When Is Reachability Intrinsically Decidable?

Barbara F. Csima[1] [⋆] and Bakhadyr Khoussainov[2] [⋆⋆]

[1] Department of Pure Mathematics, University of Waterloo
csima@math.uwaterloo.ca
[2] Department of Computer Science, University of Auckland
bmk@cs.auckland.ac.nz

Abstract. A graph $\mathcal{H}$ is **computable** if there is a graph $\mathcal{G} = (V, E)$ isomorphic to $\mathcal{H}$ where the set V of vertices and the edge relation E are both computable. In this case $\mathcal{G}$ is called a **computable copy** of $\mathcal{H}$. The **reachability problem** for $\mathcal{H}$ in $\mathcal{G}$ is, given $u, w \in V$, to decide whether there is a path from u to w. If the reachability problem for $\mathcal{H}$ is decidable in *all* computable copies of $\mathcal{H}$ then the problem is *intrinsically decidable.* This paper provides syntactic-logical characterizations of certain classes of graphs with intrinsically decidable reachability relations.

1 Introduction

The study of reachability problems has played a central role in many areas of computer science and applications. In the context of finite graphs the problem is reduced to computing the components of the graphs. Tarjan's algorithm solves the reachability problem for finite graphs in linear time [19]. In complexity theory reachability for finite graphs has been important in the investigation of subclasses of P (e.g. see [4] and its reference list). The problem plays a valuable role in model checking and verification since model checking tasks are often reduced to reachability problems [3] [2]. There is also a large interest in reachability problems in different types of computational models such as in counter automata, timed automata, pushdown automata, Petri-nets, rewriting systems, hybrid systems, systems with unbounded number of clocks, protocols, communication systems [5] [8] [9]. These all give rise to infinite graphs and many natural tasks for these graphs involve computing the reachability relation. See also [7] [14] [17] and their references. The paper [1] defines a methodology for reachability analysis of infinite-state systems based on the theory of well quasi-orderings.

Suppose that we are given a finite presentation of an infinite graph. For example, the graph can be the space of states of a system with an unbounded number of protocols or the configuration space of a Turing machine. The configuration space of a machine is the graph whose vertices are the configurations of the machine and an edge is put between configurations x and y if there is an instantaneous move of the machine from x to y. In each of these cases the graphs

[⋆] Partially supported by Canadian NSERC Discovery Grant 312501.
[⋆⋆] Partially supported by Marsden Fund of Royal New Zealand Society

have finite presentations. For example, if T is a Turing machine then T is a finite presentation of its own configuration space. It is clear that the reachability problem for such graphs is computably enumerable (that is, recognizable by Turing machines). Due to the undecidibility of the Halting problem, the reachability problem for graphs that have finite presentations is not always decidable.

The aim of this paper is to study the reachability problem for infinite graphs, and find syntactic conditions under which the reachability problem is always decidable *independent* of finite presentations. We point out that some finite presentations of graphs automatically imply decidability of the reachability problem. For example, reachability is decidable for the configuration spaces of pushdown automata [7]. Similarly, reachability is decidable for the graphs that have certain types of monadic second order interpretations in the binary tree (see for example [10] [11]). In this paper we seek conditions which entail decidability of the reachability problem in a given graph *independent* of its finite presentations. Clearly, these conditions should be intrinsic to the graphs rather than their presentations. We now set up the problem formally.

Let $\mathcal{H}$ be an infinite graph. We always assume that our graphs are undirected. We define finite presentations of graphs via Turing machines as follows:

Definition 1. *The graph $\mathcal{H}$ is* **computable** *if there exists a graph $\mathcal{G} = (V, E)$ isomorphic to $\mathcal{H}$ and there are Turing machines T_V and T_E over an alphabet Σ such that the following two properties hold: (1) The machine T_V halts on every input string, and for all $u \in \Sigma^\star$, T_V accepts u if and only if $u \in V$. (2) The machine T_E halts on every pair of input strings, and for all $u, w \in \Sigma^\star$, T_E accepts (u, w) if and only if $(u, w) \in E$. In this case the pair $P = (T_V, T_E)$ is a* **presentation** *of $\mathcal{H}$ and $\mathcal{G}$ is the* **computable copy** *of $\mathcal{H}$ given by P.*

We often abuse notation and identify the presentation P of $\mathcal{H}$ with the computable copy $\mathcal{G}$ given by P.

The reachability relation in $\mathcal{H}$ is the set $\{(x, y) \mid \text{there is a path in } \mathcal{H} \text{ from } x \text{ to } y\}$. In order to give an algorithmic spin to the reachability relation one needs to employ presentations of the graph $\mathcal{H}$:

Definition 2. *The* **reachability problem** *for $\mathcal{H}$ in presentation P (equivalently in $\mathcal{G}$) is, given $u, w \in \Sigma^\star$, to decide if there exists a path from u to w.*

The reachability problem in computable graphs $\mathcal{G}$ is computably enumerable (that is recognized by Turing machines). Indeed, given u and w one systematically searches through all paths starting with u. When a path from u to w is detected the search is terminated. We give several examples.

Example 1. In the graph $\mathcal{G} = (V, E)$ with $V = \{0, 1\}^\star$ and $E = \{(u, w) \mid |u| = |v|\}$, where $|x|$ refers to the length of x, the reachability problem is decidable.

Example 2. Let $\mathcal{G}$ be a computable graph such that each component of $\mathcal{G}$ embeds exactly one cycle. The reachability problem in $\mathcal{G}$ is decidable. Indeed, on inputs u and w search for cycles embedded into the components of u and w. If the cycles are distinct, then reject the pair (u, w); otherwise, accept.

Example 3. Consider the graph $\mathcal{G} = (Z, E)$, where Z is the set of all integers, and $E = \{(3i, 3i+1) \mid i \text{ is not negative}\} \cup \{(3i+1, 3i+2) \mid i \text{ is not negative}\}$. The reachability problem in $\mathcal{G}$ is decidable.

Example 4. This example comes from the verification community. A **parameterized system** S is an infinite sequence M_1, M_2, $\ldots$ of finite state machines where for each i, we effectively know the number of states in M_i and all the transitions in M_i. Given a specification ϕ (written in a logic such as LTL), the verification of ϕ on S consists of checking whether or not the state space of S satisfies ϕ for all n. The state space of S consists of tuples $(s_1, \ldots, s_n, n)$, where each s_i is a state of M_i and $n \geq 1$. Let $\mathcal{G}$ have an edge between $(s_1, \ldots, s_n, n)$ and $(q_1, \ldots, q_m, m)$ iff $n = m$ and there is a transition from s_i to q_i for $i = 1, \ldots, n$. Then $\mathcal{G}$ is locally finite and the reachability problem in $\mathcal{G}$ is decidable.

We stress that decidability of the reachability problem for $\mathcal{H}$ in one presentation does *not* imply reachability is decidable in *all* presentations. In this sense reachability is not absolute in terms of decidability. Here is our central definition:

Definition 3. *If the reachability problem for $\mathcal{H}$ is decidable in all presentations of $\mathcal{H}$ then we say that the problem is* **intrinsically decidable**.

For the graphs in Examples 1 and 2, reachability is intrinsically decidable. For the graphs in Example 3, reachability is not intrinsically decidable (Lemma 2 (1); note that any two singleton components arising from negative integers can be disjointly embedded into any 3-vertex component arising from a non-negative integer). In Example 4, intrinsic decidability of reachability depends on the system S.

We note that intrinsic decidability (of the reachability relation) has some similarities with finding lower bounds for regular model checking problems. In this problem the configurations of a transition system are coded by strings in such a way that the edge relation under this coding is recognized by automata. Such a coding is called a regular coding (and hence makes the transition system an automatic graph). Under the coding one then checks if a given specification is satisfied by the transition system. Finding lower bounds for this problem requires investigating *all* regular codings.

Our goal is to provide characterizations of certain classes of graphs with intrinsically decidable reachability relations. We work over a language with a single binary relation symbol, E. Our characterization involves an infinitary logic. Recall that the logic $L_{\infty\omega}$ is an extension of first order logic with infinitary $\bigvee$ and $\bigwedge$ connectives (for a treatment see [16]). We use an effective fragment of this logic, $L_{c\omega}$, defined by Ash and Nerode in [6], as follows.

Definition 4. *We say that a relation $R(\bar{x})$ in graph $\mathcal{H}$ is* **existentially definable** *if there exists a tuple $\bar{a}$ in $\mathcal{H}$ and a computable sequence of existential first order logic formulas $\psi_0(\bar{x}, \bar{a}), \psi_1(\bar{x}, \bar{a}), \ldots$ such that for all $\bar{v}$ in $\mathcal{H}$ we have*

$$\mathcal{H} \models R(\bar{v}) \iff \psi_0(\bar{v}, \bar{a}) \vee \psi_1(\bar{v}, \bar{a}) \vee \psi_2(\bar{v}, \bar{a}) \vee \ldots.$$

The reachability relation in any graph is existentially definable. The desired formula is the disjunction of formulas $d_i(x,y)$, where $d_i(x,y)$ states that the distance from x to y is at most i. That is, $d_i(x,y) = \exists x_1 ... \exists x_i (\bigwedge_{p \neq q} \neg x_p \equiv x_q) \wedge (\bigwedge_{1 \leq p < i} E x_p x_{p+1})$. The following is an obvious lemma:

Lemma 1. *If relation R of a graph $\mathcal{H}$ is existentially definable then in all presentations of $\mathcal{H}$ the relation R is computably enumerable.*

Corollary 1. *If the complement of the reachability relation of a graph $\mathcal{H}$ is existentially definable then the reachability problem for $\mathcal{H}$ is intrinsically decidable.*

Proof. Let $\mathcal{G}$ be a computable copy of $\mathcal{H}$. By Lemma 1 the complement of reachability in $\mathcal{G}$ is c.e. We also know that the reachability relation is c.e. So, the reachability problem in $\mathcal{G}$ is decidable. Hence it is intrinsically decidable. $\square$

We investigate the converse of the corollary above. We provide classes of graphs in which the complement of the reachability relation is existentially definable. These results provide logical characterizations of intrinsically decidable reachability relations.

Notations and conventions. In the rest of the paper we always assume that our graphs are infinite (save for finite subgraphs of the infinite graphs we are considering). Also, we assume that the set of vertices of computable graphs coincides with the set ω of natural numbers. We note that we may construct an infinite computable graph $\mathcal{G}$ by stages, by at each stage s of an effective construction defining a finite graph $\mathcal{G}_s$ with domain an initial segment of ω, such that $\mathcal{G}_s \subset \mathcal{G}_{s+1}$, and letting $\mathcal{G} = \cup_s \mathcal{G}_s$. The graph $\mathcal{G}$ will have domain ω, and it will be computable since we will know by stage $s = \max v, w$ whether or not there is an edge between the vertices v and w.

A partial computable function $\Phi : \omega^n \to \omega$ is one that can be computed by a Turing machine. We can systematically list all Turing machines $T_0, T_1, \ldots, T_e, \ldots$, and thus give rise to an effective listing of all partial computable functions $\Phi_0, \Phi_1, \ldots, \Phi_e, \ldots$. Finally, $\Phi_{e,s}$ denotes the following partial function. The domain of $\Phi_{e,s}$ consists of all $i \leq s$ such that $\Phi_e(i)$ is defined, and the value $\Phi_e(i)$ is obtained within s steps of the computation of the Turing machine T_e on input i. Also, $\Phi_{e,s+1}(i) \downarrow$ means that the value of $\Phi_{e,s+1}$ on input i is defined.

We let $X^{[2]} = \{\{a,b\} \mid a, b \in X \wedge a \neq b\}$, the set of unordered pairs from X. For other basic notations of computability theory the reader is referred to [18].

Our main proofs involve methods from computability theory (see for example [18]) and computable model theory (see [12] [13]). Our proofs are of two types. The first type of proofs (Lemma 2 and Proposition 1) are finite injury type of construction common in computable model theory and computability. The second type of proofs (Theorems 2 and 3) are based on more complicated constructions known as $\emptyset''$ (the second jump of the computable degree) priority tree constructions (see [18]).

2 Graphs with Computable Size Functions

In this section we consider graphs *all* of whose components are finite. We call these graphs **strongly locally finite**. Our goal is to characterize certain strongly locally finite graphs with intrinsically decidable relations.

Let $\mathcal{H}$ be a strongly locally finite graph. The component of a vertex v of $\mathcal{H}$ is denoted by $C(v)$. The **size function** $\mathrm{size}_{\mathcal{H}}$ of $\mathcal{H}$ gives for each vertex v the cardinality of $C(v)$. Thus, $\mathrm{size}_{\mathcal{H}}(v) = |C(v)|$ for all vertices v of $\mathcal{H}$.

Let $\mathcal{G} = (\omega, E)$ be a computable copy of a strongly locally finite graph $\mathcal{H}$. In this section we consider those computable graphs $\mathcal{G}$ whose size functions $\mathrm{size}_{\mathcal{G}}$ are computable. Clearly, the function $\mathrm{size}_{\mathcal{G}}$ is computable if and only if for each vertex $v \in \omega$ one can effectively compute the number of edges from v. For such a $\mathcal{G}$, we effectively list all connected components of $\mathcal{G}$ as $C_0, C_1, C_2, \ldots$. We fix this listing and use it for the next definitions and results.

Definition 5. *We say that components C_i and C_j* **disjointly embed** *into C_k if their disjoint union as a graph can be embedded into C_k. We also define the function $h : \omega^{[2]} \to \omega$ by $h(i,j) = |\{k : C_i \text{ and } C_j \text{ disjointly embed into } C_k\}|$. We call h the* **disjoint embedding function** *for the presentation $\mathcal{G}$.*

This definition implies that if C_i and C_j disjointly embed into C_k then no edge exists between the images of C_i and C_j in C_k under the embedding.

Lemma 2. *Let $\mathcal{G}$ be a computable copy of $\mathcal{H}$. Under the assumptions above about the graph $\mathcal{G}$ we have the following properties:*

1. *If for all n, there exist $i, j > n$, $i \neq j$, such that $h(i,j) = \infty$, then the reachability problem for $\mathcal{H}$ is not intrinsically decidable.*
2. *If there exists an $n \in \omega$ such that $h(i,j)$ is finite for all $i, j > n$, $i \neq j$, and h is computable, then the reachability problem for $\mathcal{H}$ is intrinsically decidable.*
3. *If there exists an $n \in \omega$ such that $h(i,j)$ is finite for all $i, j > n$, $i \neq j$, and h is not computable, then the reachability for $\mathcal{H}$ is not intrinsically decidable.*

Proof (1). To show that the reachability relation on $\mathcal{G}$ is not intrinsically decidable, we need to exhibit a computable graph $\mathcal{G}' = (\omega, E')$ isomorphic to $\mathcal{G}$ such that the reachability relation is not decidable on $\mathcal{G}'$. For that the graph $\mathcal{G}'$ must satisfy the following requirements:

$$P_e : \ \Phi_e \text{ does not decide the reachability relation on } \mathcal{G}'$$

where $\Phi_0, \Phi_1, \ldots$ is an effective list of all partial computable functions from ω^2 to $\{0,1\}$. We consider $\Phi_{e,s+1}(v,w) = 0$ to mean that Φ_e tells us that v and w are not in the same component, and $\Phi_{e,s+1}(v,w) = 1$ to mean that Φ_e tells us that v and w are in the same component.

The requirement P_e has a higher **priority** than P_t if $t > e$. We construct $\mathcal{G}'$ by stages. At stage s we construct a finite graph $\mathcal{G}'_s$ so that $\mathcal{G}'_s$ is isomorphic to $\mathcal{G}$ restricted to $C_0 \cup \ldots \cup C_{s-1}$, $\mathcal{G}'_s \subset \mathcal{G}'_{s+1}$ for all s, and f_s is the isomorphism constructed at stage s. Our desired graph will be $\mathcal{G}' = \cup_s \mathcal{G}'_s$.

At stage 0, set $\mathcal{G}'_0$ to be the empty graph. Set f_0 to be undefined. Say that all components C_i are free for all requirements P_e.

At stage $s+1$, consider $\mathcal{G}_s$ obtained by adding C_s to $\mathcal{G}_{s-1}$. Let $C'_0, \ldots, C'_{s-1}$ be all components in $\mathcal{G}'_{s-1}$ with each C'_i is isomorphic to C_i via f_s for $i < s$. Find minimal $e \leq s+1$ such that for some $\langle i, j\rangle < s$ with $i \neq j$ we have:

1. P_e requires attention and $\Phi_{e,s+1}(v,w)\downarrow$ for some $v \in C'_i$ and $w \in C'_j$
2. C_i and C_j disjointly embed into C_s or $\Phi_{e,s+1}(v,w) \neq 0$.
3. The components C_i and C_j are free for P_e.

If such e does not exist then go to stage $s+2$. If $\Phi_{e,s+1}(v,w) \neq 0$, declare P_e does not require attention, and declare C_i and C_j not free for all P_t with $t > e$. Otherwise, act as follows: (1) Extend C'_i and C'_j to a single component, denoted by C'_s, such that $C'_s \cong C_s$; (2) Build new copies C'_i and C'_j isomorphic to C_i and C_j; (3) Redefine f_s by mapping C_i to C'_i, C_j to C'_j and C_s to C'_s. Declare C_i, C_j, C_s not free for P_t with $t > e$, declare P_t requires attention for $t > e$, and declare P_e does not require attention. This completes the construction for $\mathcal{G}'_{s+1}$.

The correctness of the construction is now a standard proof. The proof is based on the following two observations. First of all, one inductively shows that each requirement P_e is satisfied. Secondly, one proves that the function $f(v) = \lim_s f_s(v)$ establishes an isomorphism between $\mathcal{G}$ and $\mathcal{G}'$ constructed. $\square$

Proof (2). Suppose $h(i,j)$ is finite for all pairs $i, j > n$ for some fixed n, and that h is computable. We show that the reachability relation for $\mathcal{H}$ is intrinsically decidable. Note that we can view $\mathcal{G}$ as an effective disjoint union of finite graphs $D_0, ..., D_n, C_0, C_1, C_2, ...$, where $h(i,j)$ is finite for all (i,j) corresponding to C_i, C_j. Let $\mathcal{G}'$ be another computable presentation of $\mathcal{G}$. We want to decide the reachability problem on $\mathcal{G}'$. Since $\mathcal{G}' \cong \mathcal{G}$, we may assume that we are given $D'_0 \cong D_0, ..., D'_n \cong D_n$, that is, we can compute membership in the D'_i. Suppose v and w are vertices of $\mathcal{G}'$. Since $\mathcal{G}'$ is computable, we can approximate each component $C(x)$ of vertex x by stages $C_0(x) \subseteq C_1(x) \subseteq \ldots$ so that $C(x) = \cup_s C_s(x)$. We can decide whether w is reachable from v using the following algorithm:

1. If $v \in D'_i$ for some $1 \leq i \leq n$, then $R(v,w) \iff w \in D'_i$.
2. If at any stage s, a path is found from v to w, then declare v, w connected.
3. Find stage s_1 at which $C_{s_1}(v) = C_i$ and $C_{s_1}(w) = C_j$ for some distinct i, j.
4. Find $C_{k_1}, ..., C_{k_{h(i,j)}}$ all the components of $\mathcal{G}$ into which C_i and C_j disjointly embed (These can be found since the size function for $\mathcal{G}$ is computable) .
5. Find the first stage $s_2 \geq s_1$ such that $\mathcal{G}'$ provides components $C'_{k_1}, ..., C'_{k_{h(i,j)}}$ isomorphic to $C_{k_1}, ..., C_{k_{h(i,j)}}$, and v and w belong to these components.
6. If v and w are in the same component C'_l for some $l \in \{k_1, \ldots, k_{h(i,j)}\}$ then declare v, w connected; otherwise, declare v, w are not connected.

By stage s_2 when the components $C'_{k_1}, ..., C'_{k_{h(i,j)}}$ are found the components $C_{s_2}(v)$ and $C_{s_2}(w)$ may now properly extend the old approximations $C_{s_1}(v)$ and $C_{s_1}(w)$. For example, it may be that $C_{s_2}(v) = C_{s_2}(w)$ in which case by item (2) v and w are declared connected. It is not too hard to see that the algorithm provided decides the reachability problem for the graph $\mathcal{G}'$. $\square$

Proof (3). We need to build a computable copy $\mathcal{G}' \cong \mathcal{G}$ such that the reachability relation is not decidable on $\mathcal{G}'$. We use the same construction as in the proof of part (1) of this theorem. The only difference is that we modify the list $C_0, C_1, \ldots$ to contain only those components of $\mathcal{G}$ for which $h(i,j)$ is finite.

Suppose P_e is the requirement with the highest priority that is not satisfied. Let s be the stage when all requirements with higher priorities are satisfied. Since Φ_e is the characteristic function of the reachability relation, we can compute the the function h as follows. Consider (i,j) such that C_i, C_j are free for P_e. Note that there are only finitely many C_i that are not free for P_e. Let t be the stage $> s$ such that $\Phi_{e,t}(v,w)$ is defined for some $v \in C'_i[t]$ and $w \in C'_j[t]$. Such a stage must exist since Φ_e is total, and since by the construction each C'_i is shifted at most finitely often. We must have $\Phi_{e,t}(v,w) = 0$, as otherwise P_e would have been satisfied at stage t. From this stage on C_i and C_j cannot be disjointly embedded into C_k for all $k > t$. Hence $h(i,j)$ can be computed effectively, a contradiction. □

Corollary 2. *The reachability relation on $\mathcal{G}$ is intrinsically decidable if and only if $h(i,j)$ is computable and there is an n such that $h(i,j)$ is finite for all $i,j > n$.*

Proof. The conditions on h in parts (1), (2) and (3) cover all possibilities, and only condition (2) gives an intrinsically decidable rechability relation.

Theorem 1. *The reachability problem for $\mathcal{G}$ is intrinsically decidable if and only if both the reachability and its complement are existentially definable.*

Proof. We need only prove the direction that if the reachability problem for $\mathcal{G}$ is intrinsically decidable, then the complement of the reachability relation is existentially definable. Suppose that the reachability problem for $\mathcal{G}$ is intrinsically decidable. Then by Corollary 2, h is computable and has finite values almost everywhere. Let $\mathcal{G}$ be viewed as an effective disjoint union of finite graphs $D_0, ..., D_n, C_0, C_1, C_2, ...$, where $h(i,j)$ is finite for all (i,j) corresponding to C_i, C_j, as in the proof of Lemma 2 (2). We need to exhibit an existential $\mathcal{L}_{c\omega}$ formula $\psi(v,w,\overline{d})$ such that for any computable presentation $\mathcal{G}'$ of $\mathcal{G}$ there exists some tuple $\overline{d}'$ of vertices from $\mathcal{G}'$, such that for any vertices v', w' of $\mathcal{G}'$, we have that $\mathcal{G}' \models \neg R(v', w') \iff \psi(v', w', \overline{d}')$. We use the proof of Lemma 2 (2) to provide the formula. For any tuple $\overline{x} = (x_0, ..., x_m)$, let $\delta(\overline{x}) := \bigwedge_{p \neq q} \neg x_p = x_q$.

For $1 \leq i \leq n$, let $\varphi_i(v,w,\overline{d}_i) :=$ "$v \in D_i$" $\wedge \neg$"$w \in D_i$". Here "$v \in D_i$" abbreviates the formula $v = d_{i,1} \vee ... \vee v = d_{i,n_i}$, where $n_i = |D_i|$.

For all pairs $(i,j) \in \omega^{[2]}$, we can compute $h(i,j)$, and since $\mathcal{G}$ has computable size function, we can compute $C_{k_1}, ...C_{k_{h(i,j)}}$, the $h(i,j)$ many distinct components in $\mathcal{G}$ into which C_i and C_j disjointly embed. That is, for each $l \in \{i, j, k_1, ..., k_{h(i,j)}\}$, we can compute the vertices $\{c_{l,0}, ..., c_{l,n_l}\}$ of C_l. We let $\overline{x}_l = (x_{l,0}, ..., x_{l,n_l})$. Now for all pairs $(i,j) \in \omega^{[2]}$, define

$$\begin{aligned}\psi_{i,j}(v,w) := \exists \overline{x}_i \overline{x}_j \overline{x}_{k_0} ... \overline{x}_{k_{h(i,j)}} [\delta(\overline{x}_i, \overline{x}_j, \overline{x}_{k_0}, ..., \overline{x}_{k_{h(i,j)}}) \\ \wedge \bigvee_{1 \le p \le n_i} v = x_{i,p} \wedge \bigvee_{1 \le p \le n_j} w = x_{j,p} \wedge \\ \bigwedge_{\binom{l,m \in \{i,j,k_1,...,k_{h(i,j)}\}}{1 \le p \le n_l, 1 \le q \le n_m}} [\bigwedge_{\mathcal{G} \models E[c_{l,p}, c_{m,q}]} E x_{l,p} x_{m,q} \wedge \bigwedge_{\mathcal{G} \not\models E[c_{l,p}, c_{m,q}]} \neg E x_{l,p} x_{m,q}]].\end{aligned}$$

There are only finitely many formulas of the form φ_i, and the formulas of the form $\psi_{i,j}$ are computable in (i,j). Hence, the following formula is an effective disjunction of existential first order formulas:

$$\psi(v, w, \overline{d}_1, ..., \overline{d}_n) := \bigvee_{1 \le i \le n} \varphi_i(v, w, \overline{d}_i) \wedge \bigvee_{(i,j) \in \omega^{[2]}} \psi_{i,j}(v,w).$$

From the proof of Lemma 2 (2) it is not hard to see that the formula above existentially defines the complement of the reachability relation. □

3 Counterexample

A natural question arises whether we can remove the assumption on computability of the size function. The goal of this section is to show the assumption *cannot* be omitted by outlining the proof of the following theorem:

Theorem 2. *There exists a strongly locally finite computable graph $\mathcal{G}$ with intrinsically decidable reachability relation such that the complement of the relation is not existentially definable.*

Proof. We give a stage-wise construction of $\mathcal{G}$ on which the reachability relation is decidable. Our construction must guarantee the following two properties of $\mathcal{G}$.

First, we need to guarantee that the complement of reachability is *not* existentially definable. Let $\psi_0, \psi_1, \psi_e, ...$ be an effective listing of all infinitary effective existential formulas with finitely many parameters from $\mathcal{G}$. For each e the graph $\mathcal{G}$ must provide vertices v and w such that $\mathcal{G} \models R[v,w] \iff \mathcal{G} \models \psi_e[v,w]$. This will guarantee that the complement of reachability is not existentially definable.

The second property is that reachability in $\mathcal{G}$ must be intrinsically decidable. This is done as follows. Let $\mathcal{G}_0, \mathcal{G}_1, \mathcal{G}_2, ...$ be an effective enumeration of all computable graphs. We ensure that if $\mathcal{G}_e \cong \mathcal{G}$ then a computable isomorphism between $\mathcal{G}$ and $\mathcal{G}_e$ exists. This with the fact that reachability will be decidable on $\mathcal{G}$ will guarantee the intrinsic decidability of the reachability relation.

We will construct our graph $\mathcal{G}$ to consist of finite chains, as described below. A **cycle** of length $n > 2$, denoted as $\mathcal{C}_n$, is a graph isomorphic to $\{\{1, ..., n\}, E\}$, where $E = \{\{1,2\}, \{2,3\}, ..., \{n-1, n\}, \{n, 1\}\}$. We **link** the cycle $\mathcal{C}_n$ to the cycle $\mathcal{C}_m$ (both share no vertex) by adding a single new vertex v that has an edge to n

in $\mathcal{C}_n$ and an edge to 1 in $\mathcal{C}_m$. Call the resulting graph the **chain** $\mathcal{C}_n\mathcal{C}_m$. Similarly, chains $\mathcal{C}_{n_1}...\mathcal{C}_{n_k}$ and $\mathcal{C}_{m_1}...\mathcal{C}_{m_l}$ are linked to form the chain $\mathcal{C}_{n_1}...\mathcal{C}_{n_k}\mathcal{C}_{m_1}...\mathcal{C}_{m_l}$.

The strategy to defeat a single $\psi_e = \bigvee_{i\in\omega} \psi_{e,i}$ (in achieving the first property) is as follows. First, construct unique cycles $\mathcal{C}_{e_1}, \mathcal{C}_{e_2}$, and $\mathcal{C}_{e_3}$ in $\mathcal{G}$ that do not use any parameters mentioned by ψ_e. Choose $v \in \mathcal{C}_{e_1}$ and $w \in \mathcal{C}_{e_2}$. If at some stage we see that $\mathcal{G}_s \models \psi_{e,i}[v,w]$, then extend the cycle $\mathcal{C}_{e_3}$ to a chain $\mathcal{C}_{e_1}\mathcal{C}_{e_3}\mathcal{C}_{e_2}$. Let v' and w' be the images of v and w under the disjoint embedding of $\mathcal{C}_{e_1}$ and $\mathcal{C}_{e_2}$ into $\mathcal{C}_{e_1}\mathcal{C}_{e_3}\mathcal{C}_{e_2}$. Then since $\psi_{e,i}$ is an existential formula, we must have $\mathcal{G}_{s+1} \models \psi_{e,i}[v',w']$, though also $\mathcal{G}_{s+1} \models R[v',w']$.

The strategy to satisfy the second property for a single $\mathcal{G}_e$ (say $\mathcal{G}_0$) in the presence of one ψ_e is the following. Suppose we knew that $\mathcal{G}_0 \cong \mathcal{G}$, and we wanted to build a computable isomorphism $g_0 : \mathcal{G} \to \mathcal{G}_0$, while still working to defeat ψ_e. In this case we would again begin by constructing cycles $\mathcal{C}_{e_1}, \mathcal{C}_{e_2}$, and $\mathcal{C}_{e_3}$ in $\mathcal{G}$. However, we would wait until $\mathcal{G}_e$ provided components isomorphic to $\mathcal{C}_{e_1}, \mathcal{C}_{e_2}$, and $\mathcal{C}_{e_3}$, and define our computable isomorphism from $\mathcal{G}$ to $\mathcal{G}_0$ accordingly, before choosing $v \in \mathcal{C}_{e_1}$ and $w \in \mathcal{C}_{e_2}$ and waiting for a stage where $\mathcal{G}_s \models \psi_{e,i}[v,w]$. At that point we would proceed as before, and we would wait for $\mathcal{G}_0$ to extend its component of $\mathcal{C}_{e_3}$ to one of the form $\mathcal{C}_{e_1}\mathcal{C}_{e_3}\mathcal{C}_{e_1}$, and we would then define g_0 on this extension. Since we do not know whether $\mathcal{G}_0 \cong \mathcal{G}$, we will need six unique cycles $\mathcal{C}_{e0_1}, \mathcal{C}_{e0_2}, \mathcal{C}_{e0_3}$ and $\mathcal{C}_{e1_1}, \mathcal{C}_{e1_2}, \mathcal{C}_{e1_3}$. The cycles $\mathcal{C}_{e0_1}, \mathcal{C}_{e0_2}, \mathcal{C}_{e0_3}$ will be used to defeat ψ_e under the assumption that $\mathcal{G}_0 \not\cong \mathcal{G}$. They will not wait for $\mathcal{G}_0$ to provide isomorphic copies of them before proceeding against ψ_e. The cycles $\mathcal{C}_{e1_1}, \mathcal{C}_{e1_2}, \mathcal{C}_{e1_3}$ will work under the assumption that $\mathcal{G}_0 \cong \mathcal{G}$. Once $\mathcal{G}_0$ exhibits copies isomorphic to $\mathcal{C}_{e1_1}, \mathcal{C}_{e1_2}, \mathcal{C}_{e1_3}$, then we will use these cycles to work against ψ_e. We will know that we do not need the cycles $\mathcal{C}_{e0_1}, \mathcal{C}_{e0_2}, \mathcal{C}_{e0_3}$, and so we will link new distinct cycles to each of $\mathcal{C}_{e0_1}, \mathcal{C}_{e0_2}, \mathcal{C}_{e0_3}$ and $\mathcal{C}_{e0_1}\mathcal{C}_{e0_3}\mathcal{C}_{e0_2}$ (if present), in order to distinguish them for future definition of the isomorphism g_0.

The general construction is a standard and quite technical $\emptyset''$ priority tree construction, where to defeat ψ_e while still building the possible computable isomorphisms for $\mathcal{G}_i$ with $i < e$, we need to have cycles of the form $\mathcal{C}_{\sigma_1}, \mathcal{C}_{\sigma_2}, \mathcal{C}_{\sigma_3}$, where $\sigma \in \{0,1\}^e$. That is, the finite binary string σ codes the guess as to which graphs of higher priority are actually isomorphic to $\mathcal{G}$, and waits for those it feels are isomorphic to show isomorphic components before proceeding against ψ_e.

4 Locally finite graphs

Here we consider locally finite graphs by allowing infinite components. Recall that a graph is *locally finite* if each vertex belongs to only finitely many edges. A locally finite computable graph $\mathcal{G}$ is *highly computable* if given a vertex v of $\mathcal{G}$ one can compute the number of edges of v. The state spaces of Turing machines are examples of highly computable graphs. Clearly, the graph constructed in Theorem 2 is *not* highly computable.

It is easy to turn the graph built in Theorem 2 into a locally finite graph with intrinsically decidable reachability relation such that (1) *all* components of the graph are infinite and (2) the complement of reachability is not existentially definable. One may assume that the reason for a such phenomenon is that the graph is not highly computable. However, by slightly modifying the construction in Theorem 2, one can prove this:

Theorem 3. *There exists a highly computable graph $\mathcal{G}$ (that necessarily possesses infinite components) with intrinsically decidable reachability relation such that the complement of the relation is not existentially definable.*

To get a positive result we use Ash-Nerode theorem [6] applied to reachability relations on computable graphs. Let $\mathcal{G} = (V, E)$ be a computable graph and R be its reachability relation. Assume that the following condition, called *Ash-Nerode condition*, holds: There exists an algorithm that given a FO existential formula $\phi(x, y, \bar{a})$, where $\bar{a} \in V$, decides if $\mathcal{G} \models \forall x \forall y(\phi(x, y, \bar{a}) \rightarrow \neg R(x, y))$. This implies that R is decidable, and given FO existential formula $\phi(\bar{a})$, where $\bar{a} \in V$, we can decide if $\mathcal{G} \models \phi(\bar{a})$.

Theorem 4 (Ash-Nerode [6]). *If the computable graph $\mathcal{G} = (V, E)$ and the reachability relation R satisfy Ash-Nerode condition then R is intrinsically decidable if and only if R and $V^2 \setminus R$ are both existentially definable.* $\square$

Below we provide a sufficient condition for reachability *not* to be intrinsically decidable. Our goal is to extend Lemma 2 (1) for locally finite computable graphs.

Assume that $g(i) = \lim_{j \to \infty} f(i, j)$ exists for every i, where $f : \omega \times \omega \to \omega$ is a computable function. Intuitively, the function f approximates the value $g(i)$ by making finitely many guesses about the value of $g(i)$ and eventually makes a correct guess. We say that a set $X \subseteq \omega$ is a Δ^0_2-set if there exists a computable function $f : \omega \times \omega \to \{0, 1\}$ such that $X(i) = \lim_j f(i, j)$ for all $i \in \omega$. The function f is called an approximation to X.

Proposition 1. *Let $\mathcal{G}$ be a locally finite computable graph with decidable reachability relation. For each $s \in \omega$, let $\mathcal{G}_s$ be the restriction of the graph of $\mathcal{G}$ to $\{0, ..., s\}$. Since $\mathcal{G}$ is computable, we can compute $\mathcal{G}_s$. For each $v \in \{0, ..., s\}$, let $C_s(v)$ denote the connected component of v in $\mathcal{G}_s$. Assume that there exists an infinite Δ^0_2-set of vertices X such that (1) Any two distinct elements of X are in distinct components of $\mathcal{G}$; and (2) For all $(x, y) \in X^{[2]}$ and for all $t \geq 1$ there exist infinitely many distinct components of $\mathcal{G}$ into which $C_t(x)$ and $C_t(y)$ disjointly embed. Then the reachability relation on $\mathcal{G}$ is not intrinsically decidable.*

Proof. Since reachability on $\mathcal{G}$ is decidable, we may assume that if $C_{\max(v,w)}(v) \neq C_{\max(v,w)}(w)$, then $C_s(v) \neq C_s(w)$ for all s. We build a computable graph $\mathcal{G}' \cong \mathcal{G}$ by meeting the requirements for $e \in \omega$:

$P_e : \Phi_e$ is not the characteristic function of the reachability relation on $\mathcal{G}'$.

We will construct $\mathcal{G}'$ by stages. At each stage s we will have a function $g_s : \mathcal{G}_s \cong \mathcal{G}'_s$ and we will ensure that $g = \lim_s g_s$ exists. If we declare that $g_s(v) = v'$, then we will define g_s such that $g_s : C_s(v) \cong C_s(v')$. If at a later stage t the component of v in $\mathcal{G}$ grows ($C_s(v) \subsetneq C_t(v)$), and we still have $g_t(v) = g_s(v)$, then we will add a new vertex to $\mathcal{G}'_t$ and define g_t to extend g_s so that $g_t : C_t(v) \cong C_t(v')$. To meet requirement P_e we will find vertices v'_e and w'_e in $\mathcal{G}'$ such that if $\Phi_e(v'_e, w'_e) = 1$ then $\mathcal{G}' \models \neg E v'_e w'_e$ and if $\Phi_e(v'_e, w'_e) = 0$ then $\mathcal{G}' \models E v'_e w'_e$. Let $\{X_s\}_{s\in\omega}$ be a computable approximation to X. Find minimal $e \leq s+1$ such that P_e requires attention and either: (1) $\Phi_{e,s+1}(v', w') \downarrow \neq 0$ for some v' and w' that are free for P_e and are such that $C_s(v') \neq C_s(w')$; or (2) $\Phi_{e,s+1}(v', w') = 0$ for the least pair $(v, w) \in X_s$ for which v' and w' both free for P_e. (We have used the shorthand $g_s^{-1}(v') = v$.)

If such e does not exist then go on to the next stage. Otherwise, if case (1), declare P_e does not require attention, and declare all current and future members of $C_s(v')$ and $C_s(w')$ not free for all P_t with $t > e$. In case (2), speed up the enumerations of X and $\mathcal{G}$ until either (A) we find a stage $t > s$ where $(v, w) \notin X_t$ or (B) we find a new component $C_t(z)$ on which we have yet to define g and into which $C_s(v)$ and $C_s(w)$ disjointly embed. In case (A), move to stage $t+1$ of the construction. In case (B), extend $C_s(v')$ and $C_s(w')$ to a single component, denoted by $C_t(z')$, such that $C_t(z') \cong C_t(z)$; build new copies $C_t(v')$ and $C_t(w')$ isomorphic to $C_t(v)$ and $C_t(w)$, respectively; redefine g_t by mapping z to z', v to (the new) v', w to (the new) w', and extending the map to an isomorphism $\mathcal{G}_t \cong \mathcal{G}'_t$. Declare all current and future members of $C_t(v')$, $C_t(w')$, and $C_t(z')$ not free for all P_t with $t > e$, declare P_t requires attention for all $t > e$, and declare P_e does not require attention. Continue to stage $t+1$ of the construction. This completes the construction for $\mathcal{G}'_{s+1}$.

One can now show by induction on e that each requirement is satisfied, having only caused finitely many components to be non-free for lower priority requirements. The marking of all relevant components as "not free" for lower priority requirements after an action for P_e ensures that g and g^{-1} are only ever re-defined finitely often on any given input. This together with the fact that at each stage s, $g_s : \mathcal{G}_s \cong \mathcal{G}'_s$ shows that g establishes an isomorphism between $\mathcal{G}$ and $\mathcal{G}'$. Thus $\mathcal{G} \cong \mathcal{G}'$, but the reachability relation is not decidable on $\mathcal{G}'$. $\square$

5 Application

We now apply results obtained to the class of automatic graphs. Recall that a graph is *automatic* if its domain is finite automaton recognizable, and there exists a two tape synchronous finite automaton that recognizes the edge relation. For precise definition and examples see [15]. Given an automatic graph $\mathcal{G}$, a vertex v, and $\Phi(x)$ a first order formula, one can effectively decide if $\Phi(v)$ is true [15]. Hence, if G is automatic then the function that outputs the number of edges for any given vertex v in G is computable. We now apply our results above to the following theorem.

Theorem 5. *Let $\mathcal{G}$ be an automatic graph such that either $\mathcal{G}$ is strongly locally finite or the reachability in $\mathcal{G}$ is finite automata recognizable. Then the reachability problem for $\mathcal{G}$ is intrinsically decidable if and only if the reachability relation and its complement both are existentially definable.*

Proof. If $\mathcal{G}$ is strongly locally finite then Theorem 1 does the job. This follows from the fact that for automatic strongly locally finite graphs the size function is always computable. For the second part of the theorem one applies the Ash-Nerode theorem. Indeed, assume that $\mathcal{G} = (V, E)$ is automatic and the reachability relation R is finite automata recognizable. Then, (V, E, R) is an automatic structure. Hence, the Ash-Nerod condition holds true for this structure. The rest follows from the Ash-Nerode theorem.

References

1. P. A. Abdulla, K. Čerāns, B. Jonsson, and Y. Tsay. Algorithmic analysis of programs with well quasi-ordered domains. *ICom*, 160:109–127, 2000.
2. P. A. Abdulla, A. Collomb-Annichini, A. Bouajjani, and B. Jonsson. Using forward reachability analysis for verification of lossy channel systems. *Formal Methods in System Design*, 25(1):39–65, 2004.
3. L. Aceto, A. Burgueno, and K.G. Larsen. Model checking via reachability testing for timed automata. In *TACAS '98: Proceedings of the 4th International Conference on Tools and Algorithms for Construction and Analysis of Systems*, pages 263–280, London, UK, 1998. Springer-Verlag.
4. E. Allender. Reachability problems: An update. In *CiE*, pages 25–27, 2007.
5. R. Alur and D. Dill. Theory of timed automata. *TCS*, 126:183–235, 1994.
6. C. J. Ash and A. Nerode. Intrinsically recursive relations. In *Aspects of effective algebra (Clayton, 1979)*, pages 26–41. Upside Down A Book Co. Yarra Glen, Vic., 1981.
7. A. Bouajjani, J. Esparza, and O. Maler. Reachability analysis of pushdown automata: Application to model checking. In *Proc. Intern. Conf. on Concurrency Theory (CONCUR'97)*. LNCS 1243, 1997.
8. A. Bouajjani, J. Esparza, S. Schwoon, and J. Strejcek. Reachability analysis of multithreaded software with asynchronous communication. In *FSTTCS*, pages 348–359, 2005.
9. A. Bouajjani and T. Touili. On computing reachability sets of process rewrite systems. In *RTA*, pages 484–499, 2005.
10. D. Caucal. On infinite terms having a decidable monadic theory. In *Mathematical foundations of computer science*, volume 2420 of *Lecture Notes in Comput. Sci.*, pages 165–176, Berlin, 2002. Springer.
11. T. Colcombet and C. Löding. Transforming structures by set interpretations, 2006. Technical report AIB-2006-07 of RWTH Aachen.
12. Yu. L. Ershov, S. S. Goncharov, A. Nerode, J. B. Remmel, and V. W. Marek, editors. *Handbook of recursive mathematics. Vol. 1*, volume 138 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, Amsterdam, 1998.
13. Yu. L. Ershov, S. S. Goncharov, A. Nerode, J. B. Remmel, and V. W. Marek, editors. *Handbook of recursive mathematics. Vol. 2*, volume 139 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, Amsterdam, 1998.
14. O. H. Ibarra, T. Bultan, and J. Su. On reachability and safety in infinite-state systems. *International Journal of Foundations of Computer Science*, 12(6):821–836, 2001.
15. B. Khoussainov and A. Nerode. Automatic presentations of structures. In *Logic and computational complexity (Indianapolis, IN, 1994)*, volume 960 of *Lecture Notes in Comput. Sci.*, pages 367–392. Springer, Berlin, 1995.

16. L. Libkin. *Elements of finite model theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer-Verlag, Berlin, 2004.
17. T. Rybina and A. Voronkov. A logical reconstruction of reachability. In *Ershov Memorial Conference*, pages 222–237, 2003.
18. R. I. Soare. *Recursively enumerable sets and degrees*. Perspectives in Mathematical Logic. Springer-Verlag, Berlin, 1987. A study of computable functions and computably generated sets.
19. R. E. Tarjan. Depth-first search and linear graph algorithms. *SIAM J. Comput.*, 1(2):146–160, 1972.