A Note about Compositional Verification of Sequential Programs

Olivier Ly

LaBRI – Bordeaux I University

April 2, 2004

Abstract. This paper deals with the *compositional verification* of sequential programs. This consists in deciding whether or not a given set of local structural properties of the functions of a program implies a given global behavioural property of the program. Here we consider properties expressed in monadic second-order logic dealing with the control flow of the program and the function calls occuring during its execution. This problem has been investigated in relation with the security of open multi-application smart cards. We prove that the compositionality is a *decidable* problem for sequential programs whose control-flow graphs are of tree-width less than a fixed integer value, which includes in particular structured programs.

Keywords. Compositional Verification, Tree Automata, Monadic Second-Order Logic

Introduction

This paper deals with the *compositional verification* of sequential programs. This consists in deciding whether or not a given set of local structural properties of the functions of a program implies a given global behavioural property of the program. This aims at reducing the verification of a global property of a program to the verifications of some independent local properties of its components.

Compositional verification applies to modular systems whose components are intended to be updated frequently: once we have a compositionality result dividing the verification of a given global property of the system into several local properties of its components, when a component is updated, it is sufficient to verify only the component in question to ensure the validity of the whole system. This applies in particular to embedded systems which must ensure their own validity themselves with a limited amount of computing resources[1].

Compositional verification is a long standing problem in the area of concurrent systems (see *e.g.* [12, 1, 16]). Here we rather consider sequential programs and we restrict our study to behavioural and structural properties dealing with the control flow of the program and the function calls occurring during execution. This framework relies on the language-independent model of programs introduced in [15] in order to catch some classical security properties. It has been studied in [4, 3, 22] (see also [10]) for compositional reasoning about security of open multi-application smart cards. Like *e.g.* [23, 2, 18] for concurrent systems, a proof system dealing with modal μ-calculus has been set up in [4] for the framework that we consider. But the question of decidability was left open. Then a decision procedure has been proposed in [22] by restricting the study to properties expressed in simulation logic. Here we consider properties expressed in *monadic second-order logic* (see *e.g.* [13]) and we prove that for any fixed integer value k, the compositionality is a *decidable* problem for sequential programs whose control-flow graphs are of tree-width less than k.

This contributes to the studies mentioned above seeing that monadic second-order logic contains modal μ-calculus and *a fortiori* simulation logic. The limitation of our solution concerns the tree-width of the control-flow graphs. Roughly speaking, the tree-width is an integer value measuring how far a graph is from being a tree (see [20]). We claim that this limitation is reasonable, because reasonable programs are indeed of bounded tree-width as it has been showed in [25]: the tree-width of the control-flow graph of a goto-free C program is at most 6. Moreover, a study reported in [14] showed that the tree-width of the control-flow graphs of the functions of the Java API source code does not exceed 5, and the average of it is only 2.7.

All the paper is devoted to the proof of the decidability result (Theorem 1), which relies on a variation of the classical link between monadic second-order logic and *automata* (see *e.g.* [24]). First, we translate the compositionality problem into a recognisable tree language problem, introducing a new kind of automata on infinite trees called *composite automata*. Via the concept of syntactic expressions, these automata encode the transition systems which encode the behaviours of the programs made of functions satisfying a given set of structural properties. Then, we set up an encoding of these transition systems by *regular infinite graphs*, i.e., infinite graphs generated by graph grammars. This allows us to translate behavioural properties into *Rabin automata*. Finally we obtain a pumping lemma concerning trees recognised simultaneously by a composite automaton and a Rabin automaton. This lemma is proved by using a variation of the classical point view of runs of automata in terms of *games* (see *e.g.* [24]). This result allows us to give out a bound for the length of solutions, and therefore to solve the problem by an enumeration method.

Preliminaries

A *hypergraph*, also called *relational structure*, is a tuple of the form $(V, L, \{R_\ell\}_{\ell \in L})$ where V is the set of vertices; L is the finite set of labels. And for each $\ell \in L$, R_ℓ is a relation over V, i.e., a subset of V^n for some strictly positive integer n. An element of R_ℓ of form $(v_1, ..., v_n)$ defines a *hyperedge* whose vertex list is $v_1, ..., v_n$; n is called the arity of the hyperedge. Let us note that the labelling of vertices is encoded by hyperedges of arity 1. *Graphs* are the hypergraphs whose hyperedges are all of arity 1 or 2. Here we actually deal with hypergraphs with *sources* which generalise pointed graphs; such a hypergraph is a hypergraph provided with a finite ordered list of pairwise distinct vertices. These distinguished vertices are called the sources of the hypergraph, the other ones are said to be internal. The sequence of sources of a hypergraph H is denoted by $src(H)$. The number of sources of a hypergraph is called its *type*.

Let Σ be a finite set. A Σ-labelled *n-ary ordered tree* (a tree for short) is a mapping $t : \mathcal{D}om(t) \subset \{0, \ldots, n-1\}^* \longrightarrow \Sigma$ such that $\mathcal{D}om(t)$ is prefix-closed. The elements of $\mathcal{D}om(t)$ are called the *nodes* of t. Let $x \in \mathcal{D}om(t)$ be a node of t, the elements of $\mathcal{D}om(t)$ of the form $x.s$ with s integer are called the *sons* of x; a node without any son is called a *leaf* of t, the other ones are called the *internal nodes*; $t(x)$ is called the *label* of x. In order to simplify notations, $\mathcal{D}om(t)$ is also denoted by t when it is clear according to the context. A *subtree* of a tree t is a mapping $t' : \mathcal{D}om(t') \subset \mathcal{D}om(t) \to \Sigma$ such that there exists a node $r \in \mathcal{D}om(t)$ and a prefix-closed set $W \subset \{0, \ldots, n-1\}^*$ such that $\mathcal{D}om(t') = r.W$, and for any $x \in \mathcal{D}om(t')$: $t'(x) = t(x)$.

Let G be a hypergraph. A *tree-decomposition* is pair (U, f) where U is an undirected tree and $f : V_U \to 2^{V_G}$ such that: $V_G = \bigcup_{i \in V_U} f(i)$; if a list of vertices $v_1 \dots v_n$ of G are connected by a hyperedge, then there exists $i \in V_U$ such that $v_1, \dots, v_n \in f(i)$; if $i, j, k \in V_U$ are such that j belongs to the shortest path between i and k, then $f(i) \cap f(k) \subseteq f(j)$. The *tree-width* of such a decomposition is defined to be $\max\{\text{card}(f(i)) \mid i \in V_U\} - 1$. The *tree-width* of G, denoted by $twd(G)$, is the minimum of the tree-widths of all its tree-decompositions.

1 Program Models

1.1 Definition

Here we define our model of programs. It generalises[2] the concepts introduced in [4] which was inspired by [15]. In particular, our model allows us to catch security properties considered in [4], see also [15, 3, 22].

Let $\mathcal{M}$ be a set of symbol whose elements are the *names* of functions. A *function* is a pair (m, G_m) where $m \in \mathcal{M}$ is the name of the function and G_m is its *control-flow graph*, i.e., a tuple $(V_m, \to_m, \lambda_m)$ such that V_m is the finite set of *program points* of m. The relation $\to_m \subset V_m \times V_m$ encode the *transfer edges* of m. $\lambda_m : V_m \to \{\mathsf{entry}, \mathsf{seq}, \mathsf{ret}\} \cup \{\mathsf{call}\ m\}_{m \in \mathcal{M}}$ associates to each program point of m a program point type. The entry points are the entry points of the function. The call points are the points where the function calls an other function. The ret points are the returning points of the function. The seq points are all the other ones, which are not distinguished. A *program* $\mathcal{P}$ consists of a set of function $\{(m, G_m)\}_{m \in \mathcal{M}}$. Let $V_\mathcal{P}$ denotes the set of all the program points of all the functions of $\mathcal{P}$. A *state* of $\mathcal{P}$ is a pair (c, σ) consisting of a program point, i.e., an element of $V_\mathcal{P}$, and an call stack $\sigma \in V_\mathcal{P}^*$.

A program $\mathcal{P}$ induces a *labelled transition system*, i.e., a graph $\mathcal{T}_\mathcal{P} = (\mathcal{S}_\mathcal{P}, \mathcal{L}_\mathcal{P}, \to_\mathcal{P})$ defined as follows: The set of *states* $\mathcal{S}_\mathcal{P}$ is the set of the states of $\mathcal{P}$ as defined above. The set of *edge labels* $\mathcal{L}_\mathcal{P}$ is defined to be $\{\tau, \mathsf{call}, \mathsf{ret}\}$. The call-labelled edges encode function calls. The ret-labelled edges encode function returns. Any other transition of the system is labelled by τ which is called the silent action. Formally, the *edge relation* $\to_\mathcal{P}$ is defined by the following rules:

$$\frac{c : \{\mathsf{entry}, \mathsf{seq}\},\ c, c' \in V_m,\ c \to_m c'}{(c, \sigma) \xrightarrow{\tau}_\mathcal{P} (c', \sigma)} \text{ (local transfer)}$$

$$\frac{\begin{array}{l} c_1 : \mathsf{call}\, m_2,\ c_1, c_1' \in V_{m_1},\ c_1 \to_{m_1} c_1' \\ c_2 : \mathsf{entry},\quad c_2 \in V_{m_2} \end{array}}{(c_1, \sigma) \xrightarrow{\mathsf{call}}_\mathcal{P} (c_2, c_1' \cdot \sigma)} \text{ (call)}$$

$$\frac{\begin{array}{c} c_1 \in V_{m_1} \\ c_2 : \mathsf{ret},\ c_2 \in V_{m_2} \end{array}}{(c_2, c_1 \cdot \sigma) \xrightarrow{\mathsf{ret}}_\mathcal{P} (c_1, \sigma)} \text{ (return)}$$

For each function name $m \in \mathcal{M}$, we consider the labelled transition system $\mathcal{T}_{\mathcal{P},m}$ defined to be the subgraph of all the vertices of $\mathcal{T}_\mathcal{P}$ which are accessible from a state of the form (c_i, ε) where c_i is an entry point of m.

1.2 Regular Graphs

The concept of *regular hypergraph*[3] extends those of context-free graphs (see [17]). It has been introduced in [8]. Such a hypergraph, possibly infinite, is defined according to a deterministic *hyperedge replacement graph grammar* (see [11]). Such a grammar is defined according to a finite set of symbols L which is divided into two subsets L_1 and L_2 whose elements are respectively called *terminal* symbols and *non-terminal* symbols. A deterministic hyperedge replacement grammar[4] is a tuple $(\ell_0, \{H_\ell\}_{\ell \in L_2})$ where $\ell_0 \in L_2$ is the initial symbol and for each $\ell \in L_2$, H_ℓ is a finite L-labelled hypergraph with source, such that for each $\ell \in L$, the arities of all the ℓ-labelled hyperedges of any $H_{\ell'}$ are equal; and if $\ell \in L_2$, this arity must be equal to the type of H_ℓ.

The construction of the regular hypergraph generated by such a grammar is as follows: First, one starts from H_{ℓ_0}. Then, according to the grammar, one replaces each hyperedge labelled by a non-terminal symbol by the associated hypergraph, gluing the sources of the hypergraph in place of the vertices of the hyperedge. One gets a new hypergraph, with possibly new hyperedges labelled by a non-terminal symbols. One replaces again these last ones as above, and so on, possibly infinitely many times, until there is no more such hyperedge.

Formally, let $(\ell_0, \{H_\ell\}_{\ell \in L_2})$ be a deterministic graph grammar. First, for each $\ell \in L_2$, let us denote by $e_{\ell 1}, ..., e_{\ell n_\ell}$ the non-terminal hyperedges of H_ℓ, i.e. the hyperedges labelled by non-terminal symbols. Second, let us consider the ordered regular tree t, whose nodes are labelled over L_2, constructed inductively as follows: the root of t is labelled by ℓ_0; and for each $\ell \in L_2$, each ℓ-labelled node of t has exactly n_ℓ sons and for $i = 1...n_\ell$, its i-th son has the same label than $e_{\ell i}$. Finally the regular hypergraph Γ associated to the grammar $(\ell_0, \{H_\ell\}_{\ell \in L_2})$ is defined as follows: for each $\ell \in L_2$ and each ℓ-labelled node μ of t, let us consider the hypergraph H_μ obtained from a copy of H_ℓ by deleting all its non-terminal hyperedges. Let $\overline{\Gamma} = \bigcup_\mu H_\mu$ be the disjoint union of the H_μ's. Let us consider the binary relation $\mathcal{R}$ over vertices of $\overline{\Gamma}$ defined as follows: for each pair (x, x') of vertices of $\overline{\Gamma}$, $(x, x') \in \mathcal{R}$ if and only if there exists a pair of nodes μ and μ' of t such that μ' is the j-th son of μ for some j, $x \in H_\mu$, $x' \in H_{\mu'}$ and there exists i such that x corresponds to the i-th vertex of $e_{\ell j}$ in H_ℓ, where ℓ denotes the label of μ, and x' is the i-th source of $H_{\mu'}$. Then the regular hypergraph Γ associated to $(\ell_0, \{H_\ell\}_{\ell \in L_2})$ is defined as a quotient of $\bigcup_\mu H_\mu$ by the transitive symmetric closure of $\mathcal{R}$. For each vertex x of $\overline{\Gamma}$, the vertex of Γ represented by x shall be denoted by $\nu(x)$. The sources of Γ are defined to be those of H_{μ_0} where μ_0 denotes the root of t.

Here we prove that the labelled transition system associated to a program is a regular graph[5] . For any $m, m' \in \mathcal{M}$, and for any $c \in V_m$, let us consider the set $D_{m',c} = \{c_1 \in V_m \mid c_1 \rightarrow_m c \text{ and } c_1 : \mathsf{call}\, m'\}$ and let $d = \max_{m',c} |D_{m',c}|$. Here we keep the notations of Section 1.1. For each $m \in \mathcal{M}$, let us consider the graph H_m obtained from the control-flow graph G_m of m as follows: Vertices of H_m are those of G_m. They are not labelled. Edges are labelled over the set $\{\tau\} \cup \mathcal{M} \times [1...d]$. Each edge of G_m whose origin is a vertex which is not labelled by call is kept in H_m. The other edges are not considered. For each $c \in V_m$ which is a successor of a call point $c' : \mathsf{call}\, m'$, we add a non-terminal hyperedge $e_{m',c}$ of arity $a = |D_{m',c}| + 1$ whose first $a - 1$ vertices are the elements of $D_{m',c}$ in some arbitrary order, and whose last vertex is c. This new edge $e_{m',c}$ is labelled by the tuple $(m', a - 1)$. No source is defined. Then, for each tuple $(m', n) \in \mathcal{M} \times [1...d]$, let us consider the graph $H_{m',n}$ constructed from $H_{m'}$ as follows: Vertices of $H_{m',n}$ are not labelled. Its edges are labelled over the

set $\{\tau\} \cup \{\mathsf{call}, \mathsf{ret}\} \cup \mathcal{M} \times [\,1...d\,]$. Starting from $H_{m'}$, one adds n new vertices $s_1...s_n$, and for each vertex of $H_{m'}$ corresponding to an entry point $c : \mathsf{entry}$ of $G_{m'}$ and for each $i \in [\,1...n\,]$, one adds an edge from s_i to c labelled by call. Then one adds an other new vertex denoted by r, and for each vertex of $H_{m'}$ corresponding to a return point c_r of $G_{m'}$, one adds a edge from c_r to r labelled by ret. The sequence of sources of $H_{m',n}$ is defined to be $(s_1, ..., s_n, r)$. $H_{m',n}$ is of type $n + 1$.

Lemma 1. *$\mathcal{T}_{\mathcal{P},m}$ is isomorphic to the regular graph defined by the deterministic graph grammar $(m, \{H_\ell\}_{\ell \in X})$ where the set of non-terminal symbols X is defined to be $\mathcal{M} \cup \mathcal{M} \times [\,1...d\,]$.*

Proof. Let Γ be the regular graph defined by $(m, \{H_\ell\}_{\ell \in X})$. On the one hand, keeping the notations of Section 1.2, let us consider the ordered regular tree t and the graph $\overline{\Gamma} = \bigcup_{\mu \in t} H_\mu$. On the other hand, let us consider the set t' of all possible values of the stack for the states of $\mathcal{T}_{\mathcal{P},m}$. The set t' has a natural tree structure.

Then t and t' are isomorphic. Indeed, keeping the notations of Section 1.2, for any $\ell \in X$, each ℓ-labelled node of t has n_ℓ sons, each one being associated to a non-terminal hyperedge of H_ℓ. Therefore, each node n of t defines a finite sequence s of such hyperedges. By construction, the last vertex of such a hyperedge is actually a return point. Therefore, by associating to such a hyperedge this return point. Each sequence of such hyperedges defines a sequence of return point in a natural way, i.e. a stack value. Altogether, this define a mapping σ associating each node μ of t to a stack value $\sigma(\mu)$, i.e. the node of t'. This mapping is one-to-one.

We now define a mapping $\overline{\beta}_\mu : H_\mu \to \mathcal{T}_{\mathcal{P},m}$, for any node μ of t as follows: Let μ be the root of t. Let us note that $\sigma(\mu) = \varepsilon$. For each vertex c of H_μ, $\overline{\beta}_\mu(c)$ is defined to be the state (c, ε). Let μ be a node of t distinct from the root. For each vertex c of H_μ, except the sources, $\overline{\beta}_\mu(c)$ is associated to the state $(c, \sigma(\mu))$. Let $(s_1, ..., s_n, r)$ be the sequence of sources of H_μ. Let μ' be the father of μ, and e the hyperedge of $H_{\mu'}$ to which is associated H_μ. For each $i \in [\,1...n\,]$, let c_i be the i-th vertex of e in $H_{\mu'}$. Then $\overline{\beta}_\mu(s_i)$ is defined to be the state $(c_i, \sigma(\mu'))$. Let c_r be the last vertex of e, then $\overline{\beta}_\mu(r)$ is defined to be the state $(c_r, \sigma(\mu'))$. Let us now consider $\overline{\beta} = \bigcup_\mu \overline{\beta}_\mu$. Then $\overline{\beta} : \overline{\Gamma} \to \mathcal{T}_{\mathcal{P},m}$ is a graph morphism. And its quotient $\beta : \Gamma \to \mathcal{T}_{\mathcal{P},m}$ by the relation $\mathcal{R}$ defined in Section 1.2 is an isomorphism. □

2 Decidability of Compositionality Problem

2.1 Compositionality Problem

We consider *monadic second-order formulæ* on hypergraphs (MSO-formulæ for short, see *e.g.* [13]). These formulæ are constructed using individual variables and set variables. Atomic formulæ are of the forms $x \in A$, $A \subset B$, or $\mathsf{edg}_\eta(x_1, \ldots, x_n)$ which encodes the fact that $x_1, \ldots, x_n$ are connected by an η-labelled hyperedge. Syntax is not restricted: we allows existential and universal quantifiers over individual and set variables, conjunctions and negations. The set of finite models (respectively infinite) of a formula φ is denoted by $\mathcal{M}(\varphi)$ (respectively $\mathcal{M}^\infty(\varphi)$).

Remark 1. This generalises the framework of [4] which considered temporal logic and modal μ-calculus whose expression powers are less than the one of monadic second-order logic.

In the following, we firstly state the compositionality problem for program models. It is undecidable. Then we give a more general statement in terms of graph grammars. Finally, we state the sub-problem that we prove to be decidable, i.e., the *bounded tree-width compositionality problem*.

Program Models. Let $\mathcal{M}$ be a finite set of names. Let us consider a finite set $(\varphi_m)_{m\in\mathcal{M}}$ of MSO-formulæ which specifies a set of programs $\mathfrak{P}((\varphi_m)_{m\in\mathcal{M}}) = \{(m, G_m)_{m\in\mathcal{M}} \mid \forall\, m \in \mathcal{M} : G_m \models \varphi_m\}$. As defined above, to each pair $(\mathcal{P}, m)$ made of a program and a function name is associated a labelled transition system $\mathcal{T}_{\mathcal{P},m}$.

Problem 1 (Program Models Compositionality).
Being given with a set of formulæ $(\varphi_m)_{m\in\mathcal{M}}$, a method name m_0, and a formula Ψ, does exist[6] a program $\mathcal{P} \in \mathfrak{P}((\varphi_m)_{m\in\mathcal{M}})$ such that $\mathcal{T}_{\mathcal{P},m_0} \models \Psi$?

Proposition 1. *Problem 1 is* undecidable.

Proof. The method is classical: Let $\mathcal{M} = \{m\}$ and let the unique φ_m be a MSO-formula whose models are the grids. Then one consider a MSO-formula Ψ specifying a finite computation of a given Turing machine on a grid specified by φ_m. This reduces Problem 1 to the Turing machine halting problem. □

Graph Grammar Specifications. Lemma 1 established a bridge between program models and graph grammars. Here we extend this link by translating the concept of program specification to graph grammar framework. Let us now consider a finite set of symbol $\mathcal{M}$ together with an *arity* mapping $\alpha : \mathcal{M} \to \mathbb{N}$. A *graph grammar specification* is a pair $\mathfrak{S} = (m_0, \{\varphi_m\}_{m\in\mathcal{M}})$ where $m_0 \in \mathcal{M}$ and for any $m \in \mathcal{M}$, φ_m is a MSO-formula such that any $g \in \mathcal{M}(\varphi_m)$ is of type $\alpha(m)$; and for any $m, m' \in \mathcal{M}$, any m'-labelled hyperedge of any $g \in \mathcal{M}(\varphi_m)$ is of arity $\alpha(m')$. Such a specification defines a family of deterministic graph grammars defined by $\mathcal{G}r(\mathfrak{S}) = \{(m_0, \{g_m\}_{m\in\mathcal{M}}) \mid \forall m \in \mathcal{M} : g_m \models \varphi_m\}$. Seeing that each deterministic graph grammar generates a regular possibly-infinite hypergraph, $\mathfrak{S}$ defines in turn a family of regular hypergraphs:
$\mathcal{R}g(\mathfrak{S}) = \{G \mid \text{there exists } (m_0, \{g_m\}_{m\in\mathcal{M}}) \in \mathcal{G}r(\mathfrak{S}) \text{ which generates } G\}$

Problem 2 (Graph Grammars Compositionality).
Being given with a graph grammar specification $\mathfrak{S} = (m_0, \{\varphi_m\}_{m\in\mathcal{M}})$ and a formula Ψ. Does exist $G \in \mathcal{R}g(\mathfrak{S})$ such that $G \models \Psi$?

On the base of Lemma 1, one gets the following result:

Lemma 2. *Problem 1 can be reduced to Problem 2.*

Proof. By Lemma 1, for a given program $\mathcal{P} = (m, G_m)_{m\in\mathcal{M}}$ and a given name m, the associated labelled transition system $\mathcal{T}_{\mathcal{P},m}$ is defined by the graph grammar $(m, \{H_\ell\}_{\ell\in X})$ as defined in Section 1.2. Here we keep the notation of Section 1.2.

Being given with a set of formulæ $(\varphi_m)_{m\in\mathcal{M}}$, one can construct a partial graph grammar specification $\mathfrak{S} = (\psi_\ell)_{\ell\in X}$ without axiom such that each graph grammar $(H_\ell)_{\ell\in X} \in \mathcal{G}r(\mathfrak{S})$ is the graph grammar constructed from some program $\mathcal{P} \in \mathfrak{P}((\varphi_m)_{m\in\mathcal{M}})$ as described in Section 1.2; and conversely the graph grammar constructed from any program $\mathcal{P} \in \mathfrak{P}((\varphi_m)_{m\in\mathcal{M}})$ belongs to $\mathcal{G}r(\mathfrak{S})$.

Therefore to decide whether there exists or not a program $\mathcal{P} \in \mathfrak{P}((\varphi_m)_{m\in\mathcal{M}})$ such that $\mathcal{T}_{\mathcal{P},m} \models \Psi$ is equivalent to decide whether there exists $G \in \mathcal{R}g(\mathfrak{S})$ such that $G \models \Psi$. Up to the construction of $\mathfrak{S} = (\psi_\ell)_{\ell\in X}$, this ends the proof.

The construction of $\mathfrak{S} = (\psi_\ell)_{\ell\in X}$ is based on the fact that for each m (respectively for each pair (m, n)) there exists a definable transduction (see [9]) which defines G_m in H_m (respectively in $H_{m,n}$). For each m (respectively each pair (m, n)) the formula ψ_m (respectively $\psi_{m,n}$) expresses firstly that its model has been constructed according to the function described in Section 1.2 which is generic, and secondly, by using the definable transduction, that it has been constructed from a graph (a G_m) satisfying φ_m. We shall not give the details of this construction. □

Now, let us fix an integer k. Let $\mathcal{R}g_k(\mathfrak{S}) = \mathcal{R}g(\mathfrak{S}) \cap \{G \mid twd(G) < k\}$ (see [20] or Section 1 for the definition of $twd(G)$). We now state the central problem of this paper which we prove to be decidable (Theorem 1):

Problem 3 (Bounded Tree-Width Compositionality).
Being given with a graph grammar specification $\mathfrak{S} = (m_0, \{\varphi_m\}_{m\in\mathcal{M}})$, an integer k, and a formula Ψ. Does exist $G \in \mathcal{R}g_k(\mathfrak{S})$ such that $G \models \Psi$?

Remark 2. Control flow graphs of structured programs have been showed to be of bounded tree-width (see [25]). For instance the tree-width of the control flow graph of a goto-free C program is at most 6. Moreover, a concrete study reported in [14] showed that the tree-width of the control flow graphs of the programs of the Java API does not exceed 5, and the average of it is only 2.7. Therefore, we claim that Problem 3 is a reasonable sub-case of Problem 2.

2.2 Specifications & Automata

Basics. First, we introduce the main tools we use for dealing with decidability issues concerning monadic second-order logic: HR-Syntactic expressions and tree automata.

HR-Syntactic expressions are the expressions based on the following operations on hypergraphs (see [8]):

- *Disjoint sum.* The disjoint sum of a hypergraph H_1 of type n and a hypergraph H_2 of type m, denoted by $H_1 \oplus_{n,m} H_2$, is defined to be the hypergraph resulting from the disjoint union of H_1 and H_2. Sources of $H_1 \oplus_{n,m} H_2$ are defined to be the concatenation of the sources of H_1 with the ones of H_2: $src(H_1 \oplus H_2) = src(H_1).src(H_2)$.
- *Redefinition of sources.* Let $\alpha : [\,1...p\,] \to [\,1...m\,]$. And let H be a hypergraph of type m. Then we consider $\sigma_{p,m,\alpha}(H)$, the hypergraph obtained from H by redefinition of sources according to α. It is defined to be the same hypergraph than H, except for the sources: let $src(H) = s_1 s_2 ... s_m$, then $src(\sigma_{p,m,\alpha}(H)) = s_{\alpha(1)} s_{\alpha(2)} ... s_{\alpha(p)}$.
- *Source fusion.* Let δ be an equivalence relation on $[\,1...m\,]$, and let H be a hypergraph of type m. Then $\delta_{m,\delta}(H)$ is obtained from H by gluing the sources of H according to δ, i.e. for any $i, j \in [\,1...m\,]$, s_i and s_j are glued together if and only if $i \equiv_\delta j$, where s_i and s_j respectively denote the i-th and the j-th source of H. The sources of $\delta_{m,\delta}(H)$ are defined to be the vertices resulting from the gluing of sources of H. They are ordered in a natural way by defining $\overline{s_i} < \overline{s_j}$ if and only if $\min[i]_\delta < \min[j]_\delta$.

– *Discrete graphs.* D_m denotes a graph with m sources, no internal vertex and no hyperedge.
– *Hyperedges.* $E_{\ell,m}$ denotes a graph with m sources and no internal vertex, connected by a ℓ-labelled hyperedge in the same ordering.

These operators are typed by the types of hypergraphs considered.

We consider the set $\mathfrak{T}_{L,k}$ (respectively $\mathfrak{T}^{\infty}_{L,k}$) made of well-formed *finite terms* (respectively *infinite*) on the set of operators dealing with hypergraphs of type bounded by k, and L-labelled hyperedges. The finite hypergraph denoted by a finite term of $\mathfrak{T}_{L,k}$, i.e. the *value* of this term, is defined inductively according to the definitions of the semantics of each operator which has been above (see [8]). The value of an infinite term $t \in \mathfrak{T}^{\infty}_{L,k}$, which is an infinite hypergraph, can be defined in two ways: as the inductive limit of a sequence of finite hypergraphs defined from an increasing sequence of sub-terms of t, or as the quotient of a graph constructed from the set of leaves of t (see [8]). The value of a term is denoted by $\overline{t}$. *Any* finite hypergraph is the value of a term of some $\mathfrak{T}_{L,k}$, and *any* regular hypergraph[7] is the value of a (regular) term of some $\mathfrak{T}^{\infty}_{L,k}$ (cf. [8]). Besides, the concept of syntactic expressions allows to give an alternative definition of the *tree-width* (cf. [20] or Section 1 for original definition of the tree-width): the tree-width of a hypergraph is the smaller k such that it is the value of a term of $\mathfrak{T}_{L,k}$ (see [8]).

The syntactic expression operators are of arity 0, 1 or 2. Therefore, the terms on these operators can be encoded by possibly infinite binary trees labelled by operators. Here we use automata on such trees: *top-down tree automata* (see [7]) and *Rabin automata* (see [19]).

A *non-deterministic tree automaton* (a tree automaton for short) is a tuple $\mathcal{A} = (Q, \Sigma, \Delta, I)$ where Q is a set whose elements are the *states* of $\mathcal{A}$, Σ is the *alphabet* of $\mathcal{A}$, $\Delta \subset Q \times \Sigma \times Q^*$ is the set of *transitions* of $\mathcal{A}$, and $I \subset Q$ is the set of *initial states*. At the moment, we do not define any accepting condition; we will do it bellow for each kind of automaton. A *run* of an automaton $\mathcal{A} = (Q, \Sigma, \Delta, I)$ on a Σ-labelled tree t is a mapping $\rho : t \to Q_{\mathcal{A}}$ such that for any internal node x of t there exists a transition $(q, a, q_0 \dots q_{n-1}) \in \Delta$ such that $\rho(x) = q$, $t(x) = a$, x has exactly n sons and for each $s \in [\,0...n-1\,]$, $\rho(x.s) = q_s$. A *partial run* of $\mathcal{A}$ on a tree t is a mapping $\rho : t \to Q_{\mathcal{A}}$ such that for any node x of t there exists a transition $(q, a, q_0 \dots q_{n-1}) \in \Delta$ such that $\rho(x) = q$, $t(x) = a$, the number of sons of x is less or equal than n, and for any $s \in [\,0...n-1\,]$ such that $x.s$ belongs to t, $\rho(x.s) = q_s$. A *top-down automaton* is a tuple $\mathcal{A} = (Q, \Sigma, \Delta, I, F)$ where (Q, Σ, Δ, I) is an automata as defined above and $F \subset Q$ is the set of final states. A tree t is recognised by $\mathcal{A}$ if and only if it is finite and there exists a run $\rho : t \to Q$ such that $\rho(root(t)) \in I$ and for any leaf $x \in t$, $\rho(x) \in F$. The language of the trees which are recognised by $\mathcal{A}$ is denoted by $\mathcal{L}(\mathcal{A})$. A *Rabin Automaton* is a tuple $\mathcal{A} = (Q, \Sigma, \Delta, I, \{(E_1, F_1) \dots (E_n, F_n)\})$ where for any $i \in [\,1...n\,]$, $E_i, F_i \subset Q$. A tree T is recognised by $\mathcal{A}$ if and only if it is infinite and there exists a run $\rho : T \to Q$ such that $\rho(root(T)) \in I$ and for any infinite path β in T, for any $i \in [\,1...n\,]$, $E_i \cap \mathit{In}(\rho|\beta) = \emptyset$ and $F_i \cap \mathit{In}(\rho|\beta) \neq \emptyset$, where $\mathit{In}(\rho|\beta) = \{q \in Q \mid |\{x \in \beta \mid \rho(x) = q\}| = \infty\}$.

The classical link between automata and monadic second-order logic applies in this framework according to the following result (see [8]): For any set $\mathfrak{F}$ of finite graphs (respectively infinite graphs) of bounded tree-width k, there exists a MSO-formula φ such that $\mathfrak{F} = \mathcal{M}(\varphi)$ (respectively $\mathfrak{F} = \mathcal{M}^{\infty}(\varphi)$) if and only if there exists a top-down automaton $\mathcal{A}$ (respectively a Rabin automaton) such that $\mathfrak{F} = \{\overline{t} \mid t \in \mathfrak{T}_{L,k} \text{ and } t \in \mathcal{L}(\mathcal{A})\}$ (respectively $\mathfrak{F} = \{\overline{t} \mid t \in \mathfrak{T}^{\infty}_{L,k} \text{ and } t \in \mathcal{L}(\mathcal{A})\}$). One can

compute $\mathcal{A}$ from φ and conversely. When φ is given, the associated automaton shall be denoted by $\mathcal{A}_k(\varphi)$.

Composite Automata. Here we introduce the concept of *composite automata*. It translates the concept of graph grammar specification in terms of automata (Lemma 3 bellow). This is the first step for proving Theorem 1.

Let us fix an integer k. Let $\mathfrak{S} = (m_0, \{\varphi_m\}_{m\in\mathcal{M}})$ be a graph grammar specification. Let $(\mathcal{A}_k(\varphi_m))_{m\in\mathcal{M}}$ be some automata associated to the φ_m's as above. Without loss of generality, one can assume that each $\mathcal{A}_k(\varphi_m)$ is such that no transition of it has an initial state in its target vector. Let us consider the automaton $\mathcal{A}_k(\mathfrak{S})$ constructed from the $\mathcal{A}_k(\varphi_m)$'s as follows:
1. First, one consider the disjoint union of all the $\mathcal{A}_k(\varphi_m)$'s.
2. The terms recognised by $\mathcal{A}_k(\varphi_m)$ represents elements of $\mathcal{M}(\varphi_m)$. In such terms, the hyperedges encoding gluings regarding the grammar are encoded by sub-terms of form $E_{m'}$ for some m'. Since these last ones are of arity 0, these sub-terms actually are leaves of the complete term, when this last one is seen as a tree. Therefore, if we consider a run ρ of $\mathcal{A}_k(\varphi_m)$ on such a term, then ρ associates a terminal state to any sub-term of form $E_{m'}$. Let us consider such a terminal state q_f. Without loss of generality, one can assume that $\mathcal{A}_k(\varphi_m)$ is such that in any of its run, all the sub-terms to which q_f is associated are of the form $E_{m'}$ for a particular fixed m' depending only of q_f. One deletes q_f and replaces its occurrences in the target vector of any transition by an initial state of $\mathcal{A}_k(\varphi_{m'})$, duplicating the transition in question for each initial state.
3. The initial states are those of $\mathcal{A}_k(\varphi_{m_0})$.
4. The accepting condition for $\mathcal{A}_k(\mathfrak{S})$ shall be defined in Definition 1 bellow.

We distinguish in $\mathcal{A}_k(\mathfrak{S})$ all the states which are initial states of some $\mathcal{A}_k(\varphi_m)$; such a state is called a *boundary state*. $\mathcal{A}_k(\mathfrak{S})$ is called a *composite automaton*.

Induced Partitions. Roughly speaking, a run of a composite automaton on a given infinite tree induces a partition of the tree into finite subtrees, each of them being recognized by a composant of the automaton, i.e., the states coming from a particular $\mathcal{A}_k(\varphi_m)$. Such a run is accepting if, during the whole run, each composant of the automaton recognizes always the same finite tree (see Definition 1 bellow). Formally, let T be an infinite tree. A *partition* of T is a set $\{t_i\}_{i\in I}$ of finite subtrees of T such that the t_i's are pairwise disjoint and $T = \bigcup_i t_i$. Let $\{t_i\}_{i\in I}$ be such a partition. For any $i \in I$, let r_i denote the root of t_i. Let us note that I has a canonical non-ordered tree structure in which, for any pair $i, i' \in I$, i is the father of i' if and only if $r_{i'}$ is the son of some node of t_i. The set $\{r_i\}_{i\in I}$ is called the *internal boundary* of $\{t_i\}_{i\in I}$, it is denoted by $\mathcal{F}(\{t_i\}_{i\in I})$. We say that $\{t_i\}_{i\in I}$ is a *tilling* if there exists a finite subset $K \subset I$, assumed to be minimal, such that there exists a mapping $\lambda : I \to K$ such that for any $i \in I$, there exists an isomorphism $\sigma_i : t_i \to t_{\lambda(i)}$. The t_k's are called the *tiles*. $|K|$ is called the *index* of the tilling, it is denoted by $idx(K)$. Such a tilling is said to be *deterministic*[8] if for any $k \in K$ there exists a mapping $\delta_k : t_k \to (K \cup \bot)^*$ where $\bot$ is a new symbol not belonging to K defined as follows: for any $\bar{x} \in t_k$, the n-th element of $\delta_k(\bar{x})$, denoted by $\delta_k(\bar{x})_n$, is equal to $\bot$ if and only if for any $i \in \lambda^{-1}(k)$ and any $x \in \sigma_i^{-1}(\bar{x})$: the n-th son of x still is in t_i. And $\delta_k(\bar{x})_n = k'$ if and only if for any $i \in \lambda^{-1}(k)$ and any $x \in \sigma_i^{-1}(\bar{x})$, the n-th son of x does not belong to t_i and is equal to some $r_{i'}$ with $\lambda(i') = k'$. Let $d_{\max} = \max_{k,x}\{|\delta_k(x)|\}$. Let us assume that there exists an run

$\rho : T \to Q_{\mathcal{A}_k(\mathfrak{S})}$ of $\mathcal{A}_k(\mathfrak{S})$ on T such that there exists a partition made of finite trees defined by the property that $\mathcal{F}(\mathfrak{P}(\rho))$ is the inverse image by ρ of the set of boundary states of $\mathcal{A}_k(\mathfrak{S})$ This partition is called the *induced partition* of T regarding ρ, it is denoted by $\mathfrak{P}(\rho)$. Let us consider the mapping $\tau : Q_{\mathcal{A}_k(\mathfrak{S})} \to \mathcal{M}$ defined by $\tau(q) = m$ if and only if q belongs to $\mathcal{A}_k(\varphi_m)$.

Definition 1 (Rigid Accepting Condition). *A run $\rho : T \to Q_{\mathcal{A}_k(\mathfrak{S})}$ is said to be* accepting *if $\mathfrak{P}(\rho)$ is a tilling $(K, \lambda, \{\sigma_i\}_{i\in I})$, and for any pair $i_1, i_2 \in I$, $\tau(\rho(r_{i_1})) = \tau(\rho(r_{i_2}))$ implies that $\lambda(i_1) = \lambda(i_2)$. The language of $\mathcal{A}_k(\mathfrak{S})$, denoted by $\mathcal{L}^\infty(\mathcal{A}_k(\mathfrak{S}))$, is defined to be the set of trees for which such a run exists.*

Lemma 3. *A graph G belongs to $\mathcal{R}g_k(\mathfrak{S})$ if and only if it has a syntactic expression tree $T \in \mathcal{L}^\infty(\mathcal{A}_k(\mathfrak{S}))$.*

Proof. Let us be given with a run $\rho : T \to Q_{\mathcal{A}_k(\mathfrak{S})}$ such as considered in Definition 1. By construction of $\mathcal{A}_k(\mathfrak{S})$, T is indeed obtained by gluing some finite pieces recognised by the φ_m's: each maximal sub-runs of ρ only using states of some $\mathcal{A}_k(\varphi_m)$ actually recognises, up to the leaves, a term encoding a finite model of φ_m. Here we consider sub-runs using some initial state only one time. The induced partition $\mathfrak{P}(\rho)$ cuts T into pieces which are these maximal sub-runs of the $\mathcal{A}_k(\varphi_m)$. Definition 1 imposes that all the tiles associated to $\mathcal{A}_k(\varphi_m)$ are the same, i.e., all along the run ρ, $\mathcal{A}_k(\varphi_m)$ recognises always the same term. Keeping the notation of Section 2.1, up to the leaves, this term encodes g_m. T indeed encodes a solution.

Conversely, being given with a solution of $\mathcal{R}g_k(\mathfrak{S})$, one consider the grammar $(g_m)_{m\in\mathcal{M}}$ which has been used to construct it. This last one belongs to $\mathcal{G}r(\mathfrak{S})$. For each m, one chooses a term encoding g_m. And by gluing copies of these terms, one constructs an infinite term T, which finally satisfies Definition 1. This completes the proof.

Let us note however that some infinite terms encoding a solution in $\mathcal{R}g_k(\mathfrak{S})$ are not recognised by the composite automaton. Such a term can be constructed by choosing several distinct terms for one g_m. □

Remark 3. In particular $\mathcal{L}^\infty(\mathcal{A}_k(\mathfrak{S}))$ only depends on $\mathfrak{S}$ and m_0.

Remark 4. The tilling is deterministic because the transitions between tiles are determined by the grammar which is deterministic. So, the accepted trees are regular.

Simplification of Accepting Runs. Let $\rho : T \to Q_{\mathcal{A}_k(\mathfrak{S})}$ be an accepting run. Here we keep the notations of Definition 1. For any $m \in \mathcal{M}$, let $k_m \in K$ be such that $\tau(\rho(r_{k_m})) = m$. Let us note that k_m is unique because K is assumed to be minimal. In particular $idx(\mathfrak{P}(\rho)) \leq |\mathcal{M}|$. In order to simplify notations, let t_m (respectively r_m) denote t_{k_m} (respectively r_{k_m}). Let us define $\sigma : T \to \bigcup_m t_m$ the canonical extension of all the σ_i defined for any $n \in T$ by $\sigma(n) = \sigma_i(n)$ where t_i is the subtree of $\mathfrak{P}(\rho)$ which contains n. Let us now consider for any $m \in \mathcal{M}$ the mapping $\rho_m = \rho|_{t_m} : t_m \to Q_{\mathcal{A}_k(\varphi_m)} \subset Q_{\mathcal{A}_k(\mathfrak{S})}$. The family $\{\rho_m\}_{m\in\mathcal{M}}$ is denoted by Δ_ρ. Let $\lambda_\rho : I \to \mathcal{M}$ be defined by $\lambda_\rho(i) = \tau(\rho(r_i))$ for any $i \in I$. Finally, let us consider the mapping $\bar{\rho} : T \to Q_{\mathcal{A}_k(\mathfrak{S})}$ defined by $\forall n \in T : \bar{\rho}(n) = \rho_{\tau(\rho(n))}(\sigma(n))$. We admit the following result which is easy.

Lemma 4. *$\bar{\rho}$ is an accepting run of $\mathcal{A}_k(\mathfrak{S})$.*

Such a run is called a *simplified* accepting run.

Remark 5. Simplified Runs are *Constructive.* T is completely determined by the family $\{t_m\}_{m\in\mathcal{M}}$. And $\{(t_m,\rho_m)\}_{m\in\mathcal{M}}$ completely defines $\bar{\rho}$. Moreover, being given with a set of such pairs, one can effectively decide whether it defines an accepting run or not.

2.3 Tree Tillings and Rabin Automata

By Lemma 3, Problem 3 is equivalent to decide whether the intersection of the languages of a given composite automaton and a given Rabin automaton is empty or not. Here we study the runs of a Rabin automaton on trees which are recognized by some composite automaton. We focus on the proof of Lemma 5 bellow which aims at simplifying such runs. This is a corner-stone of the proof the Theorem 1. Roughly speaking, it says that if a tree which has a deterministic tilling is recognized by a Rabin automaton, then there is a run which uses a bounded number of partial runs on each tile; the bound only depending on the number of tiles (and not on their sizes) and on the number of states of the automaton. By Remark 4 above, this applies to trees which are simultaneously recognized by a composite automaton and a Rabin automaton.

Formally, let us consider a Rabin automaton $\mathcal{A}'$. Let $T\in\mathcal{L}(\mathcal{A}')$, let us assume that there exists a tilling $\mathcal{T}=\{t_i\}_{i\in I}$ of T defined by a tuple $(K,\lambda,(\sigma_i)_{i\in I})$. Let $\rho':T\to Q_{\mathcal{A}'}$ be an accepting run of $\mathcal{A}'$ on T. Let us consider the collection of mappings of the form $\rho'\circ\sigma_i^{-1}:t_{\lambda(i)}\to Q_{\mathcal{A}'}$ for $i\in I$. Since all the t_k's are finite and their number is also finite, this collection of mappings is actually finite. Let us index it by a finite set J and denote it by $\Delta'_{\mathcal{T},\rho'}=\{\rho'_j:t_{k_j}\to Q_{\mathcal{A}'}\}_{j\in J}$ where $k_j\in K$. According to this notation, let $\lambda'_{\mathcal{T},\rho'}:I\to J$ be the mapping such that for any $i\in I$: $\rho'|_{t_i}\equiv\rho'_{\lambda'_{\mathcal{T},\rho'}(i)}\circ\sigma_i$. $|\Delta'_{\mathcal{T},\rho'}|$ is called the *regularity index* of ρ' regarding $\mathcal{T}$, it is denoted by $reg(\mathcal{T},\rho')$.

Lemma 5. *Let $\mathcal{A}'$ be a Rabin automaton and let $T\in\mathcal{L}(\mathcal{A}')$ having a deterministic tilling $\mathcal{T}$. Then there exists an accepting run $\rho':T\to Q_{\mathcal{A}'}$ on T such that $reg(\mathcal{T},\rho')$ is bounded by $2^{2\times|Q_{\mathcal{A}'}|\times|\Sigma_{\mathcal{A}'}|\times\max\{idx(K),|Q_{\mathcal{A}'}|\}^{d_{\max}}\times 2}$*

The Game Point of View. The proof is based on the concept of games. A *game* $\mathcal{G}$ is defined according to a *game graph* and a *winning language*. A game graph is a tuple (V_0,V_1,E,c,C) where V_0 and V_1 are disjoint at most countable sets of vertices, their union $V=V_0\cup V_1$ is the set of *game positions*; $E\subset(V_0\times V_1)\cup(V_1\times V_0)$ is an edge relation such that for each vertex the set of outgoing edges is non-empty and finite, the elements of E are called the *moves*; C is a set and $c:V\to C$ is a map called the *colouring*. A winning language W is a ω-language of infinite words over C, i.e., $W\subset C^\omega$. A *play* in $\mathcal{G}$ is a ω-word $\gamma\in(V_1V_0)^\omega$; player$_0$ wins γ if $c^\omega(\gamma)\in W$, where $c^\omega:V^\omega\to C^\omega$ denotes the canonical extension of c. A *strategy* for player$_0$ is a mapping $\mathcal{S}:D_{\mathcal{S}}\subset V_0^*\to V_1$ where $D_{\mathcal{S}}$ satisfies the following condition: $\varepsilon\in D_{\mathcal{S}}$ and for any $(w,v)\in V_0^*\times V_0$, $w.v\in D_{\mathcal{S}}$ if and only if $w\in D_{\mathcal{S}}$ and there is a move from $\mathcal{S}(w)$ to v. In this case, there must be a move from v to $\mathcal{S}(w.v)$. We say that player$_0$ follows $\mathcal{S}$ in γ if for each $i\geq 0:\gamma(2i)=\mathcal{S}(\gamma(1)\gamma(3)...\gamma(2i-1))$. We say that $\mathcal{S}$ is a *winning strategy* if any play in which player$_0$ follows $\mathcal{S}$ is winning.

Let us be given with $T\in\mathcal{L}(\mathcal{A}')$ and a deterministic tilling $\mathcal{T}=\{t_i\}_{i\in I}$ defined by a tuple $(K,\lambda,\{\sigma_i\}_{i\in I},\{\delta_k\}_{k\in K})$. Let us consider a finite set of mappings $\Delta=\{\rho'_j:t_{k_j}\to Q_{\mathcal{A}'}\}_{j\in J}$ where $k_j\in K$ such that for any $j\in J$, ρ'_j is a partial run of $\mathcal{A}'$ on t_{k_j}. For each $j\in J$ and for each node x of

t_{k_j}, let us consider the tuple $\xi_{j,x} = (q, \eta, w, p) \in Q_{\mathcal{A}'} \times \Sigma_{\mathcal{A}'} \times (K \cup Q_{\mathcal{A}'})^{*<d_{\max}} \times \mathcal{P}(Q_{\mathcal{A}'})$ defined as follows: $q = \rho'_j(x)$ and η is the label of x. The word w is obtained as follows: by definition of $\delta_{k_j}(x)$, for any position n such that $\delta_{k_j}(x)_n = \perp$, the n-th son of x is defined in t_{k_j}, let us denote it by x'. Then one puts $\rho'_j(x')$ at the n-th position in $\delta_{k_j}(x)$ in the place of $\perp$. Finally $p = \{\rho'_j(y) \mid y \in \alpha\}$ where α denotes the path of t_{k_j} starting at the root and ending at x. Let ξ_j denote the set of all the $(q, \eta, w, p) = \xi_{j,x}$ such that $x \in t_{k_j}$ and w contains at least one element of K.

For any transition $\tau = (q, \eta, (q_1, \ldots, q_{n'}))$ of $\mathcal{A}'$ and any $w \in (K \cup Q_{\mathcal{A}'})^{*<d_{\max}}$, let $\chi_{\tau,w} \subset J^{*<d_{\max}}$ be the set of words of form $j_0 \ldots j_n$ where n is the number of positions where some element of K occurs in w, and for any $\ell \in [\,1...n\,]$, $j_\ell \in J$ is such that k_{j_ℓ} is equal to the ℓ-th element of K occurring in w and $(q_1, \ldots, q_{n'})$ is obtained from w by replacing the ℓ-th element of K occurring in w by $\rho'_{j_\ell}(r_{k_{j_\ell}})$.

Let $\mathcal{G}_\Delta$ be the finite two-players game defined by the finite game graph $\Gamma_\Delta = (V_0, V_1, E, c, C)$ and the winning ω-language W where $V_0 = Q_{\mathcal{A}'} \times \Sigma_{\mathcal{A}'} \times (K \cup Q_{\mathcal{A}'})^{*<d_{\max}} \times \mathcal{P}(Q_{\mathcal{A}'})$ is the set of positions for player$_1$, $V_1 = J^{*<d_{\max}}$ is the set of positions for player$_0$, the set of transitions E groups transitions of the form $(q, \eta, w, p) \to w'$ such that $w' \in \chi_{\tau,w}$ where τ is a transition of $\mathcal{A}'$ of origin (q, η), and transitions of the form $j_0 \ldots j_n \to (q, \eta, w, p)$ such that $\exists e \in [\,1...n\,]$ such that $(q, \eta, w, p) \in \xi_{j_e}$, the colour set is $C = {Q_{\mathcal{A}'}}^*$ and $c : V \to C$ is defined by $c(v) = \varepsilon$ for any $v \in V_1$, and $c(q, \eta, w, p) = \tilde{p}$, where $\tilde{p} \in {Q_{\mathcal{A}'}}^*$ is defined to be the ordered sequence of all the elements of p according to a linear ordering of $Q_{\mathcal{A}'}$ fixed in advance once for all, finally, W is defined to be the language of infinite words accepted according to the acceptance condition of $\mathcal{A}'$ and starting at some initial state of $\mathcal{A}'$.

Following the standard schema, we show that the concepts of runs and strategies coincide. Let ρ' be a run of $\mathcal{A}'$ on T. Let $\Delta'_{\mathcal{T},\rho'}$ and $\lambda'_{\mathcal{T},\rho'}$ be as defined page 11. Then ρ' defines a strategy $\mathcal{S}_{\rho'}$ in the game $\mathcal{G}_{\Delta'_{\mathcal{T},\rho'}}$ associated to $\Delta'_{\mathcal{T},\rho'}$. Here we define $\mathcal{S}_{\rho'}$ by describing how it drives a play:

• The first move of player$_0$ is $j_{0,0} = \rho'(root(T))$.

• Next, the first move of player$_1$ is defined as follows: According to the definition of $\mathcal{G}_{\Delta'_{\mathcal{T},\rho'}}$, player$_1$ picks a node $\bar{x}_1 \in t_{k_{j_{0,0}}}$ such that $\xi_{j_{0,0},\bar{x}_1} \in \xi_{j_{0,0}}$; and he plays the tuple $(q_1, \eta_1, w_1, p_1) = \xi_{j_{0,0},\bar{x}_1}$.

• Let us consider the play at the turn number $\ell \geq 1$:

let $\gamma_\ell = \underbrace{j_{0,0}}_{\text{player}_0}\ \underbrace{(q_1, \eta_1, w_1, p_1)}_{\text{player}_1}\ \underbrace{j_{1,0}j_{1,1}\ldots j_{1,n_1}}_{\text{player}_0}\ \underbrace{(q_2, \eta_2, w_2, p_2)}_{\text{player}_1} \ldots$

$$\ldots \underbrace{j_{\ell-1,0}j_{\ell-1,1}\ldots j_{\ell-1,n_{\ell-1}}}_{\text{player}_0}\ \underbrace{(q_\ell, \eta_\ell, w_\ell, p_\ell)}_{\text{player}_1}$$

be the successive positions which have appeared before. By induction on $\ell \geq 0$, let us assume that

• there exists a sequence $t_{i_0}, \ldots, t_{i_{\ell-1}}$ of ℓ elements of $\mathfrak{P}(\rho)$ such that i_0 is the root of I according to the natural tree structure of I, i.e., r_{i_0} is the root of T (cf. page 9), and for any $h \in [\,1...\ell - 2\,]$: i_h is the father of i_{h+1}; in particular, the concatenation of all the α_h's is also a path of T;

• there exists a sequence of paths $\alpha_1 \subset t_{i_0}, \ldots, \alpha_\ell \subset t_{i_{\ell-1}}$ such that for any $h \in [\,1...\ell\,]$, α_h starts at the root $r_{i_{h-1}}$ of $t_{i_{h-1}}$ and ends at a node x_h which is the father of r_{i_h} if $h \leq \ell - 1$;

• for any $h \in [\,0...\ell - 1\,]$: there exists $e_h \in [\,1...n_h\,]$ such that $(q_{h+1}, \eta_{h+1}, w_{h+1}, p_{h+1}) = \xi_{j_{h,e_h},\sigma_{i_h}(x_{h+1})}$.

The move of player$_0$. The strategy $\mathcal{S}_{\rho'}$ defined by ρ' is defined here: player$_0$ plays a word $j_{\ell,0}j_{\ell,1}\dots j_{\ell,n_\ell}$ defined as follows: n_ℓ is the number of elements of K occurring in w_ℓ. For any $h \in [\,1...n_\ell\,]$, let l_h be the position of the h-th element of K occurring in w_ℓ, and let $s_h \in I$ be such that r_{s_h} is h-th son of x_ℓ. Then for any $h \in [\,1...n_\ell\,]$, we define $j_h = \lambda'_{\mathcal{T},\rho'}(s_h)$. This move is correct: Let τ be the transition of $\mathcal{A}'$ used by ρ' at x_ℓ. Then by construction we have that η_ℓ is the label of x_ℓ and $q_\ell = \rho'(x_\ell)$. Therefore (q_ℓ, η_ℓ) is indeed the origin of τ. By a detailed checking of the definition of χ_{τ,w_ℓ}, one also can verify that $j_{\ell,0}j_{\ell,1}\dots j_{\ell,n_\ell} \in \chi_{\tau,w_\ell}$. The move is indeed correct.
The move of player$_1$. According to the definition of $\mathcal{G}_{\Delta'_{\mathcal{T},\rho'}}$, player$_1$ picks an integer $e_\ell \in [\,1...n_\ell\,]$ and a node $\bar{x}_{\ell+1} \in t_{k_{j_{\ell,e_\ell}}}$ such that $\xi_{j_{\ell,e_\ell},\bar{x}_{\ell+1}} \in \xi_{j_{\ell,e_\ell}}$; and he plays the tuple $(q_{\ell+1}, \eta_{\ell+1}, w_{\ell+1}, p_{\ell+1}) = \xi_{j_{\ell,e_\ell},\bar{x}_{\ell+1}}$.
Construction of $\alpha_{\ell+1}$ *and* t_{i_ℓ}. Let $\bar{\alpha}_{\ell+1}$ be the path going from the root of $t_{k_{j_{\ell,e_\ell}}}$ to $\bar{x}_{\ell+1}$. Let e'_ℓ be position in w_ℓ of the e_ℓ-th element of K occurring in it. Then i_ℓ is defined by the condition that r_{i_ℓ} is the e'_ℓ-th son of x_ℓ. Let us note that we have $\lambda(i_\ell) = (w_\ell)_{e'_\ell}$. Finally $\alpha_{\ell+1}$ is defined to be $\sigma_{i_\ell}^{-1}(\bar{\alpha})$. And $x_{\ell+1} = \sigma_{i_\ell}^{-1}(\bar{x}_{\ell+1})$.

The induction hypothesis are obviously verified by construction. Finally, we get the following result:

Lemma 6. *For any run ρ' of $\mathcal{A}'$, if ρ' is an accepting run then $\mathcal{S}_{\rho'}$ is a winning strategy.*

Proof. It remains to verify that $\mathcal{S}_{\rho'}$ is winning if ρ' is accepting. Let us then suppose that ρ' is accepting. Let us consider a whole play

$$\begin{array}{lll}
\text{let } \gamma_\ell = & j_{0,0} & (\text{player}_0)\\
& (q_1, \eta_1, w_1, p_1) & (\text{player}_1)\\
& j_{1,0}j_{1,1}\cdots j_{1,n_1} & (\text{player}_0)\\
& (q_2, \eta_2, w_2, p_2) & (\text{player}_1)\\
& \dots & \\
& j_{\ell-1,0}j_{\ell-1,1}\cdots j_{\ell-1,n_{\ell-1}} & (\text{player}_0)\\
& (q_\ell, \eta_\ell, w_\ell, p_\ell) & (\text{player}_1)\\
& \dots &
\end{array}$$

Then γ is winning if and only if $c^\omega(\gamma)$ belongs to the winning language, i.e., the satisfying the accepting condition of $\mathcal{A}'$. But this last one only depends on the set of states which appear infinitely in $c^\omega(\gamma)$.

Let us write

$$c^\omega(\gamma) = c((q_1, \eta_1, w_1, p_1))\dots c((q_\ell, \eta_\ell, w_\ell, p_\ell))\dots$$

Then

$$c^\omega(\gamma) = \tilde{p}_1\tilde{p}_2\dots\tilde{p}_\ell\dots$$

But as we have seen above, $p_\ell = \rho'(\alpha_\ell)$ for any ℓ. And thus, $\tilde{p}_\ell$ and $\rho'(\alpha_\ell)$ have the same set of states. Therefore any state appearing infinitely many times in $\tilde{p}_1\tilde{p}_2\dots\tilde{p}_\ell\dots$ also appears infinitely many times in $\rho'(\alpha_1)\rho'(\alpha_2)\dots\rho'(\alpha_\ell)\dots$, i.e.,

$$\mathit{In}(\tilde{p}_1\tilde{p}_2\dots\tilde{p}_\ell\dots) = \mathit{In}(\rho'(\alpha_1)\rho'(\alpha_2)\dots\rho'(\alpha_\ell)\dots)$$

And thus $In(c^\omega(\gamma)) = In(\rho'(\alpha_1)\rho'(\alpha_2)\dots\rho'(\alpha_\ell)\dots) = In(\rho'(\alpha_1\alpha_2\dots\alpha_\ell\dots))$.

Since ρ' is accepting, $In(\rho'(\alpha_1\alpha_2\dots\alpha_\ell\dots))$ satisfies the accepting condition. And therefore, so does $In(c^\omega(\gamma))$. This completes the proof. □

Let us turn to the converse. Let $\mathcal{S}$ be a strategy in $\mathcal{G}_\Delta$. Let $j_0 = \mathcal{S}(\varepsilon)$. Let us suppose that $\rho'_{j_0}(r_{k_{j_0}})$ is an initial state of $\mathcal{A}'$. Such a strategy is said to *have the property* $\mathcal{P}_{init}$. Assuming that, $\mathcal{S}$ defines a run $\rho'_{\mathcal{S}}$ of $\mathcal{A}'$ such that $\Delta'_{\mathcal{T},\rho'_{\mathcal{S}}} \subset \{\rho'_j \in \Delta \mid j \in \mathcal{D}om(\mathcal{S})\}$. To define $\rho'_{\mathcal{S}}$, we shall construct the mapping $\lambda'_{\mathcal{T},\rho'_{\mathcal{S}}} : I \to J$ associated to $\rho'_{\mathcal{S}}$.

Let $i \in I$. Let β be the path starting at the root of T and ending at r_i, the root of t_i. Let $t_{i_0}, ..., t_{i_\theta}$ be the sequence of elements of $\mathfrak{P}(\rho)$ across which β goes; β ends at r_{i_θ} (see Figure 1). For each

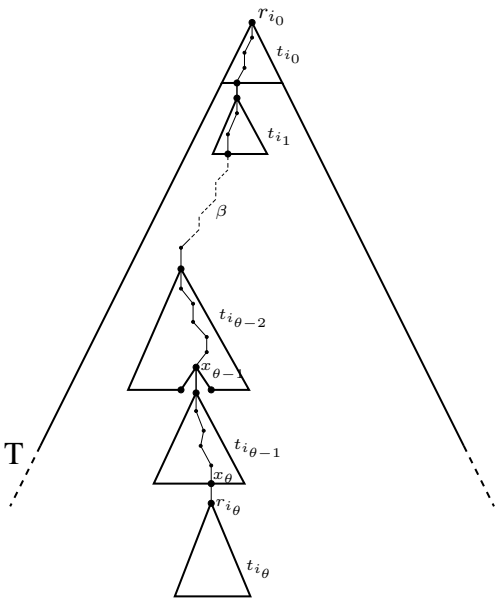

Fig. 1. Construction of ρ'

$h \in [\,1...\theta\,]$, let $x_h \in t_{i_{h-1}}$ be the father of r_{i_h}. Then $x_h \in \beta$. Let e_h be such that r_{i_h} is the e_h-th son of x_h which does not belong to $t_{i_{h-1}}$; and let $e'_h \in [\,1...n_h\,]$ be such that r_{i_h} is the e'_h-th son of x_h. Let us consider the partial play defined as follows: According to $\mathcal{S}$, the first move of player$_0$ is $j_{0,0} = \mathcal{S}(\varepsilon)$. Then, inductively, let us consider the play at the turn number $\ell \leq \theta$:

let $\gamma_\ell = \underbrace{j_{0,0}}_{\text{player}_0}\ \underbrace{(q_1,\eta_1,w_1,p_1)}_{\text{player}_1}\ \underbrace{j_{1,0}j_{1,1}\dots j_{1,n_1}}_{\text{player}_0}\ \underbrace{(q_2,\eta_2,w_2,p_2)}_{\text{player}_1}\dots$

$$\dots\ \underbrace{j_{\ell-1,0}j_{\ell-1,1}\dots j_{\ell-1,n_{\ell-1}}}_{\text{player}_0}\ \underbrace{(q_\ell,\eta_\ell,w_\ell,p_\ell)}_{\text{player}_1}\ \underbrace{j_{\ell,0}j_{\ell,1}\dots j_{\ell,n_\ell}}_{\text{player}_0}$$

be the successive positions which have appeared before. We assume that the successive game positions $j_{0,0} - j_{1,0}j_{1,1}\dots j_{1,n_1} - \dots - j_{\ell,0}j_{\ell,1}\dots j_{\ell,n_\ell}$ have been played by player$_0$ according to $\mathcal{S}$.

If $\ell = \theta$, then the construction is finite. Otherwise, we define the next moves of player$_0$ and player$_1$: First, let us deal with the next move of player$_1$. Let us note that $\lambda(i_\ell) = k_{j_{\ell,e_\ell}}$. Therefore $\bar{x}_{\ell+1} = \sigma_{i_\ell}(x_{\ell+1})$ belongs to $t_{k_{j_{\ell,e_\ell}}}$. Then we define the next move of player$_1$ to be $(q_{\ell+1}, \eta_{\ell+1}, w_{\ell+1}, p_{\ell+1}) = \xi_{j_{\ell,e_\ell},\bar{x}_{\ell+1}}$. Second, player$_0$ plays according $\mathcal{S}$:

$$j_{\ell+1,0}j_{\ell+1,1}\dots j_{\ell+1,n_{\ell+1}} = \mathcal{S}((q_1,\eta_1,w_1,p_1)\dots(q_{\ell+1},\eta_{\ell+1},w_{\ell+1},p_{\ell+1})) \tag{1}$$

By induction, we construct a play γ_θ. And finally, we set: $\lambda'_{\mathcal{T},\rho'}(i) = j_{\theta,e_\theta}$.

Lemma 7. *For any strategy $\mathcal{S}$ of player$_0$ in $\mathcal{G}_\Delta$, if $\mathcal{S}$ is a winning strategy and has the property $\mathcal{P}_{init}$, then $\rho'_{\mathcal{S}}$ is an accepting run.*

Proof. First, $\rho'_{\mathcal{S}}$ is indeed a run of $\mathcal{A}'$. The previous construction defines the pair $(\Delta'_{\mathcal{T},\rho'}, \lambda'_{\mathcal{T},\rho'})$ which defines the mapping $\rho' : T = \bigcup_i t_i \to Q_{\mathcal{A}'}$. Concerning initial condition, $\rho'(root(T))$ is indeed an initial state of $\mathcal{A}'$ because $\mathcal{S}$ has the property $\mathcal{P}_{\text{init}}$. Concerning transitions, at any node x of T which is internal to some t_i, i.e., such that x and its sons are all included in one t_i, ρ' uses the transition of $\mathcal{A}'$ that $\rho'_{\lambda'_{\mathcal{T},\rho'}(i)}$ uses at $\sigma_i(x)$; let us recall that $\rho'_{\lambda'_{\mathcal{T},\rho'}(i)}$ is a partial run of $\mathcal{A}'$. If x is not internal, then one consider the previous construction of ρ' by choosing the path from the root of T to a son of x not belonging to t_i as β. Then $x = x_\theta$. Let us consider the last move of player$_0$ in γ_θ as defined in (1). Let us recall that player$_0$ always plays according to $\mathcal{S}$ in γ_θ. Then according to the definition of $\mathcal{G}_\Delta$, there exists a transition τ such that this last move has the form $(q_\theta, \eta_\theta, w_\theta, p_\theta) \to w'$ with $w' \in \chi_{\tau,w_\theta}$. Finally, according to the definition of χ_{τ,w_θ}, one can verify that ρ' exactly uses the transition τ at $x = x_\theta$.

The verification that ρ' is accepting is similar to the function used in proof of Lemma 6, we shall not detail it. □

Lemma 8 (Strategy Reduction). *Let $\mathcal{G}$ be a game defined by a game graph $\Gamma = (V_0, V_1, E, c, C)$ with $|V_1|$ and $|C|$ finite. Let us suppose that there exists a winning strategy $\mathcal{S}$ for player$_0$ in $\mathcal{G}$. Then there exists an other strategy $\mathcal{S}'$ for player$_0$ in $\mathcal{G}$ such that $|\mathcal{S}'(\mathcal{D}om(\mathcal{S}'))| \leq |c(V_1)| \times 2^{2\times|V_0|}$. Moreover, $\mathcal{S}'$ can be computed in an effective way from $\mathcal{S}$.*

Proof. For any $v \in V_1$, let $\text{Pred}_v = \{v' \in V_0 \mid (v', v) \in E\}$ and let $\text{Succ}_v = \{v' \in V_0 \mid (v, v') \in E\}$. Let us consider the equivalence relation $\sim_\Gamma$ on $V_0 \cup V_1$ defined by: for any pair of vertices v and v' of $V_0 \cup V_1$, $v \sim_\Gamma v'$ if and only if $c(v) = c(v')$ and $\text{Pred}_v = \text{Pred}_{v'}$ and $\text{Succ}_v = \text{Succ}_{v'}$.

Let us consider $v \in V_0 \cup V_1$. Then it is equivalent to play v instead of any game position which is equivalent to it. Formally, let us consider the mapping π_v from $V_0 \cup V_1$ onto $V_0 \cup V_1 \setminus \{v' \mid v' \sim_\Gamma v \text{ and } v' \neq v\}$ defined by $\pi_v(w) = v$ if $w \sim_\Gamma v$, and $\pi_v = w$ otherwise. Let us consider the canonical extension $\pi_v^\omega : (V_0 \cup V_1)^\omega \to (V_0 \cup V_1)^\omega$ defined by $\pi_v^\omega(w_0 w_1 \dots w_n \dots) = \pi_v(w_0)\pi_v(w_1)\dots\pi_v(w_n)\dots$. Then for any play γ of $\mathcal{G}$, $\pi_v^\omega(\gamma)$ is also a play of $\mathcal{G}$, and $\pi_v^\omega(\gamma)$ is winning for player$_0$ if and only if γ is. And for any strategy $\mathcal{S}$, $\pi_v \circ \mathcal{S}$ is also a strategy. Moreover $\pi_v \circ \mathcal{S}$ is winning if and only if $\mathcal{S}$ is.

Let us consider a strategy $\mathcal{S}$ for player$_0$ in $\mathcal{G}$. Let us pick $v_1, v_2, \ldots, v_h \in V_1$ a collection of pairwise non $\sim_\Gamma$-equivalent vertices such that $V_1 = \bigcup_{n=1\ldots h}[v_n]_{\sim_\Gamma}$, i.e., a set of representatives of $\sim_\Gamma$-classes. And let us consider $\mathcal{S}' = \pi_{v_1} \circ \pi_{v_2} \circ \ldots \pi_{v_1} \circ \mathcal{S}$. Then $\mathcal{S}'(\mathcal{D}om(\mathcal{S}')) \subset \{v_1, v_2, \ldots, v_h\}$, and therefore $|\mathcal{S}'(\mathcal{D}om(\mathcal{S}'))|$ is less than the number of equivalence classes regarding $\sim_\Gamma$.

Let us note that the equivalence class of a vertex $v \in V_1$ is determined by the tuple $(c(v), \text{Pred}_v, \text{Succ}_v)$ of $c(V_1) \times \mathcal{P}(V_0) \times \mathcal{P}(V_0)$. Therefore, the number of equivalence class is bounded by $|c(V_1)| \times 2^{2|V_0|}$, which concludes the proof of the lemma. □

Remark 6. Taking the framework of $\mathcal{G}_\Delta$, if $\mathcal{S}$ has the property $\mathcal{P}_{\text{init}}$, then $\mathcal{S}'$ can be constructed in such a way that it also the property $\mathcal{P}_{\text{init}}$. Indeed, the construction of $\mathcal{S}'$ is based on the choice of a system of representatives of $\sim_\Gamma$-classes. In this process, one can choose $j_{0,0}$ as the representative of its class, assuring in this way the property $\mathcal{P}_{\text{init}}$.

Corollary 1. *If there exists a winning strategy $\mathcal{S}$ for player$_0$ in $\mathcal{G}_\Delta$, then there also exists a winning strategy $\mathcal{S}'$ for player$_0$ in $\mathcal{G}_\Delta$ such that $|\mathcal{S}'(\mathcal{D}om(\mathcal{S}'))|$ is bounded by $2^{2\times|Q_{\mathcal{A}'}|\times|\Sigma_{\mathcal{A}'}|\times\max\{idx(K),|Q_{\mathcal{A}'}|\}^{d_{\max}}\times 2^{|Q_{\mathcal{A}'}|}}$.*

Proof. First, let us note that $|c(V_1)| = |\{\varepsilon\}| = 1$. Second, we have that $|V_0|$ is bounded by $|Q_{\mathcal{A}'}| \times |\Sigma_{\mathcal{A}'}| \times \max\{idx(K), |$ We get then the result by Lemma 8. □

Proof of Lemma 5. Let us be given with an accepting run $\tilde{\rho}'$ on $T \in \mathcal{L}(\mathcal{A}')$. By Lemma 6, $\tilde{\rho}'$ defines a winning strategy $\mathcal{S}_{\tilde{\rho}'}$ for player$_0$ in $\mathcal{G}_{\Delta'_{T,\tilde{\rho}'}}$. Moreover, $\mathcal{S}_{\tilde{\rho}'}$ has the property $\mathcal{P}_{\text{init}}$. By Lemma 8, we can construct from $\mathcal{S}_{\tilde{\rho}'}$ a new winning strategy $\mathcal{S}'$ for player$_0$ such that $|\mathcal{S}'(\mathcal{D}om(\mathcal{S}'))|$ is bounded by $2^{2\times|Q_{\mathcal{A}'}|\times|\Sigma_{\mathcal{A}'}|\times\max\{idx(K),|Q_{\mathcal{A}'}|\}^{d_{\max}}\times 2^{|Q_{\mathcal{A}'}|}}$. Taking into account Remark 8, one can assume that $\mathcal{S}'$ has the property $\mathcal{P}_{\text{init}}$. By Lemma 7, $\mathcal{S}'$ defines an accepting run $\rho_{\mathcal{S}'}$ which satisfies the conditions of Lemma 5 □

2.4 Decidability

Decidability results from a pumping property obtained from Lemma 5: if the langages of a composite automaton and a Rabin automaton are not disjoint, then there is tree in their intersection which has a deterministic tilling with *bounded* tiles (Lemma 9 bellow). Deciding if such a tree exists is therefore possible by enumeration.

Formally, let $\mathfrak{S} = (m_0, \{\varphi_m\}_{m\in\mathcal{M}})$ be a graph grammar specification. Let $(\mathcal{A}_k(\varphi_m))_{m\in\mathcal{M}}$ and $\mathcal{A}_k(\mathfrak{S})$ be such as in Section 2.2. Let $T \in \mathcal{L}^\infty(\mathcal{A}_k(\mathfrak{S}))$. Let $\rho : T \to Q_{\mathcal{A}_k(\mathfrak{S})}$ be an accepting run of $\mathcal{A}_k(\mathfrak{S})$ on T, and let $\mathfrak{P}(\rho) = \{t_i\}_{i\in I}$ be the induced partition of T associated to ρ. $\mathfrak{P}(\rho)$ is a deterministic tilling, let $(K, \lambda, \{\sigma_i\}_{i\in I}, \{\delta_k\}_{k\in K})$ be a tuple associated to it. We also consider the families $\{t_m\}_{m\in\mathcal{M}}$ and $\Delta_\rho = \{\rho_m : t_m \to Q_{\mathcal{A}_k(\varphi_m)}\}_{m\in\mathcal{M}}$, and the mapping $\lambda_\rho : I \to \mathcal{M}$ as defined at page 10.

Let $\mathcal{A}'$ a Rabin automaton, let us assume that $T \in \mathcal{L}(\mathcal{A}')$. Let $\rho' : T \to Q_{\mathcal{A}'}$ be an accepting run satisfying Lemma 5 regarding the deterministic tilling defined by $\mathfrak{P}(\rho)$. We keep the notations of Section 2.4. Let us consider $\Delta'_{\mathfrak{P}(\rho),\rho'} = \{\rho'_j : t_{k_j} \to Q_{\mathcal{A}'}\}_{j\in J}$ defined in Section and the associated mapping $\lambda'_{\mathfrak{P}(\rho),\rho'} : I \to J$.

Path Collapsing Let $\alpha = x_0...x_n$ be a depth-increasing path in T; x_0 is the higher node and x_n the lower one. *Collapsing* α in T consists in deleting T_{x_0} and gluing T_{x_n} in place of it in T. The resulting tree is denoted by $\zeta_\alpha(T)$. Formally, $\zeta_\alpha(T)$ is defined to be the quotient of T regarding the following equivalence relation, denoted by $\mathcal{R}_\alpha$: for any two nodes x and x', $x\mathcal{R}_\alpha x'$ if and only if they are equal or if they both belong to $T_{x_0} \setminus (T_{x_n} \setminus \{x_n\})$. For any $x \in T$, let $[x]$ denote the $\mathcal{R}_\alpha$-equivalence class of x. Let $\pi_\alpha : T \to \zeta_\alpha(T)$ be the associated projection.

Let $\rho : T \to Q'_{\mathcal{A}}$ be a run. The operation of collapsing α in T is said to be *compatible* with ρ if $\rho(x_0) = \rho(x_n)$. If this holds, then ρ induces a run on $\zeta_\alpha(T)$ which shall be denoted by $\zeta_\alpha(\rho)$ defined by $\zeta_\alpha(\rho)([x]) = \rho(x)$ for any $x \in (T \setminus T_{x_0}) \cup T_{x_n}$. Let us note that $\zeta_\alpha(\rho)([x])$ can be alternatively defined to be the image by ρ of the highest node belonging to $[x]$, i.e. x_0. For any family $\mathcal{P}$ of pairwise disjoint paths of T, possibly infinite, the operation of collapsing all the paths of $\mathcal{P}$ simultaneously in T, denoted by $\zeta_\mathcal{P}$, is well defined by considering the equivalence relation made of the union of all the equivalence relations (seen as subsets of $T \times T$) associated to the paths of $\mathcal{P}$: $\mathcal{R}_\mathcal{P} = \bigcup_{\alpha\in\mathcal{P}} \mathcal{R}_\alpha$. Let $\pi_\mathcal{P} : T \to \zeta_\mathcal{P}(T)$ denote the associated projection. If ζ_α is compatible with ρ for any $\alpha \in \mathcal{P}$, then one defines a canonical run $\zeta_\mathcal{P}(\rho)$ of $\mathcal{A}'$ on $\zeta_\mathcal{P}(T)$ as follows: $\zeta_\mathcal{P}(\rho)([x]) = \rho(x')$ where x' is the highest node belonging to $[x]$. Let α be a path of t_m for some fixed m. Such a path defines a family of pairwise disjoint paths in T defined by $\sigma^{-1}(\alpha) = \{\sigma_i^{-1}(\alpha) \subset T \mid \lambda_\rho(i) = m\}$. Let us consider the simultaneous collapsing of all the paths of $\sigma^{-1}(\alpha)$ in T, i.e., $\zeta_{\sigma^{-1}(\alpha)}$; let it be denoted by ζ_α^∞. We also consider the associated projection $\pi_\alpha^\infty : T \to \zeta_{\sigma^{-1}(\alpha)}(T)$ and the associated run $\zeta_\alpha^\infty(\rho)$. Similarly, ζ_α^∞ is compatible with ρ' if and only if ζ_α is compatible with ρ'_j for any $j \in J$ such that $\lambda_\rho(k_j) = m$. In this case $\zeta_\alpha^\infty(\rho')$ still is a run of $\mathcal{A}'$ on $\zeta_\alpha^\infty(T)$, but non necessarily accepting.

Lemma 9 (Pumping). *Let us consider* $\mathcal{B}(\mathcal{M}, \mathcal{A}') = 2^{2\times|Q_{\mathcal{A}'}|\times|\Sigma_{\mathcal{A}'}|\times\max\{|\mathcal{M}|,|Q_{\mathcal{A}'}|\}^{d_{\max}}\times 2^{|Q_{\mathcal{A}'}|}}$. *If the height of* t_m *for some* $m \in \mathcal{M}$ *is greater than* $M = (\mathcal{B}(\mathcal{M}, \mathcal{A}')|Q_{\mathcal{A}'}| + 2)|Q_{\mathcal{A}_k(\varphi_m)}||Q_{\mathcal{A}'}|^{\mathcal{B}(\mathcal{M},\mathcal{A}')}$ *then there exists a depth-increasing path* $\alpha \subset t_m$ *such that* ζ_α^∞ *is compatible with* ρ *and* ρ'*, and such that* $\zeta_\alpha^\infty(\rho')$ *is a Rabin-accepting run of* $\mathcal{A}'$ *on* $\zeta_\alpha^\infty(T)$.

Proof. The proof of Lemma 9 uses the following classical result:

Lemma 10. *Let* $X, X_1, ..., X_n$ *be a collection of finite sets, let* $c_k = |X_k|$ *and* $c = |X|$*. For each* $k \in [\,1...n\,]$*, let* $f_k : X \to X_k$ *be a collection of mappings. Let* $A > 0$*. Then if* $|X| > A \times \prod_k c_k$*, then there exists* A *pairwise distinct elements* $x_1, ..., x_A \in X$ *such that for any* $(k, \ell) \in [\,1...n\,] \times [\,2...A\,]$*:* $f_k(x_\ell) = f_k(x_0)$.

Proof. Let $f : X \to \prod_k X_k$ defined by $f = (f_1, ..., f_n)$. We have $|X| > A \times \prod_k c_k = A \times |\prod_k X_k|$. One shows by induction on A that there exists A pairwise distinct elements $x_1, x_2, \ldots, x_A \in X$ such that $f(x_1) = f(x_2) = \cdots = f(x_A)$. This implies the conclusion of the lemma. □

Let us now turn to the proof of Lemma 9. Let $J_m = \{j \in J \mid k_j = m\}$ where J is the index set of $\Delta'_{\mathfrak{P}(\rho),\rho'}$. This set is finite, let us denote its elements by $j_1, \ldots, j_{|J_m|}$. By Lemma 5, $|J_m| \leq \mathcal{B}(\mathcal{M}, \mathcal{A}')$.

Let us consider a path α in t_m of length greater than $(|J_m| \times |Q_{\mathcal{A}'}| + 2) \times |Q_{\mathcal{A}_k(\varphi_m)}| \times |Q_{\mathcal{A}'}|^{|J_m|}$.

We consider Lemma 10 with $n = 1 + |J_m|$, $X = \alpha$, $X_1 = Q_{\mathcal{A}_k(\varphi_m)}$ and for $k \in [\,2...n\,]$, $X_k = Q_{\mathcal{A}'}$. Let $f_1 = \rho_m|_\alpha$ and for $h \in [\,1...n\,]$, $f_h = \rho'_{j_h}|_\alpha$. Let us recall that the values of ρ_m

and of ρ'_{j_h} for $j_h \in J_m$ are indeed in $Q_{\mathcal{A}_k(\varphi_m)}$. Let $A = |J_m| \times |Q_{\mathcal{A}'}| + 2$. Then there exists a sub-sequence $x_1, ..., x_A$ of pairwise distinct nodes of α such that for any $(h, \ell) \in [\,1...n\,] \times [\,2...A\,]$: $f_h(x_\ell) = f_h(x_1)$.

For each $i \in [\,1...|J_m| \times |Q_{\mathcal{A}'}| + 1\,]$, let us consider α_i the factor of α starting at x_i and ending at x_{i+1}.

Fact 1 *For each $i \in [\,1...|J_m| \times |Q_{\mathcal{A}'}| + 1\,]$, $\zeta^\infty_{\alpha_i}$ is compatible with ρ and ρ'.*

We have that for any $\ell \in [\,2...A\,]$: $f_1(x_\ell) = f_1(x_1)$ and $f_1 = \rho_m|_\alpha$. Thus, in particular for any $i \in [\,1...|J_m| \times |Q_{\mathcal{A}'}| + 1\,]$, $\rho_m(x_i) = \rho_m(x_{i+1})$. This implies that $\zeta^\infty_{\alpha_i}$ is compatible with ρ. A argument similar applies to show that $\zeta^\infty_{\alpha_i}$ is compatible with ρ'.

Let us consider the mapping $f : \mathcal{P}(t_m) \to \mathcal{P}(J_m \times Q_{\mathcal{A}'})$ defined by

$$f(X) = \bigcup_{x \in X,\, j \in J_m} \{(j, \rho'_j(x))\}$$

For each $\ell \in [\,1...|J_m| \times |Q_{\mathcal{A}'}| + 1\,]$, let us consider the path β_ℓ starting at the root of t_m and ending at x_ℓ. Let us consider $\delta_\ell = |f(\alpha_\ell) \setminus f(\beta_\ell)|$.

Fact 2 $\exists\, \ell_0 \in [\,1...|J_m| \times |Q_{\mathcal{A}'}| + 1\,]$ *such that* $\delta_{\ell_0} = 0$.

Let us suppose that for any $\ell \in [\,1...|J_m| \times |Q_{\mathcal{A}'}| + 1\,]$, $\delta_\ell > 0$. For any $\ell \in [\,1...|J_m| \times |Q_{\mathcal{A}'}| + 1\,]$, let us pick an element $z_\ell \in f(\alpha_\ell) \setminus f(\beta_\ell)$. Let us note that for any $\ell \in [\,1...|J_m| \times |Q_{\mathcal{A}'}| + 2\,]$, we have that $\bigcup_{\ell' < \ell} \alpha_\ell \subset \beta_\ell$. Therefore, the z_ℓ's are pairwise distinct. But $|\{z_\ell\}_\ell| = |J_m| \times |Q_{\mathcal{A}'}| + 1$, and z_ℓ belongs to $J_m \times Q_{\mathcal{A}'}$ whose number of element is $|J_m| \times |Q_{\mathcal{A}'}|$. This is a contradiction.

Fact 3 $\zeta^\infty_{\alpha_{\ell_0}}(\rho')$ *is a Rabin-accepting run of $\mathcal{A}'$ on $\zeta^\infty_{\alpha_{\ell_0}}(T)$.*

Let us consider an infinite path ω' starting at the root of $\zeta^\infty_{\alpha_{\ell_0}}(T)$. There exists a *unique* infinite path ω of T such that $\pi^\infty_{\alpha_{\ell_0}}(\omega) = \omega'$.

Let us consider the sequence $i_0, i_1, \ldots, i_h \ldots$ of elements of I such that ω crosses successively the sequence $t_{i_0}, t_{i_1}, \ldots, t_{i_h}, \ldots$ of finite trees of $\mathfrak{P}(\rho)$. Let us consider the decomposition $\omega = \omega_0\omega_1 \ldots \omega_h \ldots$ such that $\omega_h = \omega \cap t_{i_h}$. Then

$$\omega' = \pi^\infty_{\alpha_{\ell_0}}(\omega_0)\pi^\infty_{\alpha_{\ell_0}}(\omega_1) \ldots \pi^\infty_{\alpha_{\ell_0}}(\omega_h) \ldots \tag{2}$$

We have also that

$$\text{For any } h\text{: } \zeta^\infty_{\alpha_{\ell_0}}(\rho')(\pi^\infty_{\alpha_{\ell_0}}(\omega_h)) = \rho'(\omega_h) \tag{3}$$

On the one hand, if $\lambda_\rho(i_h) \neq m$, then $\omega_h \cap \sigma^{-1}_{i_h}(\alpha_{\ell_0}) = \emptyset$, and thus ω_h is not modified by $\zeta^\infty_{\alpha_{\ell_0}}$. So the above property holds. On the other hand, if $\lambda_\rho(i_h) = m$, then $\sigma^{-1}_{i_h}(\alpha_{\ell_0}) \subset \omega_h$ or else $\sigma^{-1}_{i_h}(\alpha_{\ell_0}) \cap \omega_h = \emptyset$. Indeed, let us suppose that $\sigma^{-1}_{i_h}(\alpha_{\ell_0}) \cap \omega_h \neq \emptyset$ but $\sigma^{-1}_{i_h}(\alpha_{\ell_0}) \not\subseteq \omega_h$. Then α_{ℓ_0} has a strict prefix α'_{ℓ_0} such that $\sigma^{-1}_{i_h}(\alpha_{\ell_0}) \cap \omega_h = \alpha'_{\ell_0}$. Let x be the last node of α'_{ℓ_0}. Then ω_h contains x and then goes across a son x' of x which does not belong to α_{ℓ_0}. Let $\omega_{x'}$ denote the end of ω from x'. $\omega_{x'}$ belongs to $T_{x'}$. But $T_{x'}$ is deleted by $\zeta^\infty_{\alpha_{\ell_0}}$, and so is $\omega_{x'}$. This is a contradiction because the image of ω by

$\zeta^{\infty}_{\alpha_{\ell_0}}$ which is ω' is infinite. Finally, we indeed have that $\sigma^{-1}_{i_h}(\alpha_{\ell_0}) \subset \omega_h$ or else $\sigma^{-1}_{i_h}(\alpha_{\ell_0}) \cap \omega_h = \emptyset$. If $\sigma^{-1}_{i_h}(\alpha_{\ell_0}) \cap \omega_h = \emptyset$, then the property holds as above. So let us assume that $\sigma^{-1}_{i_h}(\alpha_{\ell_0}) \subset \omega_h$. This implies that $\zeta^{\infty}_{\alpha_{\ell_0}}(\rho')(\pi^{\infty}_{\alpha_{\ell_0}}(\omega_h)) \subset \rho'(\omega_h)$. Let us suppose that there exists $x \in \omega_h$ such that $\rho'(x) \notin \zeta^{\infty}_{\alpha_{\ell_0}}(\rho')(\pi^{\infty}_{\alpha_{\ell_0}}(\omega_h))$. This implies that $x \in \sigma^{-1}_{i_h}(\alpha_{\ell_0})$. Let $\bar{x} = \sigma_{i_h}(x)$, $\bar{x} \in \alpha_{\ell_0}$, and $\rho'(x) = \rho'_{\lambda_{\mathfrak{P}(\rho),\rho'}(i_h)}(\bar{x})$. Let $k_h = \lambda_{\mathfrak{P}(\rho),\rho'}(i_h)$. The pair $(k_h, \rho'_{k_h}(\bar{x}))$ belongs to $f(\alpha_{\ell_0})$. Since $\delta_{\ell_0} = 0$, we have that $f(\alpha_{\ell_0}) \in f(\beta_{\ell_0})$. Therefore, $(k_h, \rho'_{k_h}(\bar{x})) \in f(\beta_{\ell_0})$. But β_{ℓ_0} is totally included in $\omega_h \setminus \alpha_{\ell_0}$. Therefore, $(k_h, \rho'_{k_h}(\bar{x})) \in f(\omega_h \setminus \alpha_{\ell_0})$. By definition of f, this means that there exists $\bar{x}' \in \omega_h \setminus \alpha_{\ell_0}$ such that $\rho'_{k_h}(\bar{x}') = \rho'_{k_h}(\bar{x})$, and thus, such that $\rho'_{k_h}(\bar{x}') = \rho'(x)$. Since $\bar{x}' \notin \alpha_{\ell_0}$, $\sigma^{-1}_{i_h}(\bar{x}')$ is not deleted by $\zeta^{\infty}_{\alpha_{\ell_0}}(\rho')$. And thus $\rho'(x) = \rho'_{k_h}(\bar{x}) = \rho'(\sigma^{-1}_{i_h}(\bar{x}')) = \zeta^{\infty}_{\alpha_{\ell_0}}(\rho')(\pi^{\infty}_{\alpha_{\ell_0}}(\sigma^{-1}_{i_h}(\bar{x}')) \in \zeta^{\infty}_{\alpha_{\ell_0}}(\rho')(\pi^{\infty}_{\alpha_{\ell_0}}(\omega_h))$, this is a contradiction. Finally we have indeed $\zeta^{\infty}_{\alpha_{\ell_0}}(\rho')(\pi^{\infty}_{\alpha_{\ell_0}}(\omega_h)) = \rho'(\omega_h)$.
From (2) and (3), we get $\mathit{In}(\zeta^{\infty}_{\alpha_{i_0}}(\rho')(w')) = \mathit{In}(\rho'(w))$. Seeing that ρ' is an accepting run, it satisfies the Rabin condition of $\mathcal{A}'$ on w, then so does $\zeta^{\infty}_{\alpha_{i_0}}(\rho')$ on w'. This ends the proof of the fact, and thus of the expected result. □

Theorem 1 (Decidability). *The bounded tree-width compositionality problem (Problem 3) is decidable.*

Proof. Let $\mathcal{M}$ be a finite set of function name. Let us be given with a graph grammar specification $\mathfrak{S} = (m_0, \{\varphi_m\}_{m \in \mathcal{M}})$, an integer k, and a formula Ψ. The problem consists in deciding whether there exists $G \in \mathcal{R}g_k(\mathfrak{S})$ such that $G \models \Psi$ or not.

On the one hand, let $\mathcal{A}_k(\mathfrak{S})$ be as constructed in Section 2.2. By Lemma 3, a graph G belongs to $\mathcal{R}g_k(\mathfrak{S})$ if and only if it has a syntactic expression tree $T \in \mathfrak{L}^{\infty}(\mathcal{A}_k(\mathfrak{S}))$. On the other hand, let $\mathcal{A}'$ be the Rabin automaton associated to Ψ, i.e., such that $\mathfrak{L}(\mathcal{A}') = \{T \in \mathfrak{T}^{\infty}_{L,k} \mid \overline{T} \models \Psi\}$. The problem is to decide whether $\mathfrak{L}^{\infty}(\mathcal{A}_k(\mathfrak{S})) \cap \mathfrak{L}(\mathcal{A}') = \emptyset$ or not.

Let us suppose that there exists $T \in \mathfrak{L}^{\infty}(\mathcal{A}_k(\mathfrak{S})) \cap \mathfrak{L}(\mathcal{A}')$. Let us consider the family $\{t_m\}_{m \in \mathcal{M}}$ as constructed in Section 2.2. By applying Lemma 9 inductively, we can reduce the t_m's and obtain a new family $\{t'_m\}_{m \in \mathcal{M}}$ such that each t'_m is of height less than M and that the infinite tree T' constructed from the t'_m's as in Remark 7 is still in $\mathfrak{L}^{\infty}(\mathcal{A}_k(\mathfrak{S})) \cap \mathfrak{L}(\mathcal{A}')$. Therefore, $\mathfrak{L}^{\infty}(\mathcal{A}_k(\mathfrak{S})) \cap \mathfrak{L}(\mathcal{A}') \neq \emptyset$ if and only if there exists a tree $T' \in \mathfrak{L}^{\infty}(\mathcal{A}_k(\mathfrak{S})) \cap \mathfrak{L}(\mathcal{A}')$ such that the associated family $\{t'_m\}_{m \in \mathcal{M}}$ is such that each t'_m is of height less than M.

Let us be given with a family $\{t_m\}_{m \in \mathcal{M}}$ of finite trees. Let us consider the infinite tree T obtained from the t_m's as in Remark 7.
• One can decide in an effective way whether $T \in \mathfrak{L}^{\infty}(\mathcal{A}_k(\mathfrak{S}))$ or not. To do that, one looks at each tuple of partial runs $\{\rho_m : t_m \to Q_{\mathcal{A}_k(\mathfrak{S})}\}_{m \in \mathcal{M}}$ and checks whether or not it can be used to construct a full run of $\mathcal{A}_k(\mathfrak{S})$.
• One also can decide in an effective way whether $T \in \mathfrak{L}(\mathcal{A}')$ or not. T is actually a regular tree; and the recognisability by Rabin automaton is decidable for regular trees (see [21]).

Finally, we enumerate all the families $\{t_m\}_{m \in \mathcal{M}}$ of finite trees of height less than M; and check for each of them if it give rise to an element of $\mathfrak{L}^{\infty}(\mathcal{A}_k(\mathfrak{S})) \cap \mathfrak{L}(\mathcal{A}')$; if no family satisfies this condition, we can conclude that $\mathfrak{L}^{\infty}(\mathcal{A}_k(\mathfrak{S})) \cap \mathfrak{L}(\mathcal{A}') = \emptyset$ This can be done in a finite time; this ends the proof. □

References

1. M. Abadi and L. Lamport. Composing Specifications. *ACM Transactions on Prog. Lang. and Systems (TOPLAS)*, 15(1):73–132, 1993.
2. H. R. Andersen, C. Stirling, and G. Winskel. A compositional proof system for the modal mu-calculus. In *9th Symp. on Logic in Comp. Sci. (LICS'94)*, pages 144–153. IEEE Comp. Soc. Press, 1994.
3. G. Barthe, P. Courtieu, G. Dufay, M. Huisman, S. Mello de Sousa, G. Chugunov, L.-A. Fredlund, and D. Gurov. Temporal Logic and Toolset for Applet Verification: Compositional Reasoning, Model Checking, Abstract Interpretation. Technical report, VERIFICARD Project, `http://www.verificard.org/`, Sept 2002. Deliverable 4.1.
4. G. Barthe, D. Gurov, and M. Huisman. Compositional Verification of Secure Applet Interactions. In *Fundamental Approaches to Soft. Eng. (FASE'02)*, volume LNCS 2306, pages 15–32, 2002.
5. D. Caucal. On infinite transition graphs having decidable monadic theory. In *ICALP'96 - LNCS*, volume 1099, pages 194–205, 1996.
6. C. Colby, P. Lee, and G. C. Necula. A proof-carrying code architecture for java. In *Computer Aided Verification (CAV'00) LNCS*, volume 1855, pages 557–560, 2000.
7. H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree Automata Techniques and Applications. Technical report, LIFL – France, 2003. `http://www.grappa.univ-lille3.fr/tata/`.
8. B. Courcelle. The Monadic Second-order Logic of Graphs II : Infinite Graphs of Bounded Width. *Math. Syst. Theory*, 21:187–221, 1989.
9. B. Courcelle. Graph Rewriting: An algebraic and Logic Approach. In *Handbook of Theoretical Computer Science vol. B, Van Leeuwen J. ed.* Elsevier Science Publishers, 1990.
10. M. Dam and D. Gurov. Compositional Verification of CCS processes. In *Proceedings of PSI'99*, volume LNCS 1755, pages 247–256, 1999.
11. F. Drewes, H.-J. Kreowski, and Habel A. Hyperedge Replacement Graph Grammars. In G. Rozenberg, editor, *Handbook of Graph Grammars and Computing by Graph Transformation*, volume 1, pages 95–162. World Scientific, 1997.
12. O. Grumberg and D. Long. Model Checking and Modular Verification. *ACM Trans. on Prog. Lang. & Syst.*, 16(3):843–871, 1994.
13. Y. Gurevich. Monadic Second-Order Theories. In J. Barwise and S. Feferman, editors, *Model Theoretic Logic*, pages 479–506. Springer, 1985.
14. J. Gustedt, O.A. Maehle, and J.A. Telle. The treewidth of java programs. In *Proc. of the 4th Workshop on Algorithm Engineering and Experiments (ALENEX'02), San Francisco*, 2002. To appear in LNCS.
15. T. Jensen, D. Le Métayer, and T. Thorn. Verifying Security Properties of Control-Flow Graphs. In *Proc. of the 20th Symposium on Security and Privacy, Berkeley*, pages 89–103. IEEE Computer Society Press, 1999.
16. Clarke E. M., D. E. Long, and K. L. McMillan. Compositional Model Checking. In *Proc. of the 4th Symp. on Logic in Comp. Sci. (LICS'89*, pages 353–362, 1989.
17. D. E. Muller and P. E. Schupp. The theory of ends, pushdown automata, and second-order logic. *Theoretical Comp. Sci.*, 37:51–75, 1985.
18. Kedar S. Namjoshi and Richard J. Trefler. On the completeness of compositional reasoning. In *Proc. of the 12th Int. Conference on Computer Aided Verification (CAV'00)*, number 1855, pages 139–153. Springer-Verlag, 2000.
19. M. O. Rabin. Decidability of Second-Order Theories and Automata on Infinite Trees. *Trans. of the A.M.S.*, 141:1–35, 1969.
20. N. Robertson and P. Seymour. Some New Results on the Well-Quasi Ordering of Graphs. *Annals of Discrete Math.*, 23:343–354, 1984.
21. G. Rozenberg. *Handbook of Graph Grammars and Computing by Graph Transformation*, volume 1. World Scientific, 1997.
22. C. Sprenger, D. Gurov, and M. Huisman. Simulation Logic, Applets and Compositional Verification. Technical Report 4890, INRIA, 2003.
23. C. Stirling. A complete compositional modal proof system for a subset of CCS. *LNCS*, 194:475–486, 1985.
24. W. Thomas. Languages, automata, and logic. In *Rozenberg and Salomaa ed., Handbook of Formal Languages. Springer Verlag*, volume 3, pages 389–455, 1997.
25. M. Thorup. All structured programs have small tree-width and good register allocation. *Inf. and Comp.*, 142(2):159–181, 1998.

3 Notes

1. Compositional verification have been studied in relation with the security model of open multi-application smart cards, actually Java Card based smart cards (cf. [4, 22]). After issuance, such a device can download applications coming from possibly untrusted providers. It must then verify that such an application does not corrupt its security. A compositionality result is used to limit the verification process to some local properties of the application to be downloaded, and doing this, ensure however the validity of some global security properties of the card, whose other components are supposed to have been already checked. This allows the card to check only the new component without checking again all the system, which would be actually impossible regarding the computing resources of a smart card. This is particularly adapted to the security model based on the concept of *proof carrying code* (cf. [6]). This last one advocates that the application provider provides with its application a proof that this last one does not corrupt the global security of the card. The proof is downloaded together with the code of the application, and it is checked by the card itself during downloading.

2. We kept the formalism used in [4] to encode control-flow graphs and transition systems. However these concepts can be easily defined as relational structures as defined Section 1.

3. Regular hypergraphs were initially called *equational hypergraphs* in the original paper [8]; the term "regular" appeared in [5] in order to refer to the fact that the concept of regular graph is a natural generalisation of the concept of *regular tree* (see *e.g.* [8]): the regular trees are exactly the regular graphs which are trees.

4. Seeing that the grammars that we consider are deterministic, i.e., it associates to each non-terminal symbol one and only one hypergraph, a production of form $\ell \to H_\ell$ is just denoted by H_ℓ.

6. We state the compositionality problem in an existential form for convenience. In our framework, this is equivalent to the classic statement seeing that the monadic second-order logic is stable by negation of formulæ.

7. However, some infinite hypergraphs such as the infinite grid have no syntactic expression in any $\mathfrak{T}^{\infty}_{L,k}$ (see [8].)

8. A tree provided with a deterministic tilling is not necessarily deterministic it-self. Indeed, several patterns could occur inside a particular tile.

5. Applying results of [8], this proves in particular that the satisfiability of monadic second-order properties is decidable for such transition systems.