]_{p,q})]_{l,r}$.
As for case *5.*, we preliminarily recall the definition of the 'Euler totient function' ϕ

$$\phi(i) = i \prod_{p \text{ prime dividing } i} \left(1 - \frac{1}{i}\right)$$

and the following two properties:

$$b^a = b^a + \sum_{i=1}^{a} \left(\binom{a}{i} b^{a-i} * 0\right) = b^a + \sum_{i=1}^{a} \left(\binom{a}{i} b^{a-i} * m^i\right) = (b+m)^a \pmod{m},$$
$$b^a = b^a * 1 = b^a * b^{\phi(m)} = b^{a+\phi(m)} \pmod{m}.$$

Since f and g satisfy Equation 1, one can compute $p, p' \geq 0$ and $q, q' \geq 0$ such that $[f(i)]_{0,m} = [f([i]_{p,q})]_{0,m}$ and $[g(i)]_{0,\phi(m)} = [g([i]_{p',q'})]_{0,\phi(m)}$. Now, by letting $r = max(p, p')$ and $s = lcm(q, q')$, we obtain

$$\begin{aligned} \left[f(i)^{g(i)}\right]_{0,m} &= \left[\left([f(i)]_{0,m}\right)^{[g(i)]_{0,\phi(m)}}\right]_{0,m} \\ &= \left[\left([f([i]_{r,s})]_{0,m}\right)^{[g([i]_{r,s})]_{0,\phi(m)}}\right]_{0,m} = \left[f\left([i]_{r,s}\right)^{g([i]_{r,s})}\right]_{0,m}. \end{aligned}$$

We can further generalize the above result for any $l \geq 0$. Let $\sigma(l)$ be the least integer i such that $i \geq r$ and $f(i)^{g(i)} \geq l$ (such a value exists and it is computable by hypothesis). Then, for every $i \geq \sigma(l)$, we have

$$\begin{aligned} \left[f\,(i)^{g(i)}\right]_{l,m} &= \left[f\,(i)^{g(i)} - l\right]_{0,m} + l = \left[\left[f\,(i)^{g(i)}\right]_{0,m} - l\right]_{0,m} + l \\ &= \left[\left[f\left([i]_{\sigma(l),s}\right)^{g([i]_{\sigma(l),s})}\right]_{0,m} - l\right]_{0,m} + l = \left[f\left([i]_{\sigma(l),s}\right)^{g([i]_{\sigma(l),s})}\right]_{l,m}. \end{aligned}$$

This proves that $h = f^g$ satisfies Equation 1.
We now prove case *6.*. We have $h(0) = 1$ and $h(i) = b^{h(i-1)}$, for every $i > 0$. Let $r > 0$ and, for every $l \geq 0$, let $\sigma(l) = \lceil \log_b(l) \rceil$. We prove, by induction on j, that, for every $0 \leq j \leq r$, $l \geq 0$, and $i \geq \sigma(l)$, the following equation holds

$$\left[h(i+j)\right]_{l,\phi^{r-j}(r)} = \left[h(i+j+1)\right]_{l,\phi^{r-j}(r)}.$$

The case $j = 0$ is almost trivial. Since $\phi(i)$ is strictly decreasing for $i > 1$, we have $\phi^r(r) = 1$. This implies that $[h(i)]_{l,\phi^r(r)} = l = [h(i+1)]_{l,\phi^r(r)}$ holds for every $i \geq \sigma(l)$. Now, let $j > 0$. For every $l \geq 0$

and for every $i \geq \sigma(l)$, we have

$$\begin{aligned}\Big[h(i+j)\Big]_{l,\phi^{r-j}(r)} &= \Big[b^{h(i+(j-1))}\Big]_{l,\phi^{r-j}(r)} = \Big[b^{[h(i+(j-1))]_{\sigma(l),\phi^{r-(j-1)}(r)}}\Big]_{l,\phi^{r-j}(r)} \\ &= \Big[b^{[h(i+(j-1)+1)]_{\sigma(l),\phi^{r-(j-1)}(r)}}\Big]_{l,\phi^{r-j}(r)} \\ &= \Big[b^{h(i+(j-1)+1)}\Big]_{l,\phi^{r-j}(r)} = \Big[h(i+j+1)\Big]_{l,\phi^{r-j}(r)}.\end{aligned}$$

In particular, by letting $j = r$, we have that, for every $l \geq 0$ and $i \geq \sigma(l)$,

$$\big[h(i+r)\big]_{l,r} = \big[h(i+r+1)\big]_{l,r}.$$

Therefore, we can conclude that $[h(i)]_{l,r} = [h([i]_{\sigma(l)+r,1}]_{l,r}$ for every $l \geq 0$, $r > 0$, and $i \geq 0$.
In case *7.*, we have $[h(i)]_{l,r} = [[h(i-2)]_{l,r} + [h(i-1)]_{l,r}]_{l,r}$ whenever $i \geq 2$. By defining $(h_1(i), h_2(i)) = ([h(i)]_{l,r}, [h(i+1)]_{l,r})$, we have

$$(h_1(i+1), h_2(i+1)) = \begin{cases} ([1]_{l,r}, [1]_{l,r}) & \text{if } i = 0, \\ (h_2(i), [h_1(i) + h_2(i)]_{l,r}) & \text{if } i > 0. \end{cases}$$

Since the values $h_1(i)$ and $h_2(i)$ range over the finite domain $\{0, ..., l + r - 1\}$, we can apply the Pigeonhole Principle and claim that there are two (computable) integers $p \geq 0$ and $q > 0$ such that $(h_1(i), h_2(i)) = (h_1(i+q), h_2(i+q))$, for every $i \geq p$. This implies that $[h(i)]_{l,r} = [h([i]_{p,q})]_{l,r}$.
We now prove case *8.* (case *9.* follows similarly). Given a function f satisfying Equation 1, we fix two integers l, r and we denote by p, q the (computable) integers such that $[f(i)]_{l,r} = [f([i]_{p,q})]_{l,r}$. We further define $S = \sum_{j=p+1}^{p+q} f(j)$ and we notice that $\big[\sum_{j=p+1}^{p+nq} f(j)\big]_{l,r} = \big[nS\big]_{l,r}$ holds, for every integer $n \geq 0$. From the Pigeonhole Principle, there are suitable integers p', q' such that $[p'nS]_{l,r} = [(p' + q')nS]_{l,r}$. Moreover, for every $i \geq p + qq'$, one can compute an integer n_i such that $i = [i]_{p,qq'} + n_i qq'$. Thus, for every $i \geq p + qq'$, we obtain

$$\begin{aligned}\Big[h(i)\Big]_{l,r} &= \Big[\sum_{j=0}^{i} f(j)\Big]_{l,r} = \Big[\sum_{j=0}^{[i]_{p,qq'}} f(j) + \sum_{j=[n]_{p,qq'}+1}^{i} f(j)\Big]_{l,r} \\ &= \Big[\sum_{j=0}^{[i]_{p,qq'}} f(j) + q' n_i S\Big]_{l,r} = \Big[\sum_{j=0}^{[i]_{p,qq'}} f(j) + q'[n_i]_{p',q'} S\Big]_{l,r} = \Big[h([i]_{p,qq'})\Big]_{l,r}.\end{aligned}$$

It remains to show case *10.*. This is immediately proved by noticing that, given $l \geq 0$ and $r > 0$, one can compute $p, p' \geq 0$ and $q, q' > 0$ satisfying

$$\big[g(f(i))\big]_{l,r} = \big[g([f(i)\big]_{p,q})]_{l,r} = \big[g([f([i]_{p',q'})]_{p,q})\big]_{l,r} = \big[g(f([i]_{p',q'}))\big]_{l,r}.$$

□

As a consequence of Proposition 3, we know that if $f : \mathbb{N} \to \mathbb{N}$ is an ultimately periodic function w.r.t. finite monoids such that $f(n+1) - f(n)$ has unbounded infimum (intuitively, the sequence $(f(n))_{n\in\mathbb{N}}$ grows in a superlinear way), then the sequence of words $(u_n)_{n\geq 0}$, where $u_0 = 0^{f(0)} \cdot 1$ and $u_{n+1} = 0^{f(n+1)-f(n)-1} \cdot 1$, is effectively residually ultimately periodic. This further implies that if w_P is the characteristic word of the predicate $P = \{f(n) : n \in \mathbb{N}\}$, then acceptance problem Acc_{w_P} is decidable and so it is the model-checking problem for MSO logic over $(\mathbb{N}, <, P)$. As an example, this result accounts for the decidability of $(\mathbb{N}, <, P)$, where P is $\{n^k : n \in \mathbb{N}\}$, $\{k^n : n \in \mathbb{N}\}$, $\{n^n : n \in \mathbb{N}\}$, $\{n! : n \in \mathbb{N}\}$, etc.

Morphic words. Here we consider the class of morphic words, namely, those words that are generated by repeatedly applying a fixed morphism. Let $\tau : A^* \to A^*$ be a morphism from $(A^*, \cdot)$ into $(A^*, \cdot)$. We denote by τ^n the n-fold iteration of τ and by x_n the word $\tau^n(a)$, where a is a distinguished symbol of A. If the first letter of $\tau(a)$ is a, then it is easily shown that each word x_n is a prefix of x_{n+1}. If furthermore the sequence $(|x_n|)_{n\geq 0}$ is not bounded, then the sequence $(x_n)_{n\geq 0}$ converges to an infinite word x, which is denoted by $\tau^\omega(a)$. Notice that x is a *fixed point* of τ, since $x = \tau(x)$.

As an example, consider the morphism τ such that $\tau(a) = ab$, $\tau(b) = ccb$ and $\tau(c) = c$. By letting $x_n = \tau^n(a)$, we have $x_0 = a$, $x_1 = ab$, $x_2 = abccb$, $x_3 = abccbccccb$, $x_4 = abccbccccbcccccb$, etc. The fixed point $\tau^\omega(a)$ of τ is easily shown to be $x = abc^2bc^4bc^6bc^8$....

Definition 13. *An infinite word x over an alphabet B is said to be* morphic *if there is a morphism τ from $(A^*, \cdot)$ into $(A^*, \cdot)$, a morphism σ from $(A^*, \cdot)$ into $(B^*, \cdot)$, and a symbol $a \in A$ such that $x = \sigma(\tau^\omega(a))$.*

Consider the morphism τ introduced before and its fixed point $\tau^\omega(a) = abc^2bc^4bc^6bc^8...$ and let σ be the morphism such that $\sigma(a) = 1$, $\sigma(b) = 1$, and $\sigma(c) = 0$. The morphic word $\sigma(\tau^\omega(a)) = 1100100001...$ is the characteristic word of the predicate $P = \{n^2 : n \in \mathbb{N}\}$. In fact, it can be proved that the characteristic words of the predicates of the form $\{n^k : n \in \mathbb{N}\}$ or $\{k^n : n \in \mathbb{N}\}$ are morphic.

We conclude the section with the following result, which implies that the acceptance problem for morphic words is decidable.

Proposition 4. *Every morphic word $x = \sigma(\tau^\omega(a))$ is effectively residually ultimately periodic.*

Proof. Let $\tau(a) = a{\cdot}u$, $u_0 = a{\cdot}u$, and $u_n = \tau^n(u)$ for all $n > 0$. Clearly, we have $x = \sigma(u_0)\sigma(u_1)\sigma(u_2)$.... Now, let μ be a generic morphism from $(A^*, \cdot)$, where A is the alphabet of x, into a finite semigroup $(S, \odot)$. Since S is finite, there are only finitely many morphisms from $(A^*, \cdot)$ into $(S, \odot)$ and hence we can find two indices p and q such that $\mu \circ \sigma \circ \tau^n = \mu \circ \sigma \circ \tau^{n+q}$, for every $n \geq p$. This implies that $\mu(u_n) = \mu(u_{n+q})$ for every $n \geq p$, which shows that $(u_n)_{n\geq 0}$ is an effectively residually ultimately periodic sequence of words. □

5.2 The case of tree structures

In this section we generalize the contraction method from expanded linear orderings to deterministic colored trees. In analogy to Elgot and Rabin method, the resulting framework exploits a notion of 'decomposition' of trees and a suitable 'indistinguishability' relation for Rabin tree automata. However, due to the complications involved by the notion of tree decomposition and due to the necessity of considering Rabin tree automata rather than Büchi tree automata, the transfer of Elgot and Rabin method to tree structures is far from being trivial and it needs a bit of notation and a number of technical results. For further details, we refer the reader to [58, 56, 57].

Rabin's Theorem [70] establishes a strong correspondence between MSO formulas satisfied by an expanded tree structure $(T, \bar{P})$ and Rabin (equivalently, Muller) tree automata accepting its characteristic tree $T_{\bar{P}}$: for every formula $\varphi(\bar{X})$, one can compute a tree automaton $\mathcal{A}$ (and, conversely, for every tree automaton $\mathcal{A}$, one can compute a formula $\varphi(\bar{X})$) such that $T \vDash \varphi[\bar{P}]$ iff $T_{\bar{P}} \in \mathscr{L}(\mathcal{A})$. Let us call *acceptance problem* of a given tree $T_{\bar{P}}$, denoted $Acc(T_{\bar{P}})$, the problem of deciding, for any tree automaton $\mathcal{A}$, whether $\mathcal{A}$ accepts $T_{\bar{P}}$. We have that the MSO-theory of a tree structure $(T, \bar{P})$ is decidable iff $Acc(T_{\bar{P}})$ is decidable. For the sake of simplicity, hereafter we shall omit the subscript $\bar{P}$, thus writing T for $T_{\bar{P}}$. Given a *regular* tree T, by viewing T as a tree automaton recognizing the singleton $\{T\}$ and by exploiting the closure of tree automata with respect to intersection and the decidability of the emptiness problem, one can easily show that the problem $Acc(T)$ is decidable. In the following, we extend such a result to a large class of tree structures, including non-regular trees. From now on, we shall use the term tree automaton to mean a Muller tree automaton, keeping in mind that the results presented in this section are applicable to any other equivalent notion of tree automaton.

Indistinguishability of trees. We start with some preliminary definitions. A *partial run* of a tree automaton $\mathcal{A} = (S, A, C \cup \{\bot\}, \delta, \mathcal{I}, \mathcal{F})$ on a *non-empty full* tree T is an S-colored tree P such that $\mathcal{D}om(P) = \mathcal{D}om(T)$ and for every internal vertex v of P, $\big(P(v), T(v), (P(va))_{a\in A}\big) \in \delta$. Given a tree automaton $\mathcal{A}$, for any non-empty full tree T and any partial run P of $\mathcal{A}$ on T, we define the *feature* $[T, P]$ as the triple

$$(P(\varepsilon), \{(T(v), P(v), \mathcal{I}mg(P|\pi_v) \,:\, v \in \mathcal{F}r(P)\}, \{\mathcal{I}nf(P|\pi) \,:\, \pi \in \mathcal{B}ch(P)\})$$

where $\mathcal{F}r(P)$ denotes the set of all leaves in P, π_v denotes the access path of a leaf v in P, $\mathcal{I}mg(P|\pi_v)$ denotes the set of states that occur along the access path π_v of P, $\mathcal{B}ch(P)$ denotes the sets of all infinite paths in P, and $\mathcal{I}nf(P|\pi)$ denotes the set of states that occur infinitely often along the infinite path π of P.

We now define (C-colored) *B-augmented* trees. These trees have internal nodes colored over C and leaves colored over $C \cup B$, with B being a finite set disjoint from C. Even though a C-colored B-augmented tree can be viewed as a $C \cup B$-colored tree, it is obviously not true that a $C \cup B$-colored tree is a C-colored B-augmented tree. On the other hand, a C-colored tree can be always thought of as a C-colored B-augmented tree, for any given B. We call the B-colored leaves in a B-augmented tree *markers*, *placeholders*, or *variables*, depending on the context and the kind of operations we are interested in.

In order to generalize the notions of partial run and feature to empty, non-full, and/or B-augmented trees, we introduce a suitable operation that extends a tree with $\bot$-colored vertices. The *B-completion* of a (B-augmented) tree T is the tree T_B defined as follows. If T is the empty tree, then T_B is the infinite complete $\bot$-colored tree. If T is a non-empty tree, then $\mathcal{D}om(T_B) = \mathcal{D}om(T) \cup (F \cdot A^*) \cup (G \cdot A^*)$, where $F = \{va \,:\, v \in \mathcal{F}r(T), T(v) \in C, a \in A\}$ and $G = \{va \,:\, v \in \mathcal{D}om(T) \setminus \mathcal{F}r(T), a \in A, va \notin \mathcal{D}om(T)\}$, $T_B(v) = T(v)$ for every $v \in \mathcal{D}om(T)$, and $T_B(v) = \bot$ for every $v \in (F \cdot A^*) \cup (G \cdot A^*)$. Notice that T_B is a non-empty full B-augmented tree, whose leaves are all and only the B-colored leaves of T (if any).

Given a B-augmented tree T and a tree automaton $\mathcal{A}$, to decide whether $T \in \mathscr{L}(\mathcal{A})$ we define the $\mathcal{A}, B$-types of T as collections of features of the form $[T_B, P]$, where P ranges over a suitable set $\mathcal{P}$ of partial runs of $\mathcal{A}$ on T_B (different choices of $\mathcal{P}$ may result into different $\mathcal{A}, B$-types of T). We allow $\mathcal{P}$ to be a *proper subset* of the set of all partial runs of $\mathcal{A}$ on T_B, because there can be partial runs which are redundant with respect to others and thus can be 'forgotten'. The notion of *redundant* partial run is defined as follows. Given a B-augmented tree T and a tree automaton $\mathcal{A}$, we define a relation $\preceq$ over the set of all partial runs of $\mathcal{A}$ on T_B such that P' is redundant with respect to P iff $P \preceq P'$. For every pair of partial runs P, P' on T_B, we have $P \preceq P'$ iff (i) $P(\varepsilon) = P'(\varepsilon)$, (ii) for every leaf v of P, there is a leaf v' of P' such that $T(v) = T(v')$, $P(v) = P'(v')$, and $\mathcal{I}mg(P|\pi_v) = \mathcal{I}mg(P'|\pi_{v'})$, and (iii) for every infinite path π in P, there is an infinite path π' in P' such that $\mathcal{I}nf(P|\pi) = \mathcal{I}nf(P'|\pi')$. Notice that the relation $\preceq$ is a *quasi-order*. Moreover, if P and P' are two runs of $\mathcal{A}$ on T, $P \preceq P'$, and P' is a successful run, then P is a successful run as well. Given a set $\mathcal{P}$ of partial runs of $\mathcal{A}$ on T_B, we say that $\mathcal{P}$ is *complete* if for every partial run P' of $\mathcal{A}$ on T_B, there is $P \in \mathcal{P}$ such that $P \preceq P'$.

Definition 14. *Given a tree automaton $\mathcal{A}$ and a B-augmented tree T, an* $\mathcal{A}, B$-type *of T is a set of features of the form $[T_B, P]$, where P ranges over a complete set of partial runs of $\mathcal{A}$ on T_B. The* basic *$\mathcal{A}, B$-type of T is the (unique) set of features of the form $[T_B, P]$, where P ranges over all partial runs of $\mathcal{A}$ on T_B.*

We denote by $\mathscr{T}_{\mathcal{A},B}$ the set of all possible $\mathcal{A}, B$-types of (B-augmented) trees. Since $\mathscr{T}_{\mathcal{A},B}$ is included in the *finite* set $\mathscr{P}(S \times \mathscr{P}(B \times S \times \mathscr{P}(S)) \times \mathscr{P}(\mathscr{P}(S)))$, there exist only finitely many $\mathcal{A}, B$-types, for any choice of $\mathcal{A}$ and B.

It is possible to show that the acceptance problem for a colored tree T is equivalent to the problem of computing (and checking) its $\mathcal{A}, \emptyset$-types (according to Definition 14, an $\mathcal{A}, \emptyset$-type is a collection of features whose second component is the empty set) [58]. More precisely, we have that for any given

tree automaton $\mathcal{A}$ and for any given $\mathcal{A},\emptyset$-type of a tree T, one can establish whether $T \in \mathscr{L}(\mathcal{A})$ by simply checking whether there exists a feature $(s, \emptyset, \{W_j\}_{j\in J})$ of the $\mathcal{A},\emptyset$-type such that (i) s is an initial state of $\mathcal{A}$ and (ii) for every $j \in J$, W_j is a set of states satisfying the acceptance condition of $\mathcal{A}$. Conversely, if $Acc(T)$ is decidable (this is the case, for instance, with regular trees), then, for any given tree automaton $\mathcal{A}$, one can compute the basic $\mathcal{A},\emptyset$-type of T by an automaton-driven selection of the $T_\emptyset$ features from the set of all candidate features. Such a selection can be done by building, for any candidate feature f, an automaton $\mathcal{A}_f$ such that $\mathcal{A}_f$ accepts $T_\emptyset$ if and only if f belongs to the basic $\mathcal{A},\emptyset$-type of T. As a general rule, the $\mathcal{A},\emptyset$-types of a tree T can be computed from its $\mathcal{A}, B$-types, for any given set B. As a corollary, we have that pairs of trees T, T' which have an $\mathcal{A}, B$-type in common are indistinguishable by the automaton $\mathcal{A}$, that is, $T \in \mathscr{L}(\mathcal{A})$ iff $T' \in \mathscr{L}(\mathcal{A})$.

From trees to their retractions. We now show how $\mathcal{A}, B$-types can actually be exploited to solve non-trivial instances of the acceptance problem. To this end, we introduce the notion of factorization, which allows us to decompose a tree T into basic components. Each component, called *factor*, is obtained by selecting the elements of T that lie in between some distinguished vertices. Taking advantage of the notion of factorization, we define *tree retractions*, which are tree-shaped arrangements of $\mathcal{A}, B$-types corresponding to the factors of a tree. Then we prove that the acceptance problem for a tree T can be reduced to the acceptance problem for a retraction of it.

Definition 15. *Given a tree T, a* factorization *of T with respect to a set B is a (possibly non-deterministic) B-labeled uncolored tree Π such that (i) $\mathcal{D}om(\Pi) \subseteq \mathcal{D}om(T)$ and $\varepsilon \in \mathcal{D}om(\Pi)$, (ii) for all $u, u' \in \mathcal{D}om(\Pi)$, with $u \neq u'$, (u, u') is an edge of Π iff u' is a descendant of u in T and there exist no other $u'' \in \mathcal{D}om(\Pi)$ that belong to the path from u to u' in T, and (iii) the labels of Π edges are arbitrarily chosen in B.*

We can graphically represent a factorization of a tree by first identifying its vertices (e.g., the black colored vertices in Figure 5.8) and then drawing the resulting edges together with the chosen labels (e.g., the bold arrows in Figure 5.8).

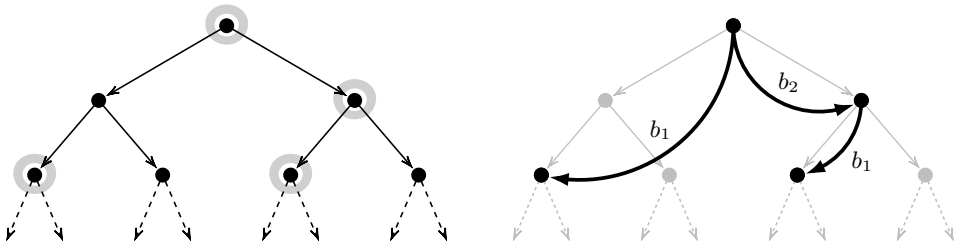

Fig. 5.8. An example of factorization.

The (marked) factors of a tree T with respect to a factorization Π are defined as follows. Let $u \in \mathcal{D}om(\Pi)$ and $Succ(u) = \{u' : (u, u') \text{ is an edge of } \Pi\}$. The *unmarked factor* of T rooted at u, denoted by $T_\Pi[u]$, is the tree obtained by selecting the vertices of $T_\emptyset$ which are descendants (in $T_\emptyset$) of u, but not proper descendants in $T_\emptyset$) of any $u' \in Succ(u)$. For any vertex $u \neq \varepsilon$, we define the *marker* of u as the label $b \in B$ of the (unique) edge in Π with target u, and we denote it by $m_\Pi[u]$. The *marked factor* of T rooted at u, denoted by $T^+_\Pi[u]$, is the tree obtained from $T_\Pi[u]$ by recoloring each leaf u' with the corresponding marker $m_\Pi[u']$. Notice that a marked factor is a non-empty full B-augmented tree whose leaves are colored over B. This allows us to define the $\mathcal{A}, B$-types of the marked factors of T.

Definition 16. *Given a tree T, a tree automaton $\mathcal{A}$, and a factorization Π of T with respect to a set B, a* retraction *of T with respect to $\mathcal{A}$ and Π is a B-labeled $\mathscr{T}_{\mathcal{A},B}$-colored tree R such that (i) $\mathcal{D}om(R) = \mathcal{D}om(\Pi)$, (ii) (u,u') is a b-labeled edge in R iff (u,u') is a b-labeled edge in Π, and (iii) each vertex u in R is colored with an $\mathcal{A},B$-type of the corresponding marked factor $T^{+}_{\Pi}[u]$.*

Both Definition 15 and Definition 16 can be generalized to B-augmented trees. In general, a retraction R, as well as a factorization Π, of T may be a nondeterministic tree, possibly having vertices with unbounded (or even infinite) out-degree. Since tree automata operate on *deterministic* trees, we restrict ourselves to retractions which are bisimilar to deterministic trees. Moreover, by definition, retractions depend on automata, but, as a matter a fact, for all the considered tree structures, we shall provide a single factorization from which we will be able to generate a suitable retraction for any tree automaton.

Given a tree T, a tree automaton $\mathcal{A}$, and a factorization Π of T with respect to B, we now build an automaton $\mathcal{A}^B$ such that $\mathcal{A}$ accepts (the $\emptyset$-completion of) T iff $\mathcal{A}^B$ accepts (the $\emptyset$-completion of) a retraction R of T with respect to $\mathcal{A}$ and Π. The automaton $\mathcal{A}^B$ mimics the behavior of $\mathcal{A}$ at a 'coarser' level and it can be effectively computed from $\mathcal{A}$ and B. Its input alphabet is the set $\mathscr{T}_{\mathcal{A},B}$ of all $\mathcal{A},B$-types, plus the additional symbol $\perp$; its states encode the finite information processed by $\mathcal{A}$ during its computations up to a certain point; its transitions compute the new states which (possibly) extend information provided by the current state with information provided by the input symbol (the $\mathcal{A},B$-type of a marked factor). The automaton $\mathcal{A}^B$ is formally defined as follows.

Definition 17. *Given a tree automaton $\mathcal{A} = (S, A, C \cup \{\perp\}, \delta, \mathcal{I}, \mathcal{F})$ and a finite set B disjoint from C, $\mathcal{A}^B = (Z, B, \mathscr{T}_{\mathcal{A},B} \cup \{\perp\}, \delta', \mathcal{I}', \mathcal{F}')$, where:*

- $Z = B \times \{0,1\} \times \mathscr{P}\big(S \times \mathscr{P}(S) \times \mathscr{P}(S)\big) \times \mathscr{P}\big(\mathscr{P}(S)\big)$;
- *for every state $z = (c, x, \mathcal{U}, \mathcal{V})$ and every tuple $(z_b)_{b\in B} \in Z^B$, $\big(z, \perp, (z_b)_{b\in B}\big) \in \delta'$ iff for all $b \in B$, $z_b = (c, 1, \mathcal{U}, \mathcal{V})$;*
- *for every $\mathcal{A},B$-type $t = \big\{\big(s_h, \{(b_i, q_{h,i}, Q_{h,i})\}_{i\in I}, \{W_{h,j}\}_{j\in J}\big)\big\}_{h\in H}$, where H, I, J are suitable sets of indices, every state $z = (c, 0, \mathcal{U}, \mathcal{V})$, where $\mathcal{U} = \{(r_l, U_l, Y_l)\}_{l\in L}$ and $\mathcal{V} = \{V_g\}_{g\in G}$, for suitable set of indices L, G, and every tuple $(z_b)_{b\in B} \in Z^B$, $\big(z, t, (z_b)_{b\in B}\big) \in \delta'$ iff for all $l \in L$, there exists $h_l \in H$ such that (i) $r_l = s_{h_l}$ and (ii) for all $b \in B$, $z_b = (b, 0, \mathcal{U}_b, \mathcal{V}_b)$, where $\mathcal{U}_b = \{(q_{h_l,i}, Q_{h_l,i}, Y_l \cup Q_{h_l,i})\}_{l\in L, i\in I, b_i = b}$ and $\mathcal{V}_b = \mathcal{V} \cup \{W_{h_l,j}\}_{l\in L, j\in J}$;*
- *$\mathcal{I}'$ consists of all states of the form $(c, 0, \{(s, \emptyset, \emptyset)\}, \emptyset)$, with $c \in B$ and $s \in \mathcal{I}$;*
- *$\mathcal{F}'$ consists of all sets of states of the form $\{(c_1, x, \mathcal{U}_1, \mathcal{V}), ..., (c_n, x, \mathcal{U}_n, \mathcal{V})\}$ such that (i) if $x = 1$, then $n = 1$, $\mathcal{U}_1 = \big\{(r_l, U_l, Y_l)\big\}_{l\in L}$, and for all $l \in L$, the $\emptyset$-completion of the singleton tree c_1 is accepted by $\mathcal{A}[\{r_l\}/\mathcal{I}]$, where $\mathcal{A}[\{r_l\}/\mathcal{I}]$ is obtained from $\mathcal{A}$ by substituting $\{r_l\}$ for $\mathcal{I}$, (ii) if $x = 0$ and for $k = 1, \ldots, n$, $\mathcal{U}_k = \{(r_{k,l}, U_{k,l}, Y_k)\}_{l\in L_k}$, where L_k is a suitable set of indices, then for all $\emptyset \subsetneq L'_1 \subseteq L_1, \ldots, \emptyset \subsetneq L'_n \subseteq L_n$, $\bigcup_{1\leq k\leq n, l\in L'_k} U_{k,l} \in \mathcal{F}$, (iii) $\mathcal{V} \subseteq \mathcal{F}$.*

The following main theorem reduces the acceptance problem for $\mathcal{A}$ over T to that for $\mathcal{A}^B$ over R.

Theorem 9. *Given a tree T, a tree automaton $\mathcal{A}$, a factorization Π of T with respect to B, and a retraction R of T with respect to $\mathcal{A}$ and Π, we have that $T \in \mathscr{L}(\mathcal{A})$ iff $R \in \mathscr{L}(\mathcal{A}^B)$.*

The proof of Theorem 9 is rather involved (details can be found in [58]). It is based on a suitable correspondence between (the features of) the runs of $\mathcal{A}^B$ on R and (the features of) the runs of $\mathcal{A}$ on the $\emptyset$-completion $T_\emptyset$ of T. More precisely, we say that a run P of $\mathcal{A}$ on $T_\emptyset$ *corresponds* to a run Q of $\mathcal{A}^B$ on R if and only if the feature $[T_\emptyset, P]$ can be computed from the feature of $[R, Q]$. It is possible to show that for every run Q of $\mathcal{A}^B$ on R, there exists a corresponding run P of $\mathcal{A}$ on $T_\emptyset$. Conversely, for every run P of $\mathcal{A}$ on $T_\emptyset$, there exists a run P' of $\mathcal{A}$ on $T_\emptyset$, with $P' \preceq P$, and a run Q of $\mathcal{A}^B$ on R such that P' corresponds to Q (notice that there may not exist a run Q of $\mathcal{A}^B$ on R that directly corresponds to P). This two-way correspondence between runs allows one to compute an $\mathcal{A},\emptyset$-type of T from any given $\mathcal{A}^B,\emptyset$-type of R. As a matter of fact, Theorem 9 can be generalized to deal with partial runs, rather than complete runs, thus allowing one to compute an $\mathcal{A},B$-type t of

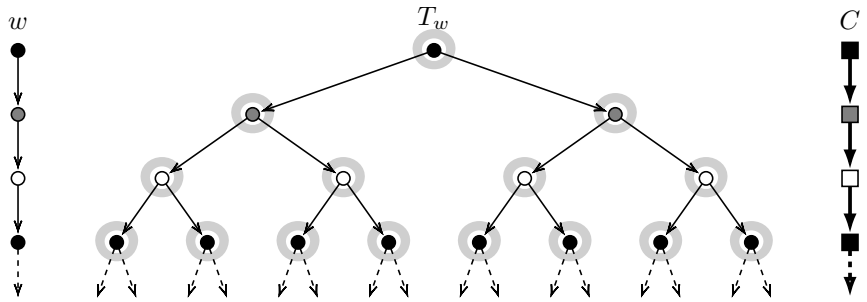

Fig. 5.9. A simple application example.

T from a given $\mathcal{A}^B, \emptyset$-type t' of R. However, even in the case in which t' is the *basic* $\mathcal{A}^B, \emptyset$-type of R, we cannot guarantee the computed t to be the *basic* $\mathcal{A}, B$-type of T. This further explains the need for the notions of *redundant* partial run and *complete* set of partial runs.

We conclude the section with a simple application of the proposed decision method. Let w be an infinite word over an alphabet C. It can be viewed as an expanded linear structure $(\omega, E, (V_c)_{c\in C})$, where $(i, j) \in E$ iff $j = i + 1$ and $i \in V_c$ iff $w[i] = c$. We denote by T_w the infinite complete $\{1, 2\}$-labeled tree obtained by coloring with $w[i]$ the vertices that belongs to the i-th level of the tree, that is, $T_w(u) = w[|u| + 1]$, for every $u \in \{1, 2\}^*$ (see Figure 5.9). If the MSO-theory of w is decidable, then that of T_w is decidable as well. This can be proved by showing that T_w is nothing but the unfolding of the graph G that is obtained from w via the MSO-definable interpretation that introduces a new copy of each edge of w. Thus, by exploiting the MSO-compatibility of unfoldings, one can first reduce the model checking problem for T_w to the one for G, which can in its turn be reduced to the one for w. Our method provides an alternative proof of the decidability of the MSO-theory of T_w, which is independent from the MSO-compatibility of the unfolding operation. Let $B = \{b\}$ and Π be the factorization of T_w with respect to B such that $\mathcal{D}om(\Pi) = \mathcal{D}om(T_w)$ (all edges of Π are labeled with the same symbol b). Given a tree automaton running on T_w, we denote by R a retraction of T_w with respect to $\mathcal{A}$ and Π. There is an obvious computable function Ω that maps each symbol $c \in C$ to the basic $\mathcal{A}, B$-type of the B-augmented tree $c\langle b, b\rangle$. Hence, a retraction R can be obtained from an MSO-definable interpretation of w which replaces every color c by the corresponding basic $\mathcal{A}, B$-type $\Omega(c)$. The decidability of the MSO-theory of R follows from the MSO-compatibility of MSO-definable interpretations. By exploiting Theorem 9, we can then conclude that the MSO-theory of T_w is decidable as well.

The class of reducible trees. Here we define the class of reducible trees, which properly includes that of regular trees, whose elements enjoy decidable acceptance problems. Roughly speaking, the decidability of the acceptance problem follows from the possibility of repeatedly reducing an instance of the problem involving a reducible trees to an equivalent instance involving one of its retractions. We also show that the class of reducible trees is closed with respect to various natural operations. These results, besides showing the robustness of the class of reducible trees, provide a neat framework to reason on retractions of trees and to easily transfer decidability results. Reducible trees are inductively defined as follows.

Definition 18. *Any regular tree is a* rank 0 tree*. Given a tree T and a natural number $n > 0$, T is a* rank n tree *if, for every tree automaton $\mathcal{A}$, there exist a finite set B, a factorization Π of T with respect to B, and a retraction R of T with respect to $\mathcal{A}$ and Π such that (i) for every $u \in \mathcal{D}om(\Pi)$, the marked factor $T^+_\Pi[u]$ is a regular tree and (ii) R is a rank $n-1$ tree. A* reducible tree *is a* rank n tree *for some $n \geq 0$.*

According to Definition 18, the decidability of the MSO-theories of reducible trees follows from Theorem 9 and the decidability of the MSO-theories of regular trees, provided that there exists an effective way to compute B, $(\Pi,)$ and R from T for any $\mathcal{A}$. Let the *footprint* of a tree T be the minimum amount of information that should be provided to make the reduction from T to its retraction feasible. Such a footprint can be inductively defined as follows. Given a rank 0 tree T, a footprint of T is any finite rooted colored graph, whose unfolding is isomorphic to T. Given a rank $n > 0$ tree T, a footprint of T is any computable function ξ mapping a tree automaton $\mathcal{A}$ to a set B and a footprint of a rank $n-1$ B-labeled tree R which is a retraction of T with respect to $\mathcal{A}$. Hereafter, we restrict ourselves to reducible trees which are modeled according to any suitable (extrinsic or intrinsic) representation system that allows the computation of their footprints. Under such a restriction, we have the following theorem.

Theorem 10. *Reducible trees enjoy a decidable acceptance problem.*

Closure properties of reducible trees. We say that the class of rank n trees (resp., reducible trees) is effectively closed under a family $\mathscr{F}$ of operations on trees whenever the application of any transformation $t \in \mathscr{F}$ results in a tree whose footprint is computable on the grounds of the footprints of the input trees. In the following, we show that reducible trees are closed under (suitable variants of) three powerful operations on trees, namely, *finite-state recolorings*, *second-order tree substitutions*, and *top-down deterministic tree transducers*.

As for the first operation, we distinguish among three different notions of recoloring: finite-state recoloring without lookahead (i.e., the output of a Mealy tree automaton working in top-down fashion on an input colored tree), finite-state recoloring with bounded lookahead (which allows the inspection of the subtree rooted at the current position up to a bounded depth and makes transitions dependent on that portion of the subtree), and finite-state recoloring with rational lookahead (which allows the inspection of the whole subtree rooted at the current position and classifies it according to a given finite class of rational tree languages).

A second-order tree substitution of the form $T[\![U_c]\!]_{c \in C}$ replaces all c-colored vertices in the tree T by a new tree U_c, simultaneously for all colors $c \in C$. The subtrees rooted at the 1-st, 2-nd, ..., k-th successor of a replaced c-colored vertex are possibly attached to the replacing tree U_c as follows. We mark the leaves of U_c with elements from A, which act as placeholders for the subtrees to be attached, making every replacing term an A-augmented tree. As an alternative, we can view any second-order tree substitution either as a function σ, specified by some replacing terms $U_1, ..., U_m$, with $C = \{c_1, ..., c_m\}$, that maps a tree T to the tree $T[\![U_1/c_1, ..., U_m/c_m]\!]$, or as a function γ, specified by a tree T and by an n-tuple of colors $c_{i_1}, ..., c_{i_n}$, with $1 \leq n \leq m$, that maps the n-tuple of A-augmented trees $(U_1, ..., U_n)$ to $T[\![U_1/c_{i_1}, ..., U_n/c_{i_n}]\!]$. These two latter views of a second-order tree substitution give rise to the notions of tree morphism and tree insertion, respectively. We say that a tree morphism (resp., a tree insertion) is regular if the trees $U_1, ..., U_m$ are regular (resp., the tree T is regular).

Top-down deterministic tree transducers are finite-state machines that process a tree in a top-down fashion and replace the vertex in the current position with a suitable regular tree, which may depend on the current state and on the color of the vertex. At each computation step, different states can be spread among different (copies of the) successors of the current vertex. Like finite-state recolorings, tree transducers can be enriched with the facility of bounded/rational lookahead.

Theorem 11. *For every natural number n, the class of rank n trees is effectively closed under finite-state recolorings with bounded lookahead and regular tree morphisms.*

Here we do not provide the details of the proofs of such closure properties (they can be found in [58]). Instead, we give an intuitive idea of how they can be proved. Roughly speaking, closure properties of reducible trees rest on the possibility of transferring the complexity of the tree resulting from the application of a transformation back to the automaton running on the original tree. Let $\mathscr{F}$

be the set of transformations of Theorem 11. First, one can easily prove that the class of regular trees is closed under transformations in $\mathscr{F}$ (as well as with respect to many other natural transformations of trees). As for the closure of rank n trees, with $n > 0$, under transformations in $\mathscr{F}$, let us consider a rank n tree T, a transformation $t \in \mathscr{F}$ mapping T to $t(T)$, and a tree automaton $\mathcal{A}$ running over $t(T)$. We can build a suitable tree automaton $\mathcal{A}'$ running on T, a rank $n-1$ retraction R of T with respect to $\mathcal{A}'$, and a transformation $t' \in \mathscr{F}$ mapping R to $t'(R)$ such that $t'(R)$ is a retraction of $t(T)$ with respect to $\mathcal{A}$. Then, by exploiting induction on n, we have that $t'(R)$ is a rank $n-1$ tree, thus showing that $t(T)$ is a rank n tree.

The tree transformations we described so far are not independent. In particular, it is possible to show that the composition of finite-state recolorings without lookahead (resp., with bounded, rational lookahead) with regular tree morphisms subsumes deterministic top-down tree transducers without lookahead (resp., with bounded, rational lookahead). More precisely, given a tree transducer $\mathcal{T}$ without lookahead (resp., with bounded, rational lookahead), the output of $\mathcal{T}$ on a tree T can be obtained by applying to T first a regular tree morphism, then a finite state-recoloring without lookahead (resp., with bounded, rational lookahead), and finally another regular tree morphism. Conversely, both finite-state recolorings without lookahead (resp., with bounded, rational lookahead) and regular tree morphisms can be thought of as special cases of tree transducers without lookahead (resp., with bounded, rational lookahead). Taking advantage of such relationships, we can exploit Theorem 11 to prove that the class of reducible trees is effectively closed with respect to top-down deterministic tree transducers without lookahead as well as with bounded lookahead. We expect the proof of the closure property for reducible trees under finite-state recolorings with bounded lookahead to be extendable to the more general case of finite-state recolorings with rational lookahead.

As for tree insertions, we have the following result.

Proposition 5. *Let $\mathcal{A}$ be a tree automaton and γ a regular tree insertion. For any given $\mathcal{A}, A$-type t, one can compute an $\mathcal{A}, A$-type t' such that, for every A-augmented tree U, if t is an $\mathcal{A}, A$-type of U, then t' is an $\mathcal{A}, A$-type of $\gamma(U)$.*

Proposition 5 states that for every regular tree insertion γ and every tree automaton $\mathcal{A}$, there exists a computable function $\gamma^{\mathcal{A}} : \mathscr{T}_{\mathcal{A},A} \rightarrow \mathscr{T}_{\mathcal{A},A}$ that maps any $\mathcal{A}, A$-type of a tree U to an $\mathcal{A}, A$-type of $\gamma(U)$. The function $\gamma^{\mathcal{A}}$ is said an *abstraction* of the regular tree insertion γ. Moreover, if we denote by $Abst_{\mathcal{A},A}$ the set of all abstractions of regular tree insertions and we endow it with the operation of functional composition, then we can give $Abst_{\mathcal{A},A}$ the status of *finite monoid*. Indeed, we have that (i) $Abst_{\mathcal{A},A}$ is a finite set (since there are only finitely many $\mathcal{A}, A$-types in $\mathscr{T}_{\mathcal{A},A}$), (ii) tree insertions are closed under functional composition (this follows from associativity of second-order tree substitutions), (iii) abstractions of regular tree insertions are closed under functional composition (since regular tree insertions map regular trees to regular trees), and (iv) there exists an abstraction id^M playing the role of the identity in $Abst_{\mathcal{A},A}$.

On the effectiveness of the method. The class of reducible trees obviously includes all regular trees; moreover, it includes a number of non-regular ones, such as, for instance, the unfoldings of context-free graphs (algebraic trees), Cachat tree generators, and various deterministic trees outside Caucal hierarchy. In the following we give a meaningful example of reducible trees (other examples can be found in [58]).

To start with, we consider the well-known example of the semi-infinite line. Let $L = (\mathbb{N}, E_a, E_b, E_c)$ be the semi-infinite line with a-labeled forward edges, b-labeled backward edges and c-labeled loops (see the top part of Figure 5.10). Let T_L be the unfolding of L from the leftmost vertex. The bottom left part of Figure 5.10 depicts the tree T_L, where, for each $i \in \mathbb{N}$, F_i denotes the unfolding from the rightmost vertex of the subgraph L_i obtained by restricting L to set of vertices $\{0, ..., i-1\}$. We give an alternative proof of the decidability of the MSO-theory of T_L, which exploits the closure properties

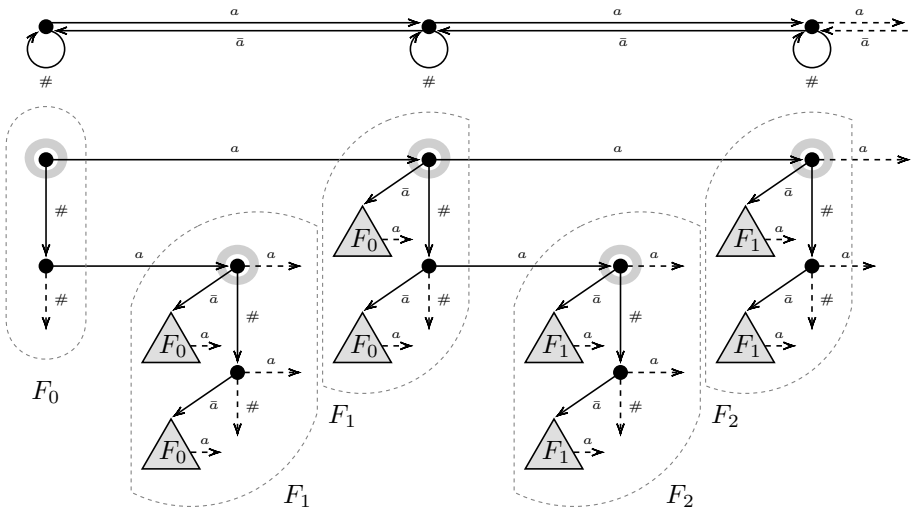

Fig. 5.10. Unfolding of the semi-infinite line.

of reducible trees instead of the MSO-compatibility of the unfolding operation. The idea is to give an inductive definition of the components $F_0, F_1, F_2, ...$ that allows us to prove that T_L is a rank 1 tree. By construction, every vertex v of T_L corresponds to a *unique* path π in L. We denote by $\vec{\pi}$ the last vertex of L along the path π and we define the factorization Π of T with respect to A by letting $\mathcal{D}om(\Pi)$ be the set of all vertices v of T_L such that there is no a proper ancestor v' of v for which $\vec{\pi}_v = \vec{\pi}_{v'}$, where π_v (resp., $\pi_{v'}$) denotes the unique path in L that corresponds to v (resp., v') (the set $\mathcal{D}om(\Pi)$ is represented in Figure 5.10 by circled nodes). Then, we label the resulting edges of Π with a single symbol a. Even though Π has unbounded degree, by identifying access paths with the same length, we can obtain a deterministic A-labeled retraction R_L of T_L with respect to Π. Let F and G be the two A-augmented trees depicted in the right part of Figure 5.10 and let γ be the regular tree insertion specified by G (the vertices that must be substituted with the input of γ are colored with x). Then, we set $F_i = \gamma^i(F)$, for every $i \in \mathbb{N}$. It is easy to see that, for every vertex u of Π at distance i from the root, the marked factor of T_L in u with respect to Π is isomorphic to the tree F_i. Thus, the tree R_L such that (i) $\mathcal{D}om(R_L) = \{a\}^*$, (ii) $R_L(\varepsilon)$ is the basic $\mathcal{A}, A$-type of F_0, and (iii) $R_L(a^{i+1}) = \gamma^{\mathcal{A}}(R_L(a^i))$, for all $i \in \mathbb{N}$, is a retraction of T_L with respect to $\mathcal{A}$ and Π. Moreover, R_L is a regular tree (this follows from the Pigeonhole Principle, since $\mathscr{T}_{\mathcal{A},A}$ is a finite set). Hence T_L is a rank 1 tree, whose footprint can be effectively computed, and, from Theorem 10, T_L enjoys a decidability MSO-theory.

The proof of the decidability of the MSO-theory of the unfolding of the semi-infinite line can be generalized to reducible trees of rank higher than 1. Consider the following class of *tree generators* for the Caucal hierarchy [20]. For any $n \in \mathbb{N}$, the level n tree generators are obtained from the regular trees via n-fold iterations of unfolding with backward edges and loops (we denote such an operation with $\mathcal{B}ack\mathcal{U}nf$). By exploiting the technique used to deal with the unfolding of the semi-infinite line, it is possible to prove the following closure property [58]).

Theorem 12. *Given a rank n tree T, $\mathcal{B}ack\mathcal{U}nf(T)$ is a rank $n+1$ tree, and thus the class of reducible trees is effectively closed under $\mathcal{B}ack\mathcal{U}nf$.*

From Theorem 12 it immediately follows that level n tree generators are rank n trees, and thus their MSO-theories are decidable.

On the relationships with Caucal hierarchy. The results about tree generators establish a connection between reducible trees and deterministic trees in the Caucal hierarchy. Every deterministic tree in the $n+1$-th level of the Caucal hierarchy (notice that the first level consists of all regular trees) can indeed be obtained from a level n tree generator via a suitable inverse rational mapping [20]. Without loss of generality, in this specific setting, we can assume that inverse rational mappings are specified by functions of the form $h : B \rightarrow \mathscr{P}(A^+)$, where A is the label set of the tree generator, B is the label set of the resulting structure, and for every $b \in B$, $h(b)$ is a rational language consisting only of non-empty words over A. From a result of Colcombet and Löding [23], any inverse rational mapping specified by a function h of the above form can be implemented via a tree transducer with *rational lookahead*. Hence, every deterministic tree in the level $n+1$ of the Caucal hierarchy can be obtained from a level n tree generator (a reducible tree) via a tree transducer with rational lookahead. Since tree transducers with rational lookahead are subsumed by finite-state recolorings with rational lookahead and regular tree morphisms, we have that, if the class of rank n trees were closed under finite-state recolorings with rational lookahead, then reducible trees would capture *all* deterministic trees in the Caucal hierarchy. As a matter of fact, we already know that rank n trees are closed under finite-state recolorings with bounded lookahead. Moreover, if h is a finite mapping, namely, $h(b)$ is a finite language for every $b \in B$, then the inverse rational mapping specified by h (inverse *finite* mapping) can be implemented via a tree transducer with *bounded lookahead*. This implies that reducible trees capture all deterministic trees obtained by iterating unfoldings and inverse finite mappings, starting from regular trees. Such a class of trees is properly included, starting from level 3, in the Caucal hierarchy (as an example, the tree whose maximal paths are all and only the words of the form $w \cdot w$, with $w \in A^*$, belongs to the 3-rd level of the Caucal hierarchy and it cannot be obtained via inverse finite mappings and unfoldings starting from regular trees).

6 Rational and automatic graphs: properties and decidability

Rational graphs have been first introduced by Morvan in [59] as a strict extension of the families of context-free graphs [62], regular (equational) graphs [25], and prefix recognizable graphs [19, 21]. To define these graphs conveniently, we identify their vertices with the finite words from a given alphabet A and we specify their edges via rational languages consisting of pairs of finite words.

6.1 Word transducers

In order to formally define languages consisting of pairs of words, we need to introduce the notion of (word) transducer. A (word) transducer is a finite-state device that transforms an input word into an output word. It can be equivalently thought of as a finite-state automaton accepting pairs of words, rather than single words.

Definition 19. *A (*word*)* transducer *is a 4-tuple $\mathcal{T} = (S, X, \delta, I, F)$, where*
- *S is a finite set of states,*
- *X is a finite alphabet,*
- *$\delta : S \times X^* \times X^* \times S$ is a transition relation,*
- *I (resp. F) is the set of initial (resp. final) states.*

The transducer $\mathcal{T}$ *accepts* the pair (u, v) of words over X iff there exist (i) a factorization $u = u_1 u_2 ... u_n$, (ii) a factorization $v = v_1 v_2 ... v_n$, and (iii) a sequence of states $s_0, s_1, ..., s_n$ such that
- $s_0 \in I$,
- $s_n \in F$,
- for all $1 \leq i \leq n$, $(s_{i-1}, u_i, v_i, s_i) \in \delta$.

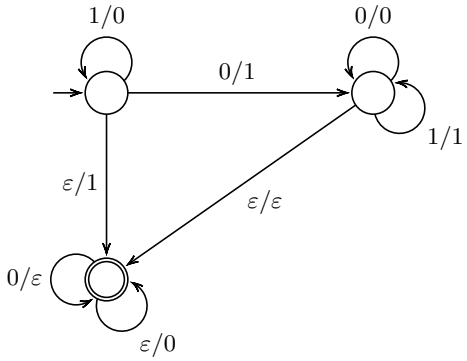

Fig. 6.11. An example of word transducer.

As an example, the transducer depicted in Figure 6.11 recognizes the set of all (reversed) binary encodings of pairs of numbers of the form $(n, n+1)$, with $n \in \mathbb{N}$.

Notice that any transducer can be put in a *normalized form*, which requires that $|u|+|v| = 1$ holds for every transition $(s, u, v, s') \in \delta$. Note that, even with transducers in normalized form, either u or v can be the empty word. This makes it possible for a transducer to accept pairs of words with different lengths; moreover, it is easy to see that there is no bound on the size difference between the input and the output in a generic transducer. A more restricted form of transducer, called *synchronized transducer*, is obtained by enforcing $|u| = |v| = 1$ for every transition $(s, u, v, s') \in \delta$, namely, by requiring that the transducer must produce exactly one symbol in output for each symbol read in input. This forces the language recognized by a synchronized transducer to consist only of pairs of words of the same length.

We say that a binary relation $R \subseteq X^* \times X^*$ between finite words is *rational* if it can be recognized by a word transducer. It is easy to see that the domain set and the image set of a rational relation are rational languages. Moreover, it is easy to prove that rational relations are closed under intersection and component-wise concatenation, namely, if R_1 and R_2 are rational relations, then the relation $R_1 \cap R_2$ and the relation $R_1 \cdot R_2 = \{(u_1 \cdot u_2, v_1 \cdot v_2) : (u_1, v_1) \in R_1, (u_2, v_2) \in R_2\}$ are rational. Finally, it should be noted that the notion of rational relation can be easily generalized to the case of a relation with an arbitrary arity (in such a case the transducer has transitions belonging to $S \times (X^*)^k \times S$, where k is the arity of the relation).

6.2 Internal presentations of rational and automatic graphs

In this section, we define rational and automatic graphs via word transducers.

Definition 20. *A (Λ-labeled)* rational graph $G = (V, (E_a)_{a \in A})$ *is defined by a tuple of (unrestricted) transducers $(\mathcal{T}_a)_{a \in A}$ as follows:*

- *E_a is the set of all pairs of the form $(u, v) \in X^* \times X^*$ that are accepted by $\mathcal{T}_a$,*
- *$V = \{u \in X^* : \exists\, v \in X^*, a \in \Lambda.\ (u, v) \in E_a\} \cup \{v \in X^* : \exists\, u \in X^*, a \in \Lambda.\ (u, v) \in E_a\}$.*

Figure 6.12 shows an example of rational graph, the *infinite grid*, together with the transducers that define its edges (note that the vertices of the graph are finite words over $\{x, y\}$ while the edges are labeled over $\{a, b\}$). It can be easily showed that the infinite grid is in fact an automatic graph, namely, it is possible to extend the words defining its vertices by padding symbols and then encode the edges by synchornized transducers.

If the transducers that define a rational graph are synchronized, then the rational graph is said to be *synchronized rational* or *automatic*. It should be noted that the languages recognized by synchronized

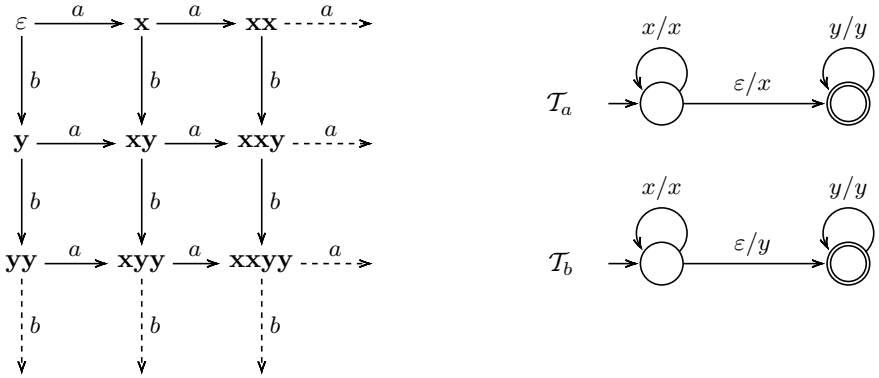

Fig. 6.12. The infinite grid and its associated transducers.

transducers are strictly contained int the languages recognized by generic transducers. However, this does not necessarily imply that the family of automatic graphs is strictly contained in the family of rational graphs. In order to separate the family of automatic graphs from that of rational graphs, one can reason on the growth rate of the out-degree of vertices in the case of graphs with finite out-degree (notice that both rational and automatic graphs can have infinite out-degree). Precisely, it can be proved (see [60]) that for any rational graph G of finite out-degree and for any vertex x in G, there exists $c \in \mathbb{N}$ such that the out-degree of the vertices at distance n of x is at most c^{c^n}. Such an upper bound can be actually reached: consider, for instance, the rational graph G_{rat} specified by a single transducer $\mathcal{T} = (S, X, \delta, I, F)$, where $S = I = F = \{q\}$, $X = \{x, y\}$, and $\delta = \{(q, \alpha, \beta\gamma, q) : \alpha, \beta, \gamma \in X\}$. On the other hand, in the case of automatic graphs of finite out-degree, the upper bound turns out to be simply exponential: for any automatic graph G of finite out-degree and for any vertex x in G, there exists $c \in \mathbb{N}$ such that the out-degree of the vertices at distance n of x is at most c^n [60]. This shows that the graph G_{rat} is rational but not synchronized and, in particular, that the family of automatic graphs is a strict sub-family of that of rational graphs.

6.3 External presentations of rational and automatic graphs

Recall that *prefix recognizable graphs* [19, 21] can be obtained from the infinite binary complete tree via *inverse rational mappings* followed by *rational restrictions*. Here we characterize the family of *rational graphs* in terms of *inverse linear mappings* and *rational restrictions* of the infinite binary complete tree.

We start with some preliminary definitions.

We denote by T_2 the infinite binary complete tree and we assume that its edges are labeled over the set $A = \{1, 2\}$. Given a context-free mapping $h : B \to \mathscr{P}((A \cup \bar{A})^*)$, we say that h is *linear* if every set $h(b)$ is a context-free linear language, namely, a language generated by a grammar $\mathcal{G}_b$ with at most one non-terminal symbol on the right-hand side of each production rule. If all production rules of each grammar $\mathcal{G}_b$ are either of the form $p \to \bar{u}\, q\, v$ or $p \to \varepsilon$, where p and q are non-terminal symbols and $u, v \in \{1, 2\}^*$, then h is said to be a *special linear mapping*. Moreover, if $|u| = |v|$ holds for every production of the form $p \to \bar{u}\, q\, v$, then the mapping is said to be *synchronized*.

Now, let $G = (V, (E_b)_{b \in B})$ be a graph whose domain consists of finite words over an alphabet X. We define the *L-restriction* of G, where $L \subseteq X^*$, the graph $G|_L = (L, (E'_b)_{b \in b})$, where $E'_b = E_b \cap L \times L$. A *rational* restriction is an L-restriction, where L is a rational language. Notice that a restriction is applicable to any graph that results from an inverse mapping of T_2 (i.e., the infinite complete binary tree).

As an example, consider the infinite complete binary tree T_2, the special linear mapping h that maps a to $\{\bar{2}^n \cdot 1 \cdot 2^n : n \in \mathbb{N}\}$ and b to $\{2\}$, and the rational restriction specified by the language $L = \{1\}^* \cdot \{2\}^*$. It is easy to see that the graph $h^{-1}(T_2)|_L$ (see Figure 6.13) is isomorphic to the infinite grid, which is a rational graph.

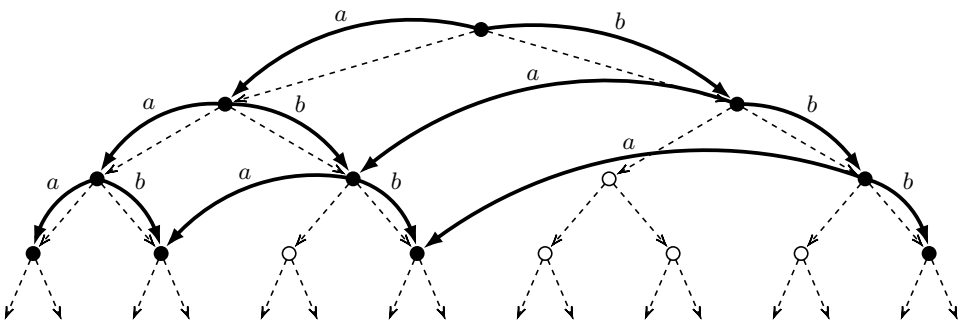

Fig. 6.13. The grid as a result of an inverse special linear mapping and a rational restriction.

In the following, we show that every rational graph can be obtained from the infinite binary complete tree by means of an inverse special linear mapping followed by a rational restriction. Also the converse result holds, thus providing an alternative (external) characterization of rational graphs.

Proposition 6 (Morvan [59]). *Any rational graph G can be obtained from T_2 via an inverse special linear mapping followed by a rational restriction.*

Proof. Let G be rational graph specified by a tuple of transducers $(\mathcal{T}_a)_{a\in A}$. For the sake of simplicity, we assume that the transducers $(\mathcal{T}_a)_{a\in A}$ work on the alphabet $X = \{1, 2\}$ (this is not a restriction since we can easily embed any non-binary tree into T_2). We denote by $d : X^* \to X^*$ the morphism that maps 1 to 11 and 2 to 22 and by $r(u)$ the reversal of a finite word u. Now, for each transducer $\mathcal{T}_a = (S_a, X, \delta_a, I_a, F_a)$, with $a \in A$, we define a new transducer $\mathcal{T}'_a = (S'_a, X, \delta'_a, I'_a, F'_a)$ such that

- $S'_a = S_a \cup \{s_{0,a}\}$, where $s_{0,a}$ is a fresh state not belonging to S_a,
- $\delta'_a = \{(s, d(u), d(v), s') : (s, u, v, s') \in \delta_a\} \cup \{(s_{0,a}, 12, 12, s) : s \in I_a\}$,
- $I'_a = \{s_{0,a}\}$,
- $F'_a = F_a$.

It is easy to check that the graph G' generated by $(\mathcal{T}'_a)_{a\in A}$ is isomorphic to G. We now show that there is a special linear mapping and a rational restriction that generate exactly G' from T_2.
For each transducer $\mathcal{T}'_a$, we define the linear grammar $\mathcal{G}_a$ as follows. We let the non-terminal symbols of $\mathcal{G}_a$ be all and only the states $s \in S_a$ of $\mathcal{T}'_a$ and we let $s' \to r(\bar{u})\, s\, v$ be a production of $\mathcal{G}_a$ for each transition (s, u, v, s') of $\mathcal{T}'_a$. We then add the production $s_{0,a} \to \varepsilon$, where $s_{0,a}$ is the (unique) initial state of $\mathcal{T}'_a$. We define:

- L_a as the union, over all final states $s \in F_a$, of the language generated by $\mathcal{G}_a$ starting from the non-terminal symbol s (notice that $L_a \subseteq (X \cup \bar{X})^*$),
- $h : A^* \to \mathscr{P}((X \cup \bar{X})^*)$ as the mapping such that $h(a) = L_a$,
- L as the language consisting of all vertices of G'.

Clearly, L_a is a finite union of linear languages and hence it is a linear language as well. Moreover, it is easy to see that (u, v) is an a-labeled edge of G' iff $r(\bar{u}) \cdot v \in L_a$. As regards the language L, it is a finite union of domains and images of rational relations and hence it is rational. We now show that $G' = h^{-1}(T_2)|_L$, which implies that G is isomorphic to $h^{-1}(T_2)|_L$. We assume that the vertices of T_2 are identified by the sequences of labels of their access paths (namely, the empty word ε identifies the root, 1 and 2 identify its successors, and so on).

Let (u, v) be an a-labeled edge of G'. Clearly, $u, v \in L$ and $r(\bar{u}) \cdot v \in h(a)$, from which it follows that (u, v) is an a-labeled edge of $h^{-1}(T_2)|_L$. Conversely, let (u, v) be an a-labeled edge of $h^{-1}(T_2)|_L$. Clearly, $u, v \in L$ and there are words $w, u_0, v_0 \in X^*$ such that (i) $u = w \cdot u_0$, (ii) $v = w \cdot v_0$, and (iii) $r(\bar{u}_0) \in L_a$. Now, since u, u_0, v, v_0 belong to L, their lengths are even integers and hence the length of w is also even. By construction, every word in L starts with 12 and then is a succession of 11's and 22's. These facts imply that w must be the empty word ε and hence $u_0 = u$ and $v_0 = v$, from which it follows that (u, v) is an a-labeled edge of G'. □

Proposition 7 (Morvan [59]). *Let $X = \{1, 2\}$ be the set of edge labels of the infinite binary complete tree T_2, $h : A \to \mathscr{P}((X \cup \bar{X})^*)$ a special linear mapping, and $L \subseteq X^*$ a rational language. The graph $G = h^{-1}(T_2)|_L$ is rational.*

Proof. By definition of special linear mapping, we have that $h(a) \subseteq \bar{X}^* \cdot X^*$. We now fix $a \in A$ and denote by $\mathcal{G}_a$ the grammar that generates $h(a)$ starting from an initial non-terminal symbol $p_{0,a}$. Suppose that the production rules of $\mathcal{G}_a$ are either of the form $p \to \bar{u}\, q\, v$ or $p \to \varepsilon$, where p and q are non-terminal symbols and $u, v \in X^*$. We define the transducer $\mathcal{T}_a = (S_a, X, \delta_a, I_a, F_a)$ as follows:

- S_a is the set of all non-terminal symbols of the grammar $\mathcal{G}_a$,
- δ_a is the set of all transitions $(q, r(u), v, p)$ such that $p \to \bar{u}\, q\, v$ is a production of $\mathcal{G}_a$ ($r(u)$ denotes the reversal of u),
- I_a is the set of all non-terminal symbols p such that $p \to \varepsilon$ is a production rul in $\mathcal{G}_a$,
- $F_a = \{p_{0,a}\}$ (namely, the unique final state of $\mathcal{T}_a$ is the initial non-terminal symbol of $\mathcal{G}_a$).

One can easily verify that, for every $u, v \in X^*$, $(r(u), v)$ is accepted by $\mathcal{T}_a$ iff $r(\bar{u})v \in h(a)$. This implies that the rational graph G_1 generated by the tuple of transducers $(\mathcal{T}_a)_{a \in A}$ is a subgraph of $h^{-1}(T_2)$. Indeed, every edge (u, v) in G_1 corresponds to a path in T_2 that goes from a vertex u to the root and from the root to a vertex v).
On the grounds of G_1, we can define another graph G_2 such that $G_2 = h^{-1}(T_2)$: if suffices to let the edges of G_2 be all and only the pairs of the form $(w \cdot u, w \cdot v)$, where $w \in X^*$ and (u, v) is an edge of G_1. It is not difficult to show that G_2 is a rational graph as well. Moreover, we can show that $G_2 = h^{-1}(T_2)$. Indeed, if (u, v) is an a-labeled edge of G_2, then there are $w, u_0, v_0 \in X^*$ such that (i) $u = w \cdot u_0$, (ii) $v = w \cdot v_0$, and (iii) (u_0, v_0) is an a-labeled edge of G_1. Since G_1 is a subgraph of $h^{-1}(T_2)$, this implies that $r(\bar{u}_0)v_0 \in h(a)$ and hence T_2 contains a $r(\bar{u}_0)$-labeled path from u_0 to the root and a v-labeled path from the root to v_0. The same holds if we consider the subtree of T_2 rooted at w instead of T_2. Therefore, $(u, v) = (w \cdot u_0, w \cdot v_0)$ is an a-labeled edge of $h^{-1}(T_2)$. The converse implication works exactly in the same way.
It remains to show that the *L-restriction* of $G_2 = h^{-1}(T_2)$ is a rational graph, but this follows easily since $G_2|_L$ can be obtained by intersecting each edge relation of G_2 with $L \times L$ and since rational relations (namely, relations defined by word transducers) are closed under intersection. □

As regards automatic graphs, it is straightforward to modify the above proofs and show that a graph is automatic iff it can be obtained from the infinite binary complete tree via an *inverse synchronized linear mapping* followed by a rational restriction.

6.4 Decidability properties of rational and automatic graphs

In this section we investigate the decidability of crucial problems (e.g., satisfiability for first-order logic) involving rational and automatic graphs. The first is a negative result, stating that the satisfiability problem for first-order logic over rational graphs is undecidable.

Proposition 8 (Morvan [59]). *The problem of establishing, given a rational graph G and a first-order formula ψ, whether ψ is satisfiable in G is undecidable.*

Proof. We reduce the Post's Correspondence Problem [48], shortly PCP, to the satisfiability problem for FO logic over rational graphs. A PCP-instance is a pair $(\bar{u}, \bar{v})$ of word lists, where $\bar{u} = (u_1, ..., u_k)$ and $\bar{v} = (v_1, ..., v_k)$. Such an instance is positive if there are n-indices $i_1, ..., i_n$, with $n \geq 1$, such that $u_{i_1} \cdot u_{i_2} \cdot ... \cdot u_{i_n} = v_{i_1} \cdot v_{i_2} \cdot ... \cdot v_{i_n}$. It is known that the halting problem for Turing machines is reducible to the PCP and hence the latter is an undecidable problem. In order to reduce the PCP to the satisfiability problem for FO logic over rational graphs, we consider a generic PCP-instance $(\bar{u}, \bar{v}) \in (X^*)^k \times (X^*)^k$ and we define the transducer $\mathcal{T}_{\bar{u},\bar{v}} = (S, X, \delta, I, F)$, where:

- $S = \{s_0, s_1\}$,
- $\delta = \{(s_0, u_i, v_i, s_1) : 1 \leq i \leq k\} \cup \{(s_1, u_i, v_i, s_1) : 1 \leq i \leq k\}$,
- $I = \{s_0\}$,
- $F = \{s_1\}$.

Clearly, the rational graph $G_{\bar{u},\bar{v}}$ generated by $\mathcal{T}_{\bar{u},\bar{v}}$ contains an edge from a vertex x back to the same vertex x iff the PCP-instance $(\bar{u}, \bar{v})$ is positive. In particular, the PCP-instance $(\bar{u}, \bar{v})$ is positive iff the formula $\psi = \exists\, x.\ (x, x) \in E$ is satisfiable in $G_{\bar{u},\bar{v}}$. □

The above result can be refined by showing that there is a *single* rational graph G with an undecidable first-order theory.

Theorem 13 (Thomas [71]). *There is a single rational graph G for which the problem of establishing whether a given first-order formula ψ is satisfiable in G is undecidable.*

Proof. In order to obtain a single rational graph with an undecidable first-order theory, we refine the construction provided in the proof of Proposition 8. We use a universal Turing machine M and the encoding of its undecidable halting problem (for different input words w) 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 [48]): a Turing machine M with input word w is converted into a PCP-instance $(\bar{u}, \bar{v})$, where $\bar{u} = (u_1, ..., u_k)$, $\bar{v} = (v_1, ..., v_k)$ and $u_i, v_i \in X^*$, with X consisting of states and tape letters of M plus a special symbol # (for the separation between the M-configurations in a computation). If the input word is $w = a_1...a_m$, then u_1 is set to be the initial configuration word $u(w) = \#q_0a_1...a_m$ of M; furthermore, we have always $v_1 = \#$ and $u_2, ..., u_k, v_2, ..., v_k$ only depend on M. Then the standard construction ensures that M halts on input w iff the PCP-instance $(\bar{u}, \bar{v})$ has a *special* solution, namely, an index sequence $(i_2, ..., i_n)$ such that $u(w) \cdot u_{i_2} \cdot ... \cdot u_{i_n} = \# \cdot v_{i_2} \cdot ... \cdot v_{i_n}$. Let G_0 be the graph obtained by following a construction in analogy to that of Proposition 8. The vertices of G_0 are words over the alphabet X and the edges are all pairs (x, y) for which there exist an input word w and some indices $i_2, ..., i_n$ such that $x = u(w) \cdot u_{i_2} \cdot ... \cdot u_{i_n}$ and $y = \# \cdot v_{i_2} \cdot ... \cdot v_{i_n}$. Clearly, G_0 is rational and it contains an edge from a word x to itself iff there is a special solution of some PCP-instance $((u(w), u_2, ..., u_k), (\#, v_2, ..., v_k))$. In order to address the input words x explicitly in the graph G_0, we add further vertices and edge relations E_a, for $a \in X$. A $u(w)$-labeled path via the vertices will lead to a vertex of G_0 with prefix $u(w)$; if the latter vertex has an edge back to itself then a special solution for the PCP-instance $((u(w), u_2, ..., u_k), (\#, v_2, ..., v_k))$ can be inferred. The new vertices are words over a copy $\underline{X}$ of the alphabet X, consisting of underlined versions of the X-letters. For any word $u(w)$ we shall add the vertices which arise from the underlined versions of the proper prefixes of $u(w)$, and we introduce an a-labeled edge from any such underlined word $\underline{z}$ to $\underline{z} \cdot \underline{a}$ (including the case $z = \varepsilon$). There are also edges to non-underlined words: we have an a-labeled edge from the underlined version of z to any non-underlined word which has $z \cdot a$ as a prefix. We denote by G the resulting graph. It is easy to prove that G is rational. Moreover, the PCP-instance $((u(w), u_2, ..., u_k), (\#, v_2, ..., v_k))$ has a special solution iff G contains a path, labeled by $u(w)$, from the vertex ε to a vertex which has an edge back to itself. Such a condition can be expressed by a suitable first-order formula ψ_w, that is defined on the grounds of the input word w. Thus, we have that the Turing machine M halts on input w iff G satisfies ψ_w. In this way, we proved that there is a graph, G, whose first-order theory is undecidable. □

We now consider the satisfiability problem for first-order logics interpreted over automatic graphs, which form a strict sub-family of rational graphs.

Theorem 14 (Hodgson [47], Khoussainov and Nerode [51]). *The satisfiability problem for first-order logic over automatic graphs is decidable.*

Proof. The proof is based on an idea which was first proposed by Büchi in [5]. Let $G = (V, (E_a)_{a \in A})$ be an automatic graph, generated by a tuple of synchronized transducers $(\mathcal{T}_a)_{a \in A}$ working on an alphabet X. We then consider a generic first-order formula $\psi(x_1, ..., x_k)$ with free variables and the k-ary relation $R_\psi = \{(v_1, ..., v_k) \in V^k : G \vDash \psi[v_1, ..., v_k]\}$. We show that each such R_ψ is a synchronized rational relation, namely, it can be recognized by a synchronized word transducer $\mathcal{T}_\psi$ working on k-tuples of words. For the sake of simplicity, we think of a synchronized transducer as a standard finite-state automaton working on a suitable alphabet. Precisely, we view a k-tuple of words $(w_1, ..., w_k)$ over the alphabet X as a single word w over the alphabet $(X \cup \{\$\})^k$, where $w[i] = (w_1[i], ..., w_k[i])$ and \$ is used to fill up the words $w_1, ..., w_k$ at the end to achieve equal length. Thus, a synchronized transducer defining a k-ary relation $R_\psi \subseteq (X^*)^k$ can be viewed as a finite-state automaton recognizing a language L_ψ over the alphabet $(X \cup \{\$\})^k$. We show by induction on the structure of ψ that there exist such an automaton $\mathcal{T}_\psi$ recognizing the language L_ψ, provided that G is an automatic graph.
The case for relations defined by atomic formulas of the form $x = y$ (resp. $(x, y) \in E_a$)) is trivial: we simply let $\mathcal{T}_\psi$ be the automaton recognizing the language $\{(w, w) : w \in V\}$, which is rational since V is rational (resp. the automaton $\mathcal{T}_a$). As for the induction steps, involving the connectives $\neg$, $\vee$, and the existential quantifier $\exists$, one uses closures of finite-state automata under boolean operations and projection. As for the connective $\vee$, it should be noted that, in general, $L_{\psi_1 \vee \psi_2}$ does not coincide with $L_{\psi_1} \cup L_{\psi_2}$ (in fact, the union may also be undefined). This happens when some free variables in two given first-order formulas, e.g. $\psi_1(x_1, ..., x_k, z_1, ..., z_m)$ and $\psi_2(y_1, ..., y_h, z_1, ..., z_m)$, are different. In order to correctly define the language $L_{\psi_1 \vee \psi_2}$, we need to preliminary perform the so-called cylindrification of L_{ψ_1} and L_{ψ_2}, which yields the languages

$$L'_{\psi_1} = \{(u_1, ..., u_k, v_1, ..., v_h, w_1, ..., w_m) : (u_1, ..., u_k, w_1, ..., w_m) \in L_{\psi_1}, (v_1, ..., v_h) \in (X \cup \{\$\})^h\},$$
$$L'_{\psi_2} = \{(u_1, ..., u_k, v_1, ..., v_h, w_1, ..., w_m) : (v_1, ..., v_h, w_1, ..., w_m) \in L_{\psi_2}, (u_1, ..., u_h) \in (X \cup \{\$\})^k\}.$$

Finite-state automata are easily proved to be closed under cylindrification. Moreover, the language $L_{\psi_1 \vee \psi_2}$ can be computed as $L'_{\psi_1} \cup L'_{\psi_2}$, which completes the proof. □

We just proved that the first-order theory of an automatic graph is decidable. However, as soon as we try to extend first-order logic with reachability or transitive closure operators, we get undecidability. In particular, this implies that the monadic second-order theories of automatic graphs are undecidable as well. Other examples of problems that turn out to be undecidable over automatic graphs are the problem of testing whether two given automatic graphs are isomorphic and the problem of testing whether a given automatic graph is connected (proofs can be found in [10]).

Proposition 9. *The reachability problem over automatic graphs is undecidable.*

Proof. Let M be a generic Turing machine. We show that the transition graph of M is automatic. As a consequence, we will obtain that the reachability problem for automatic graphs (to which the halting problem for Turing machines reduces) is undecidable. We encode an M-configuration having state q, tape contents w and head position p, by the word $w_0\, q\, w_1$, where $|w_0| = p$ and $w_0 \cdot w_1 = w$. At each transition $w_0\, q\, w_1 \xrightarrow[M]{} w'_0\, q'\, w'_1$, only the symbols around the state are changed. Thus, the set of all pair of words $(w_0\, q\, w_1, w'_0\, q'\, w'_1)$ representing a M-transitions (or, equivalently, edges of the transition graph of M) can be recognized by a suitable word transducer. □

We conclude this section with some further remarks and possible generalizations of automatic graphs (we also refer the interested reader to [8, 51]).

We already mentioned that the notion of word transducer can be generalized to allow specifications of k-ary relations on words, with k arbitrary. This makes it possible to introduce the so-called *automatic structures*, that are structures of the form $(V, (R_i^{(k_i)})_{i \in I})$, where V is a set of words over an alphabet X and each $R_i^{(k_i)}$ is a k_i-ary relation over X^*. Clearly, automatic graphs are special cases of automatic structures (where all relations are binary). However, the proof of Theorem 14 works fine also for the case of automatic structures, which shows that automatic structures enjoy a decidable first-order theory. A notable example of automatic structure is $(\mathbb{N}, +)$, where $+$ is the usual addition operator for natural numbers (thought of as a ternary relation). Indeed, one can represent natural numbers by their reversed binary encodings and then specify the relation $+$ by a suitable synchronized transducer. Thus, the decidability of the first-order theory of $(\mathbb{N}, +)$ implies the decidability of the Presburger arithmetic [46]. Such a result can be easily generalized to the expanded (automatic) structure $(\mathbb{N}, +, |_p)$, where $|_p$ is the binary relation defined, for some $p \geq 2$, by $x \ |_p \ y$ iff x is a power of p dividing y (using p-ary encodings rather than binary encodings one can easily build suitable transducers representing the relations $+$ and $|_p$).

Another possible generalization of automatic structures is the notion of ω-automatic structure. The definition is analogous to the one for automatic structures except that the elements of an ω-automatic structure are represented by infinite words from some regular ω-language and the relations are recognizable by synchronized transducers working on infinite words (or, equivalently, by Büchi automata). It is clear that all automatic structures are ω-automatic. In fact, there exist some ω-automatic structures that encompass uncountably many elements, from which it follows that automatic structures are a strict sub-family of ω-automatic structures. Moreover, like automatic structures, ω-automatic structures enjoy a decidable first-order theory. Example of ω-automatic structures are $(\mathbb{R}, +)$ and its expansion $(\mathbb{R}, +, \leq, |_p, 1)$, where $|_p$ is now defined as $x \ |_p \ y$ iff $\exists \ n, k \in \mathbb{Z}. \ x = p^n \ \wedge \ y = kx$.

7 Reachability problems: issues and algorithms

There are two main different approaches for solving a (plain) reachability problem. The first one is the *forward reachability analysis*, which starts from the set I of source vertices and computes the transitive closure of the transition relation of the graph (often referred as the *Post* relation). It turns out that (I, F) is a positive instance of the reachability problem iff $Post^*(I) \cap F \neq \emptyset$, where $Post^*(I)$ is the limit of the infinite increasing sequence $\{X_i\}_{i \geq 0}$ given by $X_0 = I$ and $X_{i+1} = X_i \cup Post(X_i)$. The *backward reachability analisys*, instead, consists in repeatedly applying the inverse of the *Post* relation (denoted *Pre*) starting from the set F of target vertices. Clearly, (I, F) is a positive instance of the reachability problem iff $Pre^*(F) \cap I \neq \emptyset$, where $Pre^*(F)$ is the limit of the infinite increasing sequence $\{Y_i\}_{i \geq 0}$ given by $Y_0 = F$ and $Y_{i+1} = Y_i \cup Pre(Y_i)$. In the case of finite graphs, both sequences $\{X_i\}_{i \geq 0}$ and $\{Y_i\}_{i \geq 0}$ reach a fixed point in a finite number of steps, thus providing effective (but usually inefficient) solutions to the reachability problem. Unfortunately, this no longer happens for any interesting class of infinite graphs, where the infinite sequences $\{X_i\}_{i \geq 0}$ and $\{Y_i\}_{i \geq 0}$ often turn out to be strictly increasing. In order to guarantee termination, suitable convergence acceleration techniques are used. Intuitively, they are based on over-approximations of the *Post* and the *Pre* relations, which make the attractor construction effective over certain classes of infinite graphs (e.g., the pushdown transitin graphs and the transition graphs of Petri nets).

Even though the forward and the backward approaches are somehow related and both of them can be applied to a number of different structures and problems (cf. [50, 1]), the latter method is usually preferred for its simplicity and robustness. As a matter of fact, it has been recognized that some algorithms which were developed for certain classes of systems (e.g. pushdown systems, Petri nets) and based on backward reachability analysis, could be generalized to cope with several natural extensions of those systems (e.g. higher-order pushdown systems [12], ground tree rewriting systems [55], timed Petri nets and vector addition systems [2]).

As regards the solutions to the variants of the plain reachability problem, it is worth to remark that there exist classes of graphs enjoying a decidable plain reachability problem but an undecidable recurrent reachability problem (e.g., transition graphs of lossy channels [3]), as well as graphs enjoying a decidable recurrent reachability problem but an undecidable constrained reachability problem (e.g., transition graphs of ground tree rewriting systems [55]). Thus, it is clear that each variant of the reachability problem should be tackled by adopting ad hoc techniques, possibly based on generalizations of the two basic approaches mentioned above. In the sequel, we shall focus our attention on backward reachability analysis only, by applying it to the plain reachability problem over pushdown transition graphs and over the transition graphs of Petri nets.

7.1 Reachability over pushdown transition graphs

In this section we tackle the (plain) reachability problem over pushdown graphs (i.e., transition graphs of ε-free pushdown systems, which are pushdown automata devoid of initial configurations) via backward reachability analysis. Since the addressed problem does not involve the edge labels of the transition graph, we can get rid of the input symbols of the pushdown automaton. Let $\mathcal{P} = (Q, \Gamma, \Delta)$ be an input-free ε-free pushdown system and let $G = (V, E)$ be the transition graph of $\mathcal{P}$. Note that the vertices of G are exactly the configurations of $\mathcal{P}$. Clearly, $((q, \gamma w), (r, vw)) \in E$ iff $(q, \gamma, v, r) \in \Delta$.

Definition 21. *The* predecessor function $Pre : \mathscr{P}(Q \times \Gamma^*) \to \mathscr{P}(Q \times \Gamma^*)$ *of a pushdown system* $\mathcal{P} = (Q, \Gamma, \Delta)$ *is defined as follows: given a set* $S \subseteq Q \times \Gamma^*$ *of configurations,* (q, w) *belongs to* $Pre(S)$ *iff there exist* $(r, v) \in S$ *such that* $((q, w), (r, v)) \in E$. *The reflexive and transitive closure of Pre is the function* Pre^* *defined by* $Pre^*(S) = \bigcup_{i \geq 0} Pre^i(S)$.

We identify the vertices in G with the finite words from the language $Q\Gamma^*$ and we symbolically represent a (possibily infinite) set of vertices by means of a *sequential finite-state automaton* over the alphabet $Q \cup \Gamma$. We then let $X_0 = F$ be the set of target vertices and X_{i+1}, for all $i > 0$, be the set $X_i \cup Pre(X_i)$. Clearly, $Pre^*(F)$ coincides with the least fixpoint $\bigcup_{i \geq 0} X_i$ of the infinite sequence $X_0, X_1, X_2, ...$ Notice that if $X_{i+1} = X_i$ holds for some $i \geq 0$, then it immediately follows that there is $i \geq 0$ such that $X_i = Pre^*(F)$. Unfortunately, it may happen that $X_0, X_1, X_2, ...$ is an infinite sequence of strictly increasing sets, thus implying that $X_i \subsetneq Pre^*(F)$ holds for every $i \geq 0$ (consider, for instance, the pushdown automaton with a single control state q, a single stack symbol γ, and a single transition $(q, \gamma, \varepsilon, q)$ and the set of target states $F = \{(q, \varepsilon)\}$).

The basic idea underlying the solution to the reachability problem over pushdown transition graphs is to compute the set $Pre^*(F)$ as the limit of another sequence of sets $Y_0, Y_1, Y_2, ...$, for which we can prove the following fundamental properties:

$$\exists\, i \geq 0. \quad Y_{i+1} = Y_i, \tag{2}$$

$$\forall\, i \geq 0. \quad X_i \subseteq Y_i, \tag{3}$$

$$\forall\, i \geq 0. \quad Y_i \subseteq \bigcup_{j \geq 0} X_j. \tag{4}$$

The first property implies that there always exists $i \geq 0$ such that $Y_{i+1} = Y_i$, thus ensuring the termination of the procedure that computes the least fixpoint of the sequence $Y_0, Y_1, Y_2, ...$ The second property ensures that the Y_i's are over-approximations of the X_i's, thus implying that the limit of the sequence $Y_0, Y_1, Y_2, ...$ includes (at least) the set $Pre^*(F)$. The third property ensures that only the elements of $Pre^*(F)$ are captured.

In the following, we define the sets Y_i by providing an effective construction of their automaton-based representations $\mathcal{A}_i$. Precisely, we build a finite-state automaton $\mathcal{A}_{i+1}$ representing the set Y_{i+1} on the grounds of a given automaton $\mathcal{A}_i$ representing the set Y_i. We start with an automaton $\mathcal{A}_0$ representing the set $Y_0 = F$ of target vertices. Without loss of generality, we can assume that

- $\mathcal{A}_0$ has a single initial state s_0, which is not final and has no incoming transitions;
- for each $q \in Q$, $\mathcal{A}_0$ contains a single q-labeled transition; such a transition connects the initial state s_0 to a state which is denoted by s_q;
- for each $q \in Q$, no other transitions, apart from (s_0, q, s_q), reach the state s_q.

The next automaton $\mathcal{A}_{i+1}$ is obtained by simply adding new transitions to the automaton $\mathcal{A}_i$, while preserving its states, as described in the following Definition 22 (note that the transitions that are added to $\mathcal{A}_i$ to lead $\mathcal{A}_{i+1}$ start from the immediate successors of the initial state). For the sake of brevity, we use $s \xrightarrow[\mathcal{A}_i]{w} s'$ to mean that there is a run of the automaton $\mathcal{A}_i$ that starts in s, ends in s', and reads the word w.

Definition 22. *Let $\mathcal{A}_i$ be an automaton representing the set Y_i of vertices, let s_0 be the (unique) initial state of $\mathcal{A}_i$, and let s_q, for each $q \in Q$, be the (unique) state such that (s_0, q, s_q) is a transition of $\mathcal{A}_i$. The automaton $\mathcal{A}_{i+1}$ representing the set Y_{i+1} is obtained from $\mathcal{A}_i$ by adding a new transition (s_q, γ, p) for each transition (q, γ, v, r) of $\mathcal{P}$ and for each state p such that $s_r \xrightarrow[\mathcal{A}_i]{v} p$, .*

We now show that the sets of vertices represented by the automata $\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2, ...$ define an increasing sequence that satisfy Properties 2, 3, and 4.

Lemma 1. *Let $\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2, ...$ be a sequence of automata satisfying Definition 22. The sequence $Y_0, Y_1, Y_2, ...$ of the sets represented by $\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2, ...$ satisfies Property 2.*

Proof. The claim is easily proved by noticing that $\mathcal{A}_{i+1}$ has the same (initial/final) states of $\mathcal{A}_i$ and the same or more transitions. Since any automaton with n states and m input symbols can have at most n^2m transitions, it immediately follows that $Y_{n^2m} = Y_{n^2m+1}$, where n is the number of states of $\mathcal{A}_0$ and $m = |Q| + |\Gamma|$. □

Lemma 2. *Let $\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2, ...$ be a sequence of automata satisfying Definition 22. The sequence $Y_0, Y_1, Y_2, ...$ of the sets represented by $\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2, ...$ satisfies Property 3.*

Proof. We prove the claim by using induction on i. The base case is trivial. Let us assume that $X_i \subseteq Y_i$ and prove that $X_{i+1} \subseteq Y_{i+1}$ follows. If a configuration $(q, \gamma w)$ of $\mathcal{P}$ belongs to X_{i+1} but not to X_i, then there must exist a configuration (r, vw) in X_i such that $(q, \gamma, v, r) \in \Delta$. Since the word rvw representing the configuration (r, vw) is accepted by the automaton $\mathcal{A}_i$, we have that

$$s_0 \xrightarrow[\mathcal{A}_i]{r} s_r \xrightarrow[\mathcal{A}_i]{v} s \xrightarrow[\mathcal{A}_i]{w} f,$$

where f is a final state of $\mathcal{A}_i$. Then, by construction,

$$s_0 \xrightarrow[\mathcal{A}_{i+1}]{q} s_q \xrightarrow[\mathcal{A}_{i+1}]{\gamma} s \xrightarrow[\mathcal{A}_{i+1}]{w} f,$$

which shows that the configuration $(q, \gamma w)$ belongs to Y_{i+1} as well. □

In order to prove Property 4, we need the following technical lemma:

Lemma 3. *Let $\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2, ...$ be a sequence of automata satisfying Definition 22. For every $i \geq 0$, if $s_0 \xrightarrow[\mathcal{A}_i]{q} s_q \xrightarrow[\mathcal{A}_i]{w} p$, then the pushdown graph G contains a path from (q, w) to another vertex (r, v), where $s_0 \xrightarrow[\mathcal{A}_0]{r} s_r \xrightarrow[\mathcal{A}_0]{v} p$.*

Proof. We prove a stronger result: if p coincides with an immediate successor s_t of the initial state, then not only the lemma holds, but also there is a path in G from (q, w) to the configuration (t, ε) (this means that we may take $r = t$ and $v = \varepsilon$). The proof is by induction on i.

The base case is trivial: we simply take $r = q$ and $v = w$. Notice that, since the automaton $\mathcal{A}_0$ has no transitions leading to the initial state or to a successor of it, if p coincides with an immediate successor s_t of the initial state, then we must have $w = \varepsilon$ and $p = s_q$ and this implies that G contains a path from (q, w) to (q, ε).
As for the induction step, let $i > 0$ and let ρ be a run satisfying $s_0 \xrightarrow[\mathcal{A}_i]{q} s_q \xrightarrow[\mathcal{A}_i]{w} p$. We prove the claim by using induction on the number k of times that ρ uses a transition of $\mathcal{A}_i$ that does not belong to $\mathcal{A}_{i-1}$. If $k = 0$, then ρ is a run of $\mathcal{A}_{i-1}$ as well and the claim follows trivially by inductive hypothesis. Otherwise, if $k > 0$, we can assume that $w = w'\gamma w''$ and

$$s_0 \xrightarrow[\mathcal{A}_i]{q} s_q \xrightarrow[\mathcal{A}_i]{w'} s_{q'} \xrightarrow[\mathcal{A}_i]{\gamma} p' \xrightarrow[\mathcal{A}_i]{w''} p,$$

where q' and p' are suitable states of $\mathcal{A}_i$ and the transition $(s_{q'}, \gamma, p')$ does not belong to $\mathcal{A}_{i-1}$ (notice that all transitions which are added to $\mathcal{A}_{i-1}$ to yield $\mathcal{A}_i$ must start from a successor of the initial vertex). Now, we consider the run of $\mathcal{A}_i$ on w' starting in s_q and ending in $s_{q'}$. From the inductive hypothesis, we know that G contains a path from (q, w') to (q', ε). Moreover, since the transition $(s_{q'}, \gamma, p')$ has been added to the automaton $\mathcal{A}_{i-1}$, there must exist a state $r' \in Q$ and a word $v' \in \Gamma^*$ such that $(q', \gamma, v', r') \in \Delta$ and

$$s_{r'} \xrightarrow[\mathcal{A}_{i-1}]{v'} p'.$$

From the latter property and from the main inductive hypothesis, we know that G contains a path from (r', v') to another vertex (r'', v''), where

$$s_{r''} \xrightarrow[\mathcal{A}_0]{v''} p'.$$

This implies that there is a run ρ' of $\mathcal{A}_i$ from $s_{r''}$ to p reading $v''w''$ and containing less than k applications of new transitions. Thus, by applying the inductive hypothesis on ρ', we know that G contains a path from $(r'', v''w'')$ to another vertex (r, v), where

$$s_r \xrightarrow[\mathcal{A}_0]{v} p.$$

Since we already know that G contains (i) a path from (q, w') to (q', ε), (ii) an edge from (q', γ) to (r', v'), and (iii) a path from (r', v') to (r'', v''), then G contains also a path from (q, w) to (r, v), where $s_0 \xrightarrow[\mathcal{A}_0]{r} s_r \xrightarrow[\mathcal{A}_0]{v} p$.
Assume now that p is an immediate successor s_t of the initial state s_0. Since

$$s_0 \xrightarrow[\mathcal{A}_i]{q} s_q \xrightarrow[\mathcal{A}_i]{w'} s_{q'} \xrightarrow[\mathcal{A}_i]{\gamma} p' \xrightarrow[\mathcal{A}_i]{w''} s_t,$$

and since every transition of $\mathcal{A}_i$ that leads to s_t must starts either from s_0 or from a successor of s_0 (this is easily verified by applying the construction of Definition 22), p' must also be a successor $s_{t'}$ of s_0. Moreover, from the above equation and from inductive hypothesis, we know that G contains (i) a path from (q, w') to (q', ε), (ii) an edge from (q', γ) to (p', ε), and (iii) a path from (p', w'') to (p, ε). This immediately implies that G contains a path from (q, w) to (p, ε).
This is the result we wished to prove. □

Lemma 4. *Let $\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2, \ldots$ be a sequence of automata satisfying Definition 22. The sequence $Y_0, Y_1, Y_2, \ldots$ of the sets represented by $\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2, \ldots$ satisfies Property 4.*

Proof. The proof is by induction on i. The case $i = 0$ is trivial since $Y_0 = F$. So, let assume that $(q, w) \in Y_i$, with $i > 0$. By definition of Y_i, $s_0 \xrightarrow[\mathcal{A}_i]{q} s_q \xrightarrow[\mathcal{A}_i]{w} p$, where p is a final state of $\mathcal{A}_i$. By Lemma 3, G contains a path from (q, w) to another vertex (r, v), where $r \xrightarrow[\mathcal{A}_0]{v} p$. Since $Y_0 = F$, $(r, v) \in F$ and hence $(q, w) \in \mathit{Pre}^*(F)$. □

Putting together Lemma 1, 2, and 4, we have the following main result.

Theorem 15. *Given a pushdown system $\mathcal{P}$ and an automaton $\mathcal{A}$ representing a set F of target vertices of the transition graph of $\mathcal{P}$, one can compute an automaton $\mathcal{A}_{Pre^*}$ representing the set $Pre^*(F)$.*

It should be noted that the construction of the automaton $\mathcal{A}_{Pre^*}$ requires only polynomial time in the size (i.e. number of states and transitoins) of $\mathcal{P}$ and $\mathcal{A}$. Indeed, if we denote by m the size of $\mathcal{P}$ and by n the size of $\mathcal{A}$, we have:

- $\mathcal{A}_{Pre^*}$ has the same number n of states of $\mathcal{A}$,
- at most n^2m transitions are added to $\mathcal{A}$ to yield $\mathcal{A}_{Pre^*}$,
- during the construction of each automaton $\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2, ...$, polynomial time suffices to decide if a new transition can be added to the current automaton.

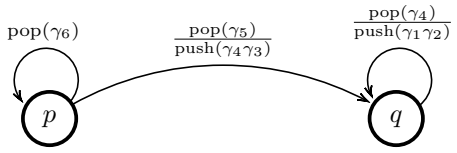

Fig. 7.14. A pushdown system.

We conclude the section with an example of application of the described construction. Let $\mathcal{P} = (Q, \Gamma, \Delta)$ be a pushdown system, where $Q = \{p, q\}$, $\Gamma = \{\gamma_1, \gamma_2, \gamma_3, \gamma_4, \gamma_5, \gamma_6\}$, and $\Delta = \{(p, \gamma_6, \varepsilon, q), (p, \gamma_5, \gamma_4\gamma_3, q), (q, \gamma_4, \gamma_1\gamma_2, q)\}$ (see Figure 7.14). Consider now the set of target configurations $F = \{(q, \gamma_1\gamma_2\gamma_3)\}$, which can be represented by an automaton $\mathcal{A}_0$. In the sequel, the reader should refer to Figure 7.15 in order to follow easily the construction. During the first step of the backward reachability analysis, we know that $\mathcal{A}_0$ can go from the initial state to the final state reading the word $q\gamma_1\gamma_2\gamma_3$, so we add to $\mathcal{A}_0$ a γ_4-labeled transition from the state s_q to its successor. In such a way we obtain an automaton $\mathcal{A}_1$ representing the set of configurations $Y_1 = \{(q, \gamma_1\gamma_2\gamma_3), (q, \gamma_4\gamma_3)\}$. During the second step, we add to $\mathcal{A}_1$ a γ_5-labeled transition from s_p to the final state, thus obtaining the automaton $\mathcal{A}_2$ representing the set of configurations $Y_2 = \{(q, \gamma_1\gamma_2\gamma_3), (q, \gamma_4\gamma_3), (p, \gamma_5)\}$. At the third step, we add to $\mathcal{A}_2$ a γ_6-labeled transition from s_p to s_p and we obtain the automaton $\mathcal{A}_3$ representing $Y_3 = \{(q, \gamma_1\gamma_2\gamma_3), (q, \gamma_4\gamma_3), (p, \gamma_6^*\gamma_5)\}$. At this point the construction stops since no further transitions can be added. Hence, we have $\mathcal{A}_{Pre^*} = \mathcal{A}_3$ and $Pre^*(F) = Y_3$.
Notice that in this example we have $X_1 = Y_1$, $X_2 = Y_2$, but $X_3 \subset Y_3$. Indeed, in the third step of the construction, after adding the γ_6-labeled transition, $\mathcal{A}_3$ accepts all the configurations of the form $(p, \gamma_6^k\gamma_5$, with $k \geq 1$. However, despite the fact that these configurations are not immediate predecessors of elements in X_2, they are all in $Pre^*(F)$ because $(p, \gamma_6^k\gamma_5) \in X_{k+2}$ for all $k \geq 1$.

The backward reachability analysis described in this section can be generalized to cope with more powerful variants of pushdown systems. As a matter of fact, in [11] Bouajjani et al. proved that a similar construction can be performed over alternating pushdown systems. Differently from a simple pushdown automaton, which can non-deterministically choose a successful computation, an alternating pushdown automaton is able to spawn different computations at the same time and either check that at least one successful computation exists (existential non-determinism) or check that all computations from that point are successful (universal non-determinism). By analogy, the class of automata that is used to represent increasing sets of reachable configurations is the alternating counterpart of the class of sequential finite-state automata. This general setting allows to reason in a uniform way about analysis problems where existential and universal path quantification must be considered. Meaningful examples of applicative areas are the model-checking problems for branching-time logics (e.g. alternation-free propositional μ-calculus) and the synthesis of winning strategies for 2-player games.

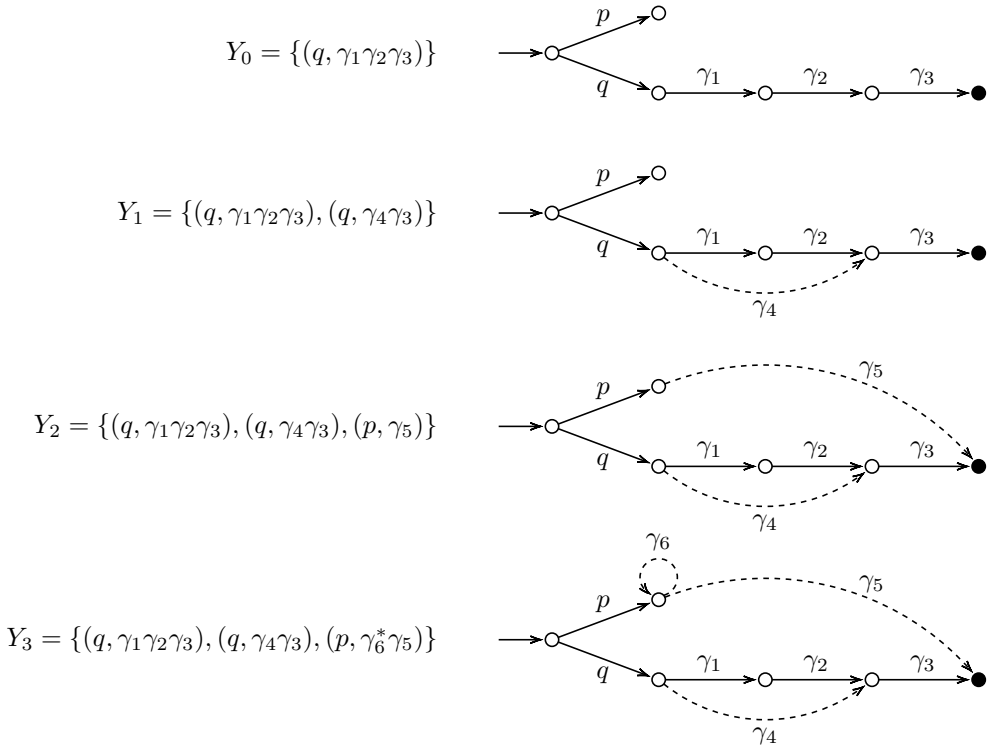

Fig. 7.15. Construction of an automaton representing the set $Pre^*(F)$.

It is also believed that the backward reachability analysis could be generalized to cope with the so-called higher-order pushdown systems, which, roughly speaking, are automata working on level n stacks rather than simple stacks (a level 1, or simple, stack is a finite word, a level $n+1$ stack is a stack of level n stacks). In this respect, partial results have already been obtained in the literature for a restricted class of higher-order pushdown systems, precisely the higher-order pushdown systems with no control states [12].

7.2 Reachability over transition graphs of Petri Nets

Petri nets, or place-transition nets, turned out to be classical models of concurrency, non-determinism, and control flow. A Petri net can be viewed as a (non-simple[6]) directed bipartite graph, where the domain is partitioned into two sets, the set of places and the set of transitions, and where arcs connect places to transitions and vice versa (hence arcs from places to places or from transitions to transitions are forbidden). The places from which an arc leads to a transition are called the input places of the transition; the places to which arcs run from a transition are called the output places of the transition. Places may contain any number of tokens. A distribution of tokens over the places of a Petri net is called a marking (this defines a configuration of the corresponding transition system). A transition is said to be enabled if there are tokens in every input place; when an enabled transition is activated (notice that transitions are activated non-deterministically) a token is consumed from each input place and a new token is put into each output place.

Here we report a formal definition of Petri net (a variety of other definitions exist in the literature, which allows, for instance, the labeling of the transitions, the specification of how many tokens are

[6] In a non-simple graph more than one edge can connect the same pair of vertices.

consumed/produced during the activation of each transition, or the association of a time stamp to each token).

Definition 23. *A* Petri net *is a* 4*-tuple* (P, T, I, O)*, where*

- P *is a finite set of* places*,*
- T *is a finite set of* transitions *(disjoint from* P*),*
- $I : P \times T \rightarrow \mathbb{N}$ *is the* input arc function *(it specifies how many arcs go from a certain place to a certain transition)*
- $O : T \times P \rightarrow \mathbb{N}$ *is the* output arc function *(it specifies how many arcs go from a certain transition to a certain place)*

Let $\mathcal{P} = (P, T, I, O)$ be a Petri net. A *marking* of $\mathcal{P}$ is a function $m : P \rightarrow \mathbb{N}$ that associates to each place a certain number of tokens. We say that a marking m *enables* a transition t if for every place p, $m(p) \geq I(p, t)$ (namely, if each place p contains at least $I(p, t)$ tokens). A computation step of $\mathcal{P}$ is then specified by the relation Δ, which defined as follows. Given two markings $m, m' : P \rightarrow \mathbb{N}$ and a transition $t \in T$, $(m, m') \in \Delta$ iff m enables t and m' satisfies $m'(p) = m(p) - I(p, t) + O(t, p)$ for every $p \in P$. A *run* of $\mathcal{P}$ is a (possibly infinite) sequence of markings $m_0, m_1, m_2...$ such that, for every $i \geq 0$, $(m_i, m_{i+1}) \in \Delta$.

Graphically the places of a Petri net $\mathcal{P}$ are represented by circles and its transitions by horizontal or vertical bars. A marking m of $\mathcal{P}$ net is represented by putting $m(p)$ spots inside each circle p. As an example, the Petri net depicted in Figure 7.16 models a simple mutual exclusion protocol between two two processes ('customer1' and 'customer2') that compete to access a shared resource ('waiter').

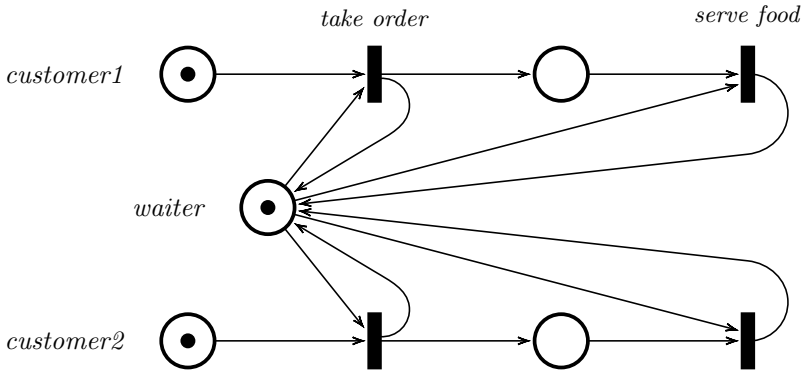

Fig. 7.16. A Petri net modeling a simple mutual exclusion protocol.

A digression towards well quasi-orderings. We introduce some basic notations and terminology about well quasi-orderings. We say that a binary relation $\preceq$ over a domain D is a *quasi-ordering* if it satisfies the reflexive property ($a \preceq a$ for all $a \in D$) and the transitive property ($a \preceq b$ and $b \preceq c$ imply $a \preceq c$). $\preceq$ is then said to be a *well quasi-ordering* if there are no infinite strictly decreasing sequences nor infinite sequences of pairwise incomparable elements. More formally, given any infinite sequence of elements $a_0, a_1, a_2, ... \in D$, it never happens that $a_i \not\preceq a_j$ for all $i < j$. Moreover, we say that a set A is *upward closed* if, for every $a \in A$ and for every $b \succeq a$, $b \in A$ holds. The following lemma implies that, if $\preceq$ is a well quasi-ordering, then there is no infinite sequence $A_0 \subset A_1 \subset A_2 \subset ...$ of strictly increasing upward closed sets.

Lemma 5. *Given a quasi-ordering* $\preceq$ *on* D*,* $\preceq$ *is a well quasi-ordering iff for every infinite sequence* $A_0 \subseteq A_1 \subseteq A_2 \subseteq ...$ *of upward closed subsets of* D*, there is* k *such that* $A_k = A_{k+1}$*.*

Proof. As for the left to right implication, suppose, by contradiction, that we have an infinite sequence $A_0 \subset A_1 \subset A_2 \subset ...$ of upward closed sets. It follows that there is a sequence $a_0, a_1, a_2, ...$ of elements in D such that for every $i \geq 0$ and for every $j < i$, $a_i \in A_i$ and $a_j \notin A_j$. This means that $a_j \not\preceq a_i$ for all $j < i$ (otherwise $a_i \in A_j$ since A_j is upward closed), which violates the well quasi-ordering assumption.
As for the converse implication, suppose that we have an infinite sequence of elements $a_0, a_1, a_2, ...$ in D where $a_j \not\preceq a_i$ for all $j < i$. We define the infinite sequence $A_0, A_1, A_2, ...$ of upward closed sets, where $A_i = \{b \in D\} \exists\ j \leq i.\ a_j \preceq b$. It is clear that $A_0 \subset A_1 \subset A_2 \subset ....$ □

Definition 24. *Given a well quasi-ordering $\preceq$ and a (possibly infinite) set A, we say that M is a* minor set *of A iff (i) $M \subseteq A$, (ii) for every $a, b \in M$, $a \not\preceq b$ (namely, M contains pairwise incomparable elements), and (iii) for every $a \in A$, there is $b \in M$ such that $b \preceq a$ (namely, M elements below each element of A).*

Lemma 6. *If $\preceq$ is a well quasi-ordering, then for each set A there exists at least one finite minor set of A.*

Proof. Suppose that no finite minor set of A exists. We show that $\preceq$ is not a well quasi-ordering. We define the infinite sequence $a_0, a_1, a_2, ...$ of elements in A as follows. Let a_0 be any arbitrary element in A. We choose a_{i+1} such that $a_j \not\preceq a_{i+1}$ for every $0 \leq j \leq i$. The element a_{i+1} exists, since otherwise we could easily construct a minor set of the finite set $\{a_0, a_1, ..., a_i\}$, which would be also a minor set of A. It is clear that the infinite sequence $a_0, a_1, a_2, ...$ violates the well quasi-ordering assumption. □

Notice that there may exist infinitely many minor sets of a given set A. However, if $\preceq$ is a *partial order*, namely it satisfies also the anti-symmetric property (i.e. $a \preceq b$ and $b \preceq a$ imply $a = b$), then there is a *unique* minor set of each set A, which we denote by $min(A)$. Moreover, notice that if A is *upward closed*, then the upward closure $\{b \in D\} \exists\ a \in min(A).\ a \preceq b$ of $min(A)$ coincides with A. Thus, we can finitely represent an upward closed set A by its minor set $min(A)$.

Petri nets are well-structured systems. We now switch back to Petri nets and disclose some relevant properties of them. Given a Petri net $\mathcal{P} = (P, T, I, O)$, we denote by $\mathcal{M}_\mathcal{P}$ the set of all markings of the form $m : P \rightarrow \mathbb{N}$ and by $(\mathcal{M}_\mathcal{P}, \Delta)$ the transition system that consists of all possible markings of $\mathcal{P}$ and all computation steps $(m, m') \in \Delta$. We then extend the usual ordering $\leq$ of the naturals to a partial ordering $\preceq$ over $\mathcal{M}_\mathcal{P}$ as follows: for every pair of ω-markings $m, m' \in \mathcal{M}_\mathcal{P}$, $m \preceq m'$ iff for every $p \in P$, $m(p) \leq m'(p)$. Notice that $\preceq$ is a *well* partial ordering and hence, by Lemma 6, we can use minor sets to finitely represent upward closed sets of markings.

Definition 25. *The* predecessor function $Pre : \mathscr{P}(\mathcal{M}_\mathcal{P}) \rightarrow \mathscr{P}(\mathcal{M}_\mathcal{P})$ *of the transition system $(\mathcal{M}_\mathcal{P}, \Delta)$ corresponding to a Petri net $\mathcal{P} = (P, T, I, O)$ is defined as follows: given a set $M \subseteq \mathcal{M}_\mathcal{P}$ of markings, m belongs to $Pre(M)$ iff there exist $m' \in M$ such that $(m, m') \in \Delta$. The reflexive and transitive closure of Pre is the function Pre^* defined by $Pre^*(M) = \bigcup_{i \geq 0} Pre^i(M)$.*

Proposition 10. *The function Pre of the transition system $(\mathcal{M}_\mathcal{P}, \Delta)$ corresponding to a Petri net $\mathcal{P}$ is monotonic w.r.t. $\preceq$, namely, it maps upward closed sets of markings to upward closed sets of markings.*

Proof. Suppose that $M' \subseteq \mathcal{M}_\mathcal{P}$ is an upward closed set of markings for $\mathcal{P}$ and let $M = Pre(M')$. Consider a generic marking $m \in M$ and a marking $n \succeq m$. By definition of M, there is a marking $m' \in M'$ such that $(m, m') \in \Delta$. In particular, there is a transition $t \in T$ that is enabled in m and such that $m'(p) = m(p) - I(p, t) + O(t, p)$ for all $p \in P$. Clearly, t is enabled in n as well, and hence $(n, n') \in \Delta$, where $n'(p) = n(p) - I(p, t) + O(t, p)$ for every $p \in P$. This implies that $n' \succeq n$ and hence, since M' is upward closed, $n' \in M'$. Again, by definition of M, we have $n \in M$, which shows that M is upward closed. □

Proposition 10 yields to a straightforward algorithm that performs backward reachability analysis on Petri nets. It simply starts from a given set F of target markings (represented by $min(F)$) and repeatedly applies the Pre function, thus obtaining a sequence $F = X_0 \subseteq X_1 \subseteq X_2 \subseteq ...$ of increasing upward closed sets. It should be noted that, at each step, the minor set $min(Pre(X_i))$ representing X_{i+1} can be effectively built from $min(X_i)$. Lemma 5 then ensures termination of the algorithm, since it implies that there always exists an index k such that $X_k = X_{k+1}$ and hence $X_k = Pre^*(F)$.

It is worth to remark that the described approach can be applied to several different systems that satisfy similar properties, namely:

- the existence of a well quasi-ordering on the data domain,
- the monotonicity of the corresponding Pre function,
- the computability of a minor set of $Pre(X_i)$ from a minor set of X_i.

Meaningful examples of such systems, which are called *well-structured* in [1, 2, 44], are lossy channels systems, timed Petri nets, vector addition systems, basic parallel processes, real time automata, relational automata, etc.

References

[1] P.A. Abdulla, K. Cerans, B. Jonsson, and Yih-Kuen Tsay. General decidability theorems for infinite-state systems. In *Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science (LICS)*, page 313. IEEE Computer Society, 1996.

[2] P.A. Abdulla, K. Cerans, B. Jonsson, and Yih-Kuen Tsay. Algorithmic analysis of programs with well quasi-ordered domains. *Information and Computation*, 160(1-2):109–127, 2000.

[3] P.A. Abdulla and B. Jonsson. Verifying programs with unreliable channels. *Information and Computation*, 127(2):91–101, 1996.

[4] K. Barthelmann. On equational simple graphs. Technical Report 9, Universität Mainz, Institut für Informatik, 1997.

[5] J.R. Büchi. Weak second order arithmetic and finite automata. *Z. Math. Logik Grundlag. Math.*, 6:66–92, 1960.

[6] J.R. Büchi. On a decision method in restricted second order arithmetic. In *Proceedings of the International Congress for Logic, Methodology and Philosophy of Science*, pages 1–11. Stanford University Press, 1962.

[7] J.R. Büchi. State-strategies for games in $f_{\sigma\delta} \cap g_{\delta\sigma}$. *Journal of Symbolic Logic*, 48:1171–1198, 1983.

[8] A. Blumensath. Automatic structures, 1999.

[9] A. Blumensath. Prefix-recognizable graphs and monadic second-order logic. Technical Report AIB-06-2001, RWTH Aachen, 2001.

[10] A. Blumensath and E. Grädel. Automatic structures. In *Logic in Computer Science*, pages 51–62, 2000.

[11] A. Bouajjani, J. Esparza, and O. Maler. Reachability analysis of pushdown automata: Application to model-checking. In *International Conference on Concurrency Theory*, pages 135–150, 1997.

[12] A. Bouajjani and A. Meyer. Symbolic reachability analysis of higher-order context-free processes. In *Proceedings of the 24th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, volume 3328 of *Lecture Notes in Computer Science*. Springer, 2004.

[13] J. Bradfield and C. Stirling. Modal logics and mu-calculi: an introduction. In J. Bergstra, A. Ponse, and S. Smolka, editors, *Handbook of Process Algebra*. Elsevier, 2001.

[14] A. Carayol and S. Wöhrle. The Caucal hierarchy of infinite graphs in terms of logic and higher-order pushdown automata. In *Proceedings of the 23rd Conference on Foundations of Software Technology and Theoretical Computer Science*, volume 2914 of *LNCS*, pages 112–123. Springer-Verlag, 2003.

[15] O. Carton and W. Thomas. The monadic theory of morphic infinite words and generalizations. In *Proceedings of the 25th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, LNCS, pages 275–284. Springer-Verlag, 2000.

[16] O. Carton and W. Thomas. The monadic theory of morphic infinite words and generalizations. *Information and Computation*, 176(1):51–65, 2002.

[17] D. Caucal. On the regular structure of prefix rewriting. In *Proceedings of the 15th International Colloquium on Trees in Algebra and Programming (CAAP)*, volume 431 of *LNCS*, pages 87–102. Springer-Verlag, 1990.

[18] D. Caucal. On the regular structure of prefix rewriting. *Theoretical Computer Science*, 106:61–86, 1992.
[19] D. Caucal. On infinite transition graphs having a decidable monadic theory. In *Proceedings of the 23nd International Colloquium on Automata, Languages and Programming (ICALP)*, volume 1099 of *LNCS*, pages 194–205. Springer-Verlag, 1996.
[20] D. Caucal. On infinite terms having a decidable monadic theory. In *Proceedings of the 27th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, volume 2420 of *LNCS*, pages 165–176. Springer-Verlag, 2002.
[21] D. Caucal. On infinite transition graphs having a decidable monadic theory. *Theoretical Computer Science*, 290:79–115, 2003.
[22] D. Caucal and T. Knapik. An internal presentation of regular graphs by prefix-recognizable graphs. *Theory of Computing Systems*, 34(4):299–336, 2001.
[23] T. Colcombet and C. Löding. On the expressiveness of deterministic transducers over infinite trees. In *Proceedings of the 21st Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 2996 of *LNCS*, pages 428–439. Springer-Verlag, 2004.
[24] B. Courcelle. The monadic second-order logic of graphs II: Infinite graphs of bounded tree width. *Mathematical Systems Theory*, 21:187–221, 1989.
[25] B. Courcelle. Graph rewriting: An algebraic and logic approach. In J. Van Leeuwen, editor, *Handbook of Theoretical Computer Science*, pages 193–242. Elsevier, 1990.
[26] B. Courcelle. The monadic second-order logic of graphs VII: Graphs as relational structures. *Theoretical Computer Science*, 101(1):3–33, 1992.
[27] B. Courcelle. Monadic second-order graph transductions: a survey. *Theoretical Computer Science*, 126:53–75, 1994.
[28] B. Courcelle. The monadic second-order theory of graphs IX: Machines and their behaviors. *Theoretical Computer Science*, 151:125–162, 1995.
[29] B. Courcelle. The expression of graph properties and graph transformations in monadic second-order logic. In G. Rozenberg, editor, *Handbook of graph grammars and computing by graph transformations*, volume 1, pages 313–400. World Scientific, 1997.
[30] B. Courcelle and T. Knapik. The evaluation of first-order substitution is monadic second-order compatible. *Theoretical Computer Science*, 281(1-2):177–206, 2002.
[31] B. Courcelle and I. Walukiewicz. Monadic second-order logic, graph coverings and unfoldings of transition systems. *Annals of Pure Applied Logic*, 92(1):35–62, 1998.
[32] W. Damm. An algebraic extension of the Chomsky-hierarchy. In *Proceedings of the 8th Symposium on Mathematical Foundations of Computer Science (MFCS)*, volume 74 of *LNCS*, pages 266–276. Springer-Verlag, 1979.
[33] W. Damm. The IO- and OI-hierarchies. *Theoretical Computer Science*, 20:95–207, 1982.
[34] W. Damm and A. Goerdt. An automata-theoretic characterization of the OI-hierarchy. In *Proceedings of the 9th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 140 of *LNCS*, pages 141–153. Springer-Verlag, 1982.
[35] W. Damm and A. Goerdt. An automata-theoretical characterization of the OI-hierarchy. *Information and Control*, 71(1):1–32, 1986.
[36] C.C. Elgot and M.O. Rabin. Decidability and undecidability of extensions of second (first) order theory of (generalized) successor. *Journal of Symbolic Logic*, 31(2):169–181, 1966.
[37] E.A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 995–1072. Elsevier, 1990.
[38] J. Engelfriet. Bottom-up and top-down tree transformations – a comparison. *Mathematical Systems Theory*, 9(3):198–231, 1975.
[39] J. Engelfriet. A characterization of context-free NCE graph languages by monadic second-order logic on trees. In *Proceedings of the 4th International Workshop on Graph-grammars and Their Application to Computer Science*, pages 311–327. Springer, 1991.
[40] J. Engelfriet. Iterated stack automata and complexity classes. *Information and Computation*, 95(1):21–75, 1991.
[41] J. Engelfriet. Context-free graph grammars. In G. Rozenberg and A. Salomaa, editors, *Handbook of formal languages*, volume 3, pages 125–213. Springer, 1997.
[42] J. Engelfriet and L.M. Heyker. Hypergraph languages of bounded degree. *Journal of Computer and System Sciences*, 48(1):58–89, 1994.

[43] J. Engelfriet, L.M. Heyker, and G. Leih. Context-free graphs languages of bounded degree are generated by apex graph grammars. *Acta Informatica*, 31(4):341–368, 1994.
[44] A. Finkel and P. Schnoebelen. Well-structured transition systems everywhere! *Theoretical Computer Science*, 256(1-2):63–92, 2001.
[45] F. Gécseg and M. Steinby. Tree automata. *Akadémiai Kiodó*, 1984.
[46] S. Ginsburg and E. Spanier. Semigroups, Presburger formulas and languages. *Pacific Journal of Mathematics*, 16(2):285–296, 1966.
[47] B.R. Hodgson. On direct products of automaton decidable theories. *Theoretical Computer Science*, 19:331–335, 1982.
[48] J.E. Hopcroft, R. Motwani, and J.D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, 2001.
[49] D. Janin and I. Walukiewicz. On the expressive completeness of the propositional mu-calculus with respect to monadic second order logic. In U. Montanari and V. Sassone, editors, *Proceedings of the 7th International Conference on Concurrency Theory (CONCUR)*, volume 1119, pages 263–277. Springer, 1996.
[50] R.M. Karp and E. Miller. Parallel program schemata. *Computer and System Sciences*, 3(2):147–195, 1969.
[51] B. Khoussainov and A. Nerode. Automatic presentations of structures. In *Selected Papers from the International Workshop on Logical and Computational Complexity (LCC)*, volume 960 of *Lecture Notes in Computer Science*, pages 367–392. Springer, 1994.
[52] T. Knapik, D. Niwiński, and P. Urzyczyn. Deciding monadic theories of hyperalgebraic trees. *Lecture Notes in Computer Science*, 2044:253–267, 2001.
[53] T. Knapik, D. Niwinski, and P. Urzyczyn. Higher-order pushdown trees are easy. In *Foundations of Software Science and Computation Structure*, pages 205–222, 2002.
[54] D. Kozen. Results on the propositional mu-calculus. *Theoretical Computer Science*, 27:333–354, 1983.
[55] C. Löding. Reachability problems on regular ground tree rewriting graphs. *Theory of Computing Systems*, 39(2):347–383, 2006.
[56] A. Montanari and G. Puppis. Decidability of MSO theories of tree structures. In *Proceedings of the 24th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, volume 3328 of *LNCS*, pages 430–442. Springer-Verlag, 2004.
[57] A. Montanari and G. Puppis. Decidability of the theory of the totally unbounded ω-layered structure. In *Proceedings of the 11th International Symposium on Temporal Representation and Reasoning (TIME)*, pages 156–160, 2004.
[58] A. Montanari and G. Puppis. Deciding MSO theories of tree structures by tree contractions. Research Report UDMI/2005/10, Dipartimento di Matematica e Informatica, Università di Udine, Italy, 2005.
[59] C. Morvan. On rational graphs. In *Proceedings of 3rd International Conference on Foundations of Software Science and Computation Structures (FOSSACS)*, pages 252–266, 2000.
[60] C. Morvan. *Les graphes rationnels*. PhD thesis, IFSIC Université de Rennes 1, 2001.
[61] A.W. Mostowski. Regular expressions for infinite trees and a standard form of automata. In *Proceedings of the 5th Symposium on Computation Theory*, volume 208 of *LNCS*, pages 157–168. Springer-Verlag, 1984.
[62] D. Muller and P. Schupp. The theory of ends, pushdown automata, and second-order logics. *Theoretical Computer Science*, 37:51–75, 1985.
[63] G. Puppis. *Automata for branching and layered temporal structure*. PhD thesis, Dipartimento di Matematica e Informatica, Università di Udine, Italy, 2006.
[64] M.O. Rabin. Decidability of second-order theories and automata on infinite trees. *Transactions of the American Mathematical Society*, 141:1–35, 1969.
[65] A. Rabinovich. On decidability of monadic logic of order over the naturals expanded by unary predicates. In *Logic Colloquium*, 2005.
[66] D. Seese. The structure of models of decidable monadic theories of graphs. *Annals of Pure and Applied Logic*, 53(2):169–195, 1991.
[67] A.L. Semenov. Decidability of monadic theories. In *Proceedings of the 11th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, volume 176 of *LNCS*, pages 162–175. Springer-Verlag, 1984.
[68] C. Stirling. Decidability of bisimulation equivalence for pushdown processes. Research Report EDI-INF-RR-0005, Division of Informatics, University of Edinburgh, 2000.

[69] W. Thomas. Infinite trees and automaton definable relations over ω-words. In *Proceedings of the 7th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 415 of *LNCS*, pages 263–277. Springer-Verlag, 1990.

[70] W. Thomas. Languages, automata, and logic. In G. Rozemberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 3, pages 389–455. Springer-Verlag, 1997.

[71] W. Thomas. A short introduction to infinite automata. In *Revised Papers from the 5th International Conference on Developments in Language Theory*, pages 130–144. Springer, 2002.

[72] W. Thomas. Constructing infinite graphs with a decidable MSO-theory. In *Proceedings of the 28th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, LNCS, pages 113–124. Springer-Verlag, 2003.

[73] I. Walukiewicz. Monadic second-order logic on tree-like structures. In *Proceedings of the 13th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 1046 of *LNCS*, pages 401–413. Springer-Verlag, 1996.

[74] I. Walukiewicz. Monadic second-order logic on tree-like structures. *Theoretical Computer Science*, 275:311–346, 2002.