The Boundary Between Decidability and Undecidability for Transitive Closure Logics

N. Immerman
Dept. of Comp. Sci.
UMass, Amherst
immerman@cs.umass.edu

A. Rabinovich
School of Comp. Sci.
Tel-Aviv Univ.,
rabinoa@post.tau.ac.il

T. Reps
Comp. Sci. Dept.,
Univ. of Wisconsin, Madison
reps@cs.wisc.edu

M. Sagiv
School of Comp. Sci.
Tel-Aviv Univ.,
msagiv@post.tau.ac.il

G. Yorsh
School of Comp. Sci.
Tel-Aviv Univ.,
gretay@post.tau.ac.il

Abstract

To reason effectively about programs it is important to have some version of a transitive closure operator so that we can describe such notions as the set of nodes reachable from a program's variables. On the other hand, with a few notable exceptions, adding transitive closure to even very tame logics makes them undecidable.

In this paper we explore the boundary between decidability and undecidability for transitive closure logics. Rabin proved that the monadic second order theory of trees is decidable although the complexity of the decision procedure is not elementary. If we go beyond trees, however, undecidability comes immediately.

We have identified a rather weak first-order language called $\exists\forall(\mathrm{DTC}[E])$ that goes beyond trees, includes a version of transitive closure, and is decidable. We show that satisfiability of $\exists\forall(\mathrm{DTC}[E])$ is NEXPTIME complete. We furthermore show that essentially any reasonable extension of $\exists\forall(\mathrm{DTC}[E])$ is undecidable.

Our main contribution is to demonstrate these sharp divisions between decidable and undecidable. We also compare the complexity and expressibility of $\exists\forall(\mathrm{DTC}[E])$ with related decidable languages including MSO(trees) and guarded fixed point logics.

We mention possible applications to systems some of us are building which use decidable logics to reason about programs.

1 Introduction

To reason effectively about programs it is important to have some version of a transitive closure operator so that we can describe such notions as the set of nodes reachable from a program's variables. On the other hand, with a few notable exceptions, adding transitive closure to even very tame logics makes them undecidable.

In this paper we explore the boundary between decidability and undecidability for transitive closure logics. Rabin [12] proved that the monadic second order theory of trees is decidable although the complexity of the decision procedure is not elementary. If we go beyond trees, however, undecidability comes immediately.

Modal logics such as the μ calculus have proved extremely useful. The μ calculus has an EXPTIME-complete satisfiability problem [3] and the same has been shown true even for the more expressive guarded fixed point logic, as long as the vocabulary remains of bounded arity [6]. Guarded fixed point logic can express reachability from a specific constant, or from some point of a specific color, and it can restrict this reachability to be along paths specified, for example, by a regular expression. What it cannot express is a reachability relation between a pair of variables, i.e., that there is a path from u to v.

We have identified a rather weak first-order language called $\exists\forall(\mathrm{DTC}[E])$ that goes beyond trees, includes a version of this sort of transitive closure, and is decidable. We show that satisfiability of $\exists\forall(\mathrm{DTC}[E])$ is NEXPTIME complete. We furthermore show that **essentially any reason-**

able extension of $\exists\forall(\text{DTC}[E])$ is undecidable.

The main contribution of this paper is to demonstrate the above sharp divisions between decidable and undecidable. We also compare the complexity and expressibility of $\exists\forall(\text{DTC}[E])$ with related decidable languages including MSO(trees) and guarded fixed point logics.

The main application we have in mind is for the static analysis methods that we are pursuing. Very generally, we model the properties of an infinite set of data structures that can be generated by the program we are analyzing, using a bounded set of first-order, three-valued structures, [13]. In [14] it is shown that this modeling can be improved so that it computes the most precise possible transformation summarizing each program step, through the use of decidable logics.

Furthermore, in [9] we show that we can use a method we call "structure simulation" to significantly extend the sets of data structures that we can model while staying in the setting of trees (for monadic, second-order logic) or graphs for which the transitive closure information that we need is restricted to deterministic paths (for $\exists\forall(\text{DTC}[E])$).

The advantage of $\exists\forall(\text{DTC}[E])$ compared with MSO(trees) is that while the former is usually much more expressive, we can go beyond trees in the latter. As an example, to express reachability in dynamic, undirected graphs as in [2], we need not only a spanning forest, but a record of all the remaining edges in the undirected graph [9].

Fig. 1 summarizes results concerning the decidability and complexity of satisfiability for relevant logics. All the languages will be defined precisely in the next two sections. For previously known results we include a reference and for results new to this paper we include the number of the relevant theorem. **Note:** we assume throughout this paper that the arity of all relation symbols is bounded. (In all of our actual applications, relations are of arity at most two. If we allowed unbounded arity, i.e., arity n, then some of the complexity bounds would significantly increase.)

2 Background and Tiling

As we have mentioned, being able to express reachability is crucial for our applications. However, adding a transitive-closure operator tends to make even very tame logics undecidable. We use $(\text{TC}_{u,u'}\,\varphi)$ to denote the reflexive, transitive closure of binary relation $\varphi(u,u')$ [8]. **Note:** In this paper we will confine our attention to applications of $\text{TC}[\varphi]$ for which φ is quantifier-free and TC-free.

For example, consider the simple, decidable logic FO^2. This is first-order logic restricted to having only two variables, x, y. Grädel et al. [5] prove that if we add the transitive closure operator (TC) to FO^2 then the result is undecidable. In fact, they prove that even $\text{FO}^2(\text{DTC})$ is undecidable. Here DTC — deterministic transitive closure — is a restriction of transitive closure to paths that have no choices. For the binary relation $E(x,y)$, define $E_d(x,y)$ as follows,

$$E_d(x,y) \overset{\text{def}}{=} E(x,y) \ \wedge\ \forall z(E(x,z) \rightarrow z = y)\ .$$

That is, if vertex v has more than one E-edge leaving it, then it has no E_d-edges. Then define DTC as follows:

$$\text{DTC}[E] \overset{\text{def}}{=} \text{TC}[E_d]\ .$$

It is surprising that $\text{FO}^2(\text{DTC})$ is undecidable, but the proof is that even this seemingly very weak language is strong enough to express tilings.

Definition 2.1 Define a *tiling problem*, $\mathcal{T} = \langle T, R, D\rangle$, to consist of a finite set of tile types, $T = \{t_0, \ldots t_k\}$, together with horizontal and vertical adjacency relations, $R, D \subseteq T^2$. Here $R(a,b)$ means that tiles of type b fit immediately to the right of tiles of type a, and $D(a,b)$ means that tiles of type b fit one step down from those of type a. A *solution* to a tiling problem is an arrangement of instances of the tiles in a rectangular grid such that a t_0 tile occurs in the top left position and a t_k tile occurs in the bottom right position, and all adjacency relationships are respected. $\square$

Given a Turing machine, M, and an input, w, we can build a tiling problem, $\mathcal{T}$, of size $O(|M|+|w|)$, such that $\mathcal{T}$ has a solution iff M on input w eventually halts. Here any correct tiling solution would represent an accepting computation of M on input w. Think of t_0 as representing the initial state and t_k as representing the final accepting state. Thus, as is well known, any logic that can express tilings has undecidable finite satisfiability – and general satisfiability – problems.

(Standard definitions of tiling problems only require t_0 at the top left, and do not also ask for t_k at the lower right. This minor change does not affect the undecidability and complexity results, but makes some of our constructions slightly simpler.) See [1] for a nice treatment of tiling problems, as well as discussions of many relevant decidable and undecidable logics.

3 Decidability of $\exists\forall(\text{DTC}[E])$

We start with the first-order logic $\exists\forall$ consisting of first-order formulas in prenex form with all existential quantifiers preceding all universal quantifiers. It is well known and easy to see that the satisfiability problem for $\exists\forall$ is decidable: Let $\varphi \in \exists\forall$. Form the Skolemization, φ_S, by replacing the existential quantifiers, $\exists x_1, \ldots, x_k$, by new constants, $c_1, \ldots, c_k$. Suppose $\mathcal{A} \models \varphi_S$. Let $\mathcal{C}$ be the substructure of $\mathcal{A}$ whose universe consists of the constant symbols appearing in φ_S. Since φ_S is universal, we have that $\mathcal{C} \models \varphi_S$. Thus φ has a model iff it has a small model, i.e., one of size less than $|\varphi|$. We say that $\exists\forall$ has the *small model property*, in this case with models of at most linear size. To test if a universal formula, φ_S, is satisfiable, we would guess a structure, $\mathcal{A}$, of size at most $n = |\varphi_S|$ and then check that

Decidable Language	Complexity of Satisfiability	Citation
μ calculus	EXPTIME complete	[3]
Guarded Fixed Point	EXPTIME complete	[6]
MSO(trees)	non-elementary	[12]
FO^2	NEXPTIME complete	[10, 4]
$\exists\forall$	Σ_2^p complete	[1]
$\exists\forall(\mathrm{TC}^-)$	Σ_2^p complete	Prop 3.1
$\exists\forall(\mathrm{DTC}[E])$	NEXPTIME complete	Th 3.2, 3.4
$\exists\forall(\mathrm{TC}, f)$	NEXPTIME complete	Cor 4.1

Undecidable Language	Citation
$\mathrm{FO}^2(\mathrm{TC})$	[5]
$\mathrm{FO}^2(\mathrm{DTC})$	[5]
$\forall(\mathrm{TC}^+[E])$	Cor 5.4
$\forall(\mathrm{DTC}^+)$	Th 5.4
$\forall(\mathrm{DTC}^-[E])$	Th 6.3

Figure 1. Summary of the decidability and complexity, and the undecidability of the logics we study. The arity of all relation symbols is bounded. The results are the same for $\forall$ and $\exists\forall$.

$\mathcal{A} \models \varphi_S$. Testing whether a given structure satisfies an input, universal first-order formula is co-NP complete. Thus satisfiability of $\exists\forall$ formulas is in, and in fact complete for, Σ_p^2, the second-level of the polynomial-time hierarchy.

Since the existential quantifiers in $\exists\forall$ formulas can be eliminated by adding constants, we will limit our discussion to universal formulas. Let $\forall(\mathrm{DTC})$ consist of universal formulas in which DTC may occur. Unfortunately, as we will see, satisfiability of $\forall(\mathrm{DTC})$ and $\forall(\mathrm{TC})$ are undecidable (Theorem 5.4).

It is the positive occurrences of TC that mess up the satisfiability of $\forall(\mathrm{TC})$. Let $\exists\forall(\mathrm{TC}^-)$ consist of formulas in prenex form in which all occurrences of TC are negative.

Proposition 3.1 *Satisfiability of* $\exists\forall(\mathrm{TC}^-)$ *is decidable with complexity complete for* Σ_p^2.

Proof: The above argument for $\exists\forall$ continues to work. If $\varphi \in \exists\forall(\mathrm{TC}^-)$ is satisfiable, let $\mathcal{A} \models \varphi_S$, where φ_S is the Skolemization of φ. As above, let $\mathcal{C}$ be the substructure of $\mathcal{A}$ whose universe consists of the constant symbols appearing in φ_S. Then $\mathcal{C} \models \varphi_S$ because if a path did not exist in $\mathcal{A}$ then it still does not exist in $\mathcal{C}$. (Recall that we only apply TC to quantifier-free formulas.) Furthermore, we can test in polynomial time whether such a path exists in $\mathcal{C}$. Thus the complexity of satisfiability remains Σ_p^2 complete. □

Define $\exists\forall(\mathrm{DTC}[E])$ to be the restriction of $\exists\forall(\mathrm{DTC})$ in which the language has only one binary relation symbol, E, (plus unary relation symbols and constants), and all applications of DTC are positive occurrences of the form $\mathrm{DTC}[E]$. In addition, we include in $\exists\forall(\mathrm{DTC}[E])$ **arbitrary negative occurrences** of $\mathrm{TC}[\varphi]$ for φ quantifier-free. However, it is very important that there are **no negative occurrences of** DTC, for otherwise the language would become undecidable (Theorem 6.3).

Theorem 3.2 $\exists\forall(\mathrm{DTC}[E])$ *has the small model property, with models of size at most* $2^{O(n^2)}$*, where* n *is the size of the formula. Thus satisfiability of* $\exists\forall(\mathrm{DTC}[E])$ *is decidable, with complexity at most NEXPTIME.*

Proof: Let $\varphi \in \exists\forall(\mathrm{DTC}[E])$ be satisfiable and let $\mathcal{A} \models \varphi$. We will show that there exists a model $\mathcal{B} \models \varphi$ such that $\|\mathcal{B}\| \leq 2^{O(n^2)}$. Here $\|\mathcal{B}\|$ denotes the cardinality of the universe of the structure $\mathcal{B}$, and $n = |\varphi|$,

Let $c_1 \dots c_k$ be the constants occurring in φ. For each pair of constants, c_i, c_j, such that $\mathcal{A} \models \mathrm{DTC}[E](c_i, c_j)$, there is a unique path p_{ij} from c_i to c_j in $\mathcal{A}$. Let $\mathcal{A}'$ be the substructure of $\mathcal{A}$ whose universe consists of the constants plus all vertices on all the paths, p_{ij}.

We claim that $\mathcal{A}' \models \varphi$. To see this, first observe that for any two elements a, b of the universe of $\mathcal{A}'$ we have that,

$$\mathcal{A} \models \mathrm{DTC}[E](a,b) \Rightarrow \mathcal{A}' \models \mathrm{DTC}[E](a,b) \qquad \textbf{(3.3)}$$

(That the converse need not hold is exploited in the proof of Theorem 6.2.) Since a and b occur on paths p_{ij}, if $\mathcal{A} \models \mathrm{DTC}[E](a,b)$ then the path from a to b must be along the paths, p_{ij}. Thus $\mathcal{A}' \models \mathrm{DTC}[E](a,b)$ holds as well.

Since $\mathcal{A}'$ is a substructure of $\mathcal{A}$ and φ is a universal formula with only positive occurrences of DTC, it follows from Equation (3.3) that $\mathcal{A}' \models \varphi$. (Note that the negative occurrences of $\mathrm{TC}[\varphi]$ with φ quantifier-free do not bother us. Since $\mathcal{A}'$ is a substructure of $\mathcal{A}$ it follows that if $\mathcal{A} \models \neg\mathrm{TC}[\varphi](a,b)$, then $\mathcal{A}' \models \neg\mathrm{TC}[\varphi](a,b)$ as well.)

Structure $\mathcal{A}'$ consists of a set of "trees" directed from leaf to root, all of whose leaves and roots are constants; however, (1) some of the "trees" may end in a cycle rather than a root; and (2) multiple edges may occur from some of the roots to other vertices. Note that if there is more than one edge from vertex v, then v does not occur on any DTC path except perhaps as the last point. For this reason, if there are multiple edges in $\mathcal{A}$ from constant c_i, then we will remove all such edges and instead add a new unary relation symbol Q_i true of all the vertices that had edges from c_i; and we will modify φ accordingly. (More explicitly, we would change all occurrences of "$E(x,y)$" to "$E(x,y) \vee (x = c_i \wedge Q_i(y))$".) Thus we have eliminated (2), and we may assume that the graph $\mathcal{A}'$ has outdegree at most one.

Note that some of the paths, $p_{i,j}$, $p_{i',j'}$ may intersect. If so, for simplicity we identify the first point of intersection for each pair of paths as a new constant symbol. Observe that there are a total of at most $k-1$ such new constant symbols. Thus from now on we will only consider *direct paths* $p_{i,j}$ containing no intermediate constants. See Fig. 2 for a sample such graph, $\mathcal{A}'$, where constant symbols c_7, c_8, c_9 have been added.

Thus $\mathcal{A}'$ consists of k' constants and at most k' direct paths, $p_{i,j}$, where $k' \leq 2k-1$. Let r be the number of unary relation symbols and let m be the number of (universally quantified) variables occurring in φ. We claim that no direct path, $p_{i,j}$, need have length greater than $2^{rm}+m+1$. Suppose on the contrary that the length of $p_{1,2}$ is greater than $2^{rm}+m+1$. Let the color of a vertex be the set of unary relation symbols that it satisfies. Thus there are 2^r possible colors and 2^{rm} possible m-tuples of colors. Thus there must be at least two identically colored m-tuples, $u_1, \ldots, u_m$, and $v_1, \ldots, v_m$, in the interior of $p_{1,2}$. (By an m-tuple we mean m vertices occurring consecutively along the path.) Form the structure $\mathcal{B}$ from $\mathcal{A}'$ by deleting vertices u_2 through v_1 and adding an edge from u_1 to v_2.

We claim that $\mathcal{B} \models \varphi$. It suffices to show that for any m-tuple of points from $\mathcal{B}$, $b_1, b_2, \ldots, b_m$, there is a corresponding, isomorphic[1] m-tuple from $\mathcal{A}'$, $a_1, a_2, \ldots, a_m$. Note that every point in $\mathcal{B}$ is in $\mathcal{A}'$, and furthermore, the only difference between $\mathcal{B}$ and $\mathcal{A}'$ concerning these points is that $E(u_1, v_2)$ holds in $\mathcal{B}$ but not in $\mathcal{A}'$.

If any b_i is not on the path $p_{1,2}$, then it is answered by the identical point in $\mathcal{A}'$. We may thus assume the most difficult case namely that $b_1, b_2, \ldots, b_m$ are all in the path $p_{1,2}$. Assume for simplicity that they occur in order. Our only problem is if for some ℓ, $b_\ell = u_1$ and $b_{\ell+1} = v_2$. In this case, we let $a_t = b_t$ for $t \leq \ell$, but we let $a_{\ell+1} = u_2$. Similarly, if $b_{\ell+i-1} = v_i$ for all $i \in \{2, \ldots s\}$, then we must let $a_{\ell+i-1} = u_i$. Consider the first gap (if any), i.e., b_i and b_{i+1} are not consecutive. We have that $b_i = v_z$ and $a_i = u_z$, for some z. We can let $a_j = b_j$ for $j > i$, see Fig. 3. Note that we have replaced some v_i's by u_i's but all unary relations, edge relations and connectivity have been preserved. Thus as desired $a_1, a_2, \ldots, a_m$ is isomorphic to $b_1, b_2, \ldots, b_m$.

Thus $\mathcal{B} \models \varphi$ as desired. We can continue shortening any remaining paths of length greater than $2^{rm}+m+1$. It follows that there is a model $\mathcal{B}$ of φ and $\|B\| \leq (2k-1)\,(2^{rm}+m+1) \leq 2^{|\varphi|^2}$, as desired. □

It follows from Theorem 3.2 that the satisfiability of $\exists\forall(\mathrm{DTC}[E])$ formulas can be checked in NEXPTIME. We next show that this cannot be improved.

Theorem 3.4 *The satisfiability of* $\exists\forall(\mathrm{DTC}[E])$ *formulas is NEXPTIME-complete.*

Proof: Let $\mathcal{T}$ be a tiling problem as in Definition 2.1 and let n be a natural number. It is an NEXPTIME-complete problem to test on input $(\mathcal{T}, 1^n)$ whether there is a $\mathcal{T}$-tiling of a square grid of size 2^n by 2^n [11].

We will write the formula φ_n so that it expresses exactly a solution to this tiling problem. There will be two constants, s, denoting the cell in the upper left corner and t denoting the cell in the lower right corner. The desired model will look like the following:

$$\begin{array}{rllll} s = & [1,1,t_0] & \cdots & [1,2^n,t] & \\ & [2,1,t'] & \cdots & [2,2^n,t''] & \\ & \vdots & & \vdots & \\ & [2^n,1,t'''] & \cdots & [2^n,2^n,t_k] & = t \end{array}$$

The binary relation E will hold between each pair of consecutive tiles, including, for example, $[1,2^n,t]$ and $[2,1,t']$. We will include the following unary relation symbols: $H_1, \ldots H_n$ indicating the horizontal position as an n-bit number, $V_1, \ldots V_n$ indicating the vertical position, and $T_0, \ldots T_k$ indicating the tile type.

The formula φ_n is the conjunction of the following assertions:

1. $T_0(s) \ \wedge \ \bigwedge_{i=1}^{n} \big(\neg H_i(s) \wedge \neg V_i(s)\big)$
2. $T_k(t) \ \wedge \ \bigwedge_{i=1}^{n} \big(H_i(t) \wedge V_i(t)\big)$
3. $\forall x \bigwedge_{0 \leq i < j \leq k} \neg(T_i(x) \wedge T_j(x))$
4. $\forall x, y \big((\mathrm{Suc}_v(x,y) \rightarrow \mathrm{Vert}(x,y)) \ \wedge \ (\mathrm{Suc}_h(x,y) \rightarrow \mathrm{Hor}(x,y))\big)$
5. $\mathrm{DTC}[E](s,t) \ \wedge \ \forall x, y\big(E(x,y) \rightarrow \mathrm{Next}(x,y)\big)$

Here (1) says that s is the first tile and has tile type t_0. Similarly, (2) says that t is the last tile and has tile type t_k. We have chosen for simplicity to encode the tile types in unary so we need (3) which says that tile types are mutually exclusive.

Conjunct (4) says that the arrangement of tiles honors $\mathcal{T}$'s adjacency requirements. The abbreviation $\mathrm{Suc}_h(x,y)$ means that x and y have the same vertical position and y's horizontal position is one more than that of x. $\mathrm{Suc}_v(x,y)$ means that x and y have the same horizontal position and y's vertical position is one more than that of x. The abbreviations $\mathrm{Hor}(x,y)$ and $\mathrm{Vert}(x,y)$ are disjunctions over the tile types asserting that the tiles in positions x and y are horizontally, respectively vertically, compatible, for example,

$$\mathrm{Hor}(x,y) \ \equiv \ \bigvee_{R(t_i,t_j)} (T_i(x) \wedge T_j(y)) \qquad \textbf{(3.5)}$$

Finally, (5) says that there is a path from s to t. The abbreviation $\mathrm{Next}(x,y)$ means $\mathrm{Suc}_h(x,y)$ or x has horizontal

[1]More explicitly, we mean that the map taking each b_i to a_i is an isomorphism of the induced substructures of $\mathcal{B}$ and $\mathcal{A}'$ generated by $b_1, \ldots, b_m$ and $a_1, \ldots, a_m$, respectively. This may be thought of as an Ehrenfeucht-Fraïssé game in which the spoiler chooses the b_i's and the duplicator answers with the a_i's [8].

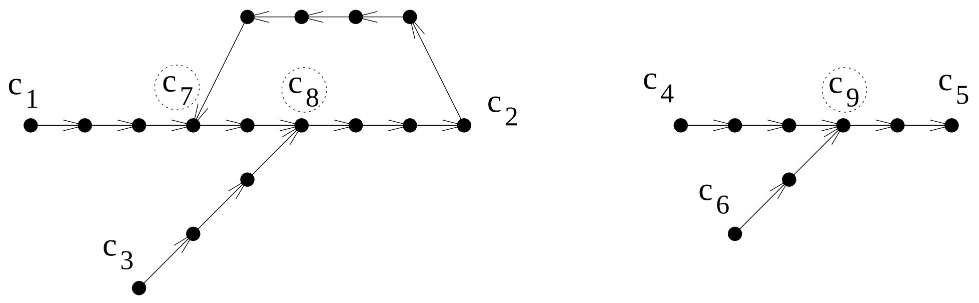

Figure 2. In proof of Theorem 3.2: sample configuration of $\mathcal{A}'$ with constants c_7, c_8, c_9 added.

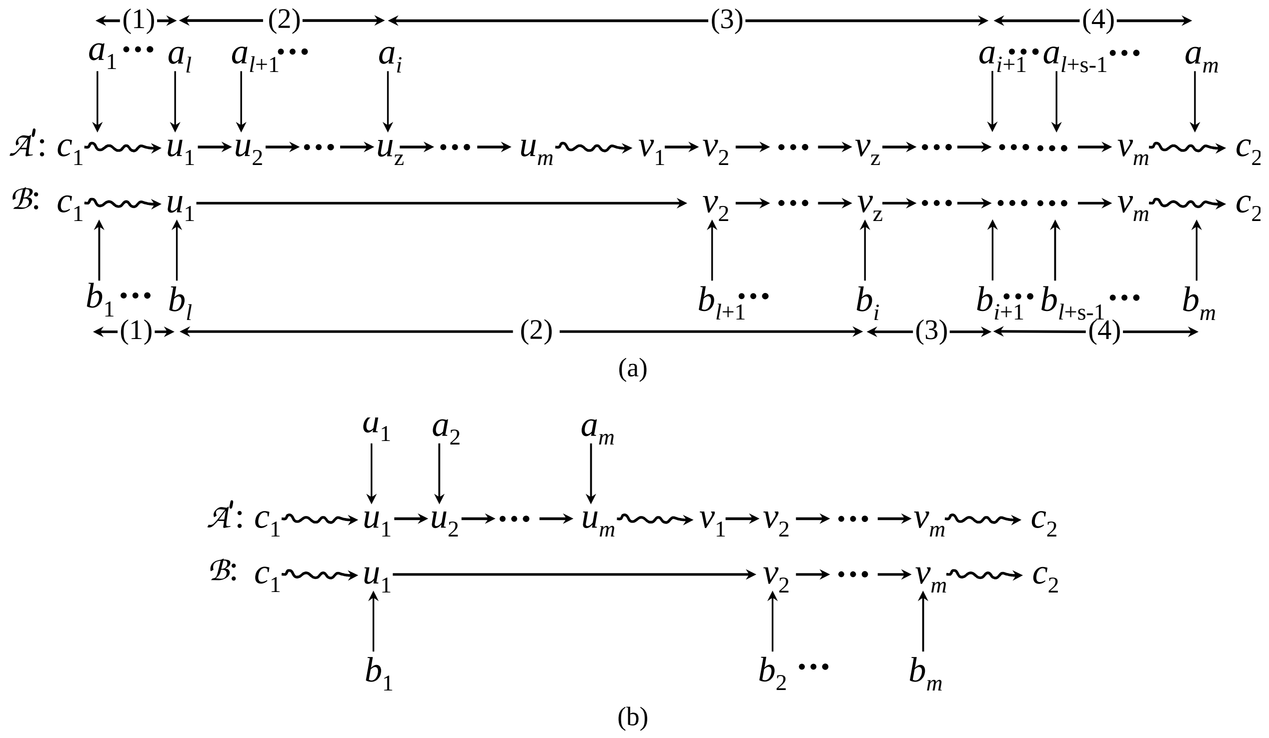
(1)
(2)
(3)
(4)
$a_1 \cdots$
a_l
$a_{l+1} \cdots$
a_i
$a_{i+1} \cdots a_{l+s-1} \cdots$
a_m
$\mathcal{A}'$: c_1 u_1 u_2 u_z u_m v_1 v_2 v_z v_m c_2
$\mathcal{B}$: c_1 u_1 v_2 v_z v_m c_2
$b_1 \cdots$
b_l
$b_{l+1} \cdots$
b_i
$b_{i+1} \cdots b_{l+s-1} \cdots$
b_m
(a)
a_1
a_2
a_m
b_1
$b_2 \cdots$
b_m
(b)

Figure 3. Illustration of how for every m-tuple of points $b_1, \dots, b_m$ from $\mathcal{B}$ there is a corresponding isomorphic m-tuple of points $a_1, \dots, a_m$ from $\mathcal{A}'$. (a) In region (2) of $\mathcal{B}$, $b_l, \dots, b_i$ are assigned consecutive points; similarly, in region (2) of $\mathcal{A}'$, $a_l, \dots, a_i$ are assigned consecutive points. Because b_i and b_{i+1} are separated by two or more E edges in region (3) of $\mathcal{B}$ (i.e., there a "gap"), the assignments for $a_{i+1}, \dots, a_m$ in region (4) of $\mathcal{A}'$ can match those for $b_{i+1}, \dots, b_m$ in region (4) of $\mathcal{B}$ exactly. (b) Degenerate case of "no gap": $b_1, b_2, \dots, b_m$ are assigned the m consecutive points $u_1, v_2, \dots, v_m$, and there is no pair b_i, b_{i+1} that are separated by two or more E edges in $\mathcal{B}$. In this case, $a_1, a_2, \dots, a_m$ are assigned the consecutive points $u_1, u_2, \dots, u_m$.

position 2^n, y has horizontal position 1, and y's vertical position is one more than that of x. $\Box$

The formula φ_n described in the above proof can be written in length $O(n)$ using only two variables. When satisfiable, it has a minimal model of size $2^{\Omega(n)}$. In Corollary 7.3 we extend the above argument, showing that the $2^{O(n^2)}$ bound of Theorem 3.2 is in fact optimal. For this we need a variant of the above formulas that use n variables.

4 Logics With One Function Symbol

We next discuss the language, $\forall(\mathrm{TC}, f)$, consisting of universal first-order logic with a transitive closure operator, one unary function symbol, plus arbitrary unary relation symbols and constants. This is closely related to the language $\exists\forall(\mathrm{DTC}[E])$. One important difference is that in $\forall(f)$ we may write a formula that has only infinite models.[2]

It is well known that the satisfiability and finite-satisfiability problems for monadic second-order logic with a single unary function symbol are decidable[3], although their complexities are not elementary, even when restricted to first-order quantification [12, 1, 7].

It is not hard to modify the proofs of Theorems 3.2 and 3.4 to apply to $\forall(\mathrm{TC}, f)$. (For functions, the implication of Equation (3.3) is a biimplication and thus the result goes through for positive and negative DTC's.)

Corollary 4.1 *The finite satisfiability problem for $\forall(\mathrm{TC}, f)$ is NEXPTIME complete.*[4]

Proof: If a formula $\varphi \in \forall(\mathrm{TC}, f)$ has a finite model, $\mathcal{A}$, then it must have a model of the form $\mathcal{A}'$ as in the proof of Theorem 3.2. The only difference is that since f must be a total function there are no roots so all trees end in cycles. The size of the smallest model is still $2^{O(n^2)}$. The difference in counting is slight namely applications of the function symbol, f, can extend the apparent number of constant symbols: $f(c_i)$ behaves like a new constant symbol, c_i' and $f(x)$ behaves like a new universally quantified variable, y, such that $E(c_i, c_i')$ and $E(x, y)$, respectively, must hold. Thus, the proof of Theorems 3.2 and 3.4 go through if we replace k and m by qk and qm, respectively, where q is the number of occurrences of f in φ. $\Box$

[2]For example: $\forall x, y(c \neq f(x) \ \wedge \ (f(x) = f(y) \to x = y))$.

[3]This is equivalent to the MSO theory of trees with multiple successor functions.

[4]This holds as well for the general satisfiability problem. For infinite structures there is a similar "small model" except that from some constants there is an infinite chain that intersects no other vertices of the structure. The infinite chain must repeat an m-tuple of colors and can from thereafter repeat exactly. Thus it has a representation of size $2^{O(n^2)}$.

5 Undecidability of Related Logics

We next show that most reasonable extensions of the language $\exists\forall(\mathrm{DTC}[E])$ can express the solution to tiling problems and thus are undecidable. To begin we show:

Theorem 5.1 *Satisfiability of $\forall(\mathrm{DTC}^+[V], \mathrm{DTC}^+[H])$ — universal logic with two binary relations, V, H, and their positive deterministic transitive closure — is undecidable.*

Proof: Let $\mathcal{T}$ be a tiling problem (Definition 2.1). We show how to write a formula $\varphi \in \forall(\mathrm{DTC}^+[V], \mathrm{DTC}^+[H])$ such that φ is satisfiable iff $\mathcal{T}$ has a solution.

Formula φ contains four constant symbols, a, b, c, d representing the four corners of the solution to $\mathcal{T}$, see Fig. 4.

We assert that every element satisfies exactly one of the tile relations, $T_0, \ldots, T_k$. We assert $T_0(a) \wedge T_k(d)$, i.e., the upper left tile is t_0 and the lower right is t_k. We assert that H and V paths exist between the four corners: $\mathrm{DTC}[H](a, b) \wedge \mathrm{DTC}[H](c, d) \wedge \mathrm{DTC}[V](a, c) \wedge \mathrm{DTC}[V](b, d)$.

We add a unary predicate, *Last*, true of tiles in the rightmost column and in this column we have H-edges go down along the V-edges, i.e., $\mathit{Last}(x) \wedge \mathit{Last}(y) \to (H(x, y) \leftrightarrow V(x, y))$. We do this so that we can express the fact that H-edges continue all the way to the right in every row. We do this by asserting, $\forall x \mathrm{DTC}[H](x, d)$.

We next assert that H and V edges satisfy the corresponding horizontal and vertical tiling constraints, using the formulas Hor and Vert as in Equation (3.5). $\forall x, y((H(x, y) \wedge \neg \mathit{Last}(x) \ \to \ \mathrm{Hor}(x, y)) \wedge (V(x, y) \ \to \ \mathrm{Vert}(x, y)))$.

Next we assert that the intermediate rows are filled in: $\forall x, y, x', y' \big((H(x, y) \wedge V(x, x') \wedge V(y, y')) \to H(x', y')\big)$.

Finally, we assert that the columns are filled in and line up: $\forall x, y, x', y' \big((\neg \mathit{Last}(x) \wedge H(x, y) \wedge V(x, x') \wedge H(x', y')) \to V(y, y')\big)$.

It is not hard to see that the conjunction of the above assertions is equivalent to the existence of a solution to the tiling problem, $\mathcal{T}$. Thus satisfiability of $\forall(\mathrm{DTC}^+[V], \mathrm{DTC}^+[H])$ is undecidable. $\Box$

Theorem 5.4 shows that a second binary relation over which we can take DTC causes undecidability. We can modify the proof to show that even if there is only one (positive) occurrence of DTC, the logic is still undecidable if a second binary relation is allowed, or if DTC is allowed to be taken not just over the relation E, but over a formula that also involves unary relation symbols.

Theorem 5.2 *Satisfiability of $\forall(\mathrm{DTC}^+)$ is undecidable. This holds even if there is only one occurrence of DTC and only one binary relation symbol.*

Proof: We modify the proof of Theorem 5.4 so that the path from a to d through the tiled rectangle is along a single snake-like path of the edge predicate, E, as in Fig. 5.

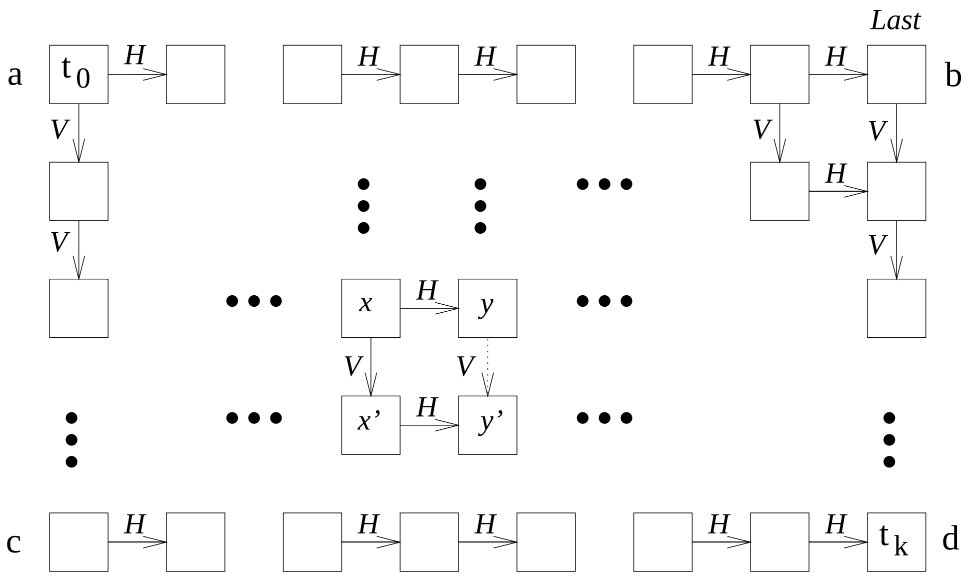

Figure 4. A tiling as expressed in Theorem 5.4. (The last column satisfies *Last*.)

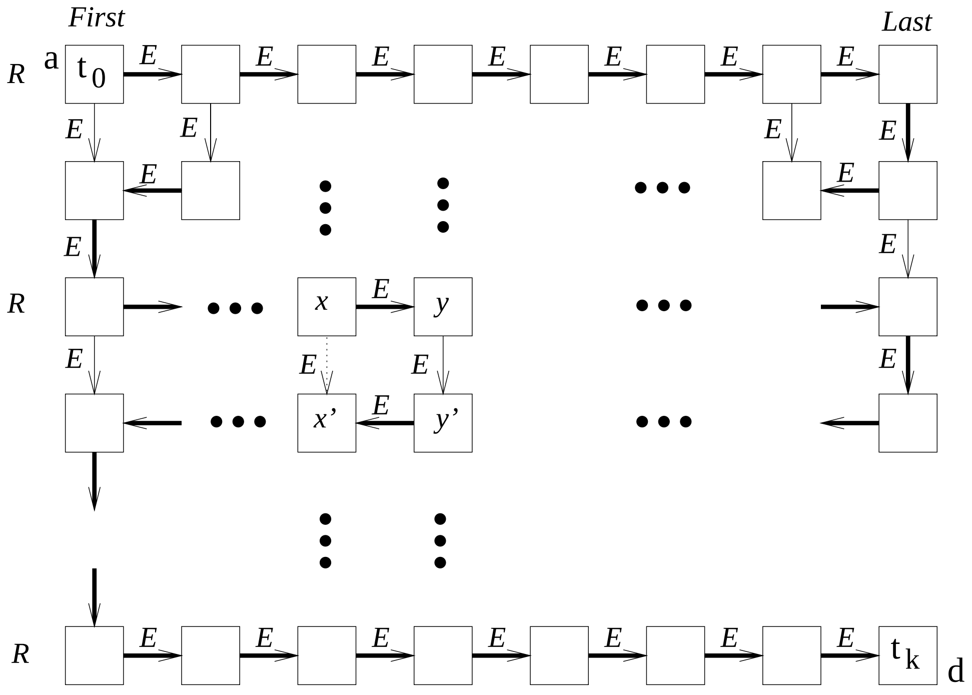

Figure 5. A tiling expressed with a single occurrence of DTC as in Theorem 5.4.

We do this by adding unary relation *First* denoting the first column of the tiling rectangle, plus the relation R modifying alternating rows of the rectangle indicating that the E-path goes to the right on the odd-numbered rows.

Define the edges along the snake-like path, $\sigma(x,y) \equiv E(x,y) \wedge ((R(x) \leftrightarrow R(y)) \vee (\mathit{First}(x) \wedge \neg R(x) \wedge R(y)) \vee (\mathit{Last}(x) \wedge R(x) \wedge \neg R(y)))$.

The single use of DTC is the assertion $\mathrm{DTC}[\sigma](a,d)$. We also assert the completion of squares (see Fig. 5),

$(E(x,y) \wedge E(y,y') \wedge E(y',x') \wedge (R(x) \leftrightarrow R(y)) \wedge (R(x') \leftrightarrow R(y')) \wedge (R(y) \leftrightarrow \neg R(y'))) \rightarrow E(x,x')$.

Finally, we add the following assertions which together make sure that all models must be valid tilings:

1. $T_0(a) \wedge T_k(d) \wedge \mathit{First}(a) \wedge \mathit{Last}(d)$
2. $\neg\big(\mathit{First}(x) \wedge \mathit{Last}(x)\big)$
3. $\bigvee_{i=0}^{k} T_i(x) \ \wedge \bigwedge_{0\le i<j\le k} \neg(T_i(x) \wedge T_j(x))$
4. $(E(x,y) \wedge (R(x) \leftrightarrow \neg R(y))) \rightarrow \big((\mathit{First}(x) \leftrightarrow \mathit{First}(y)) \wedge (\mathit{Last}(x) \leftrightarrow \mathit{Last}(y))\big)$
5. $E(x,y) \rightarrow \neg\Big((R(x) \wedge R(y) \wedge (\mathit{Last}(x) \vee \mathit{First}(y))) \vee (\neg R(x) \wedge \neg R(y) \wedge (\mathit{Last}(y) \vee \mathit{First}(x)))\Big)$
6. $\big((E(x,y) \wedge R(x) \wedge R(y)) \vee (E(y,x) \wedge \neg R(x) \wedge \neg R(y))\big) \rightarrow \mathrm{Hor}(x,y)$
7. $\big(E(x,y) \wedge (R(x) \leftrightarrow \neg R(y))\big) \rightarrow \mathrm{Vert}(x,y)$

Again formulas Hor and Vert are as in Equation (3.5). The conjunction of the universal closure of all the above assertions thus assert a solution to the tiling problem, $\mathcal{T}$, as desired. □

We remark that if in the proof of Theorem 5.4 we reverse the edges that are not σ edges, then we can use $\mathrm{TC}[E]$ in lieu of $\mathrm{DTC}[\sigma]$ and the proof goes through. Thus we have,

Corollary 5.3 *Satisfiability of* $\forall(\mathrm{TC}[E])$ *is undecidable. This holds even if there is only one occurrence of TC and it occurs as TC[E] where E is the only binary relation symbol.*

Note that the formulas in Theorems , , and Corollary use only two variables except in the completion of squares formula. In fact, we can write the whole thing with only two variables as follows: We reverse the vertical edges in the even columns. We then assert that each tile is in a cycle of 2×2 tiles: $\forall xy(E(x,y) \rightarrow \mathrm{DTC}[E](y,x))$.

Corollary 5.4 *The undecidability results of Theorems , , and Corollary all remain true for the corresponding languages with only two variables.*

6 Undecidability of $\forall(\mathrm{DTC}^+[E], \mathrm{DTC}^-[E])$

We were quite surprised to find that although $\forall(\mathrm{TC}^-)$ is decidable, $\forall(\mathrm{DTC}^-)$, and even $\forall(\mathrm{DTC}^-[E])$ are not. We give the somewhat subtle proof in this section. First we show that $\forall(\mathrm{DTC}^-[E])$ has an infinity axiom.

Proposition 6.1 *There is a sentence in* $\forall(\mathrm{DTC}^-[E])$ *that is satisfiable, but only in an infinite model.*

Proof: The idea is that we know that if $E(c_0,c_1)$ and $\neg\mathrm{DTC}[E](c_0,c_1)$ both hold, then there must be another edge from c_0. We can use this observation to write an infinity axiom that essentially expresses the existence of a successor function. We simply express the conjunction of the following formulas:

1. $\forall v(v \neq c_1 \rightarrow (E(v,c_1) \wedge \neg\mathrm{DTC}[E](v,c_1)))$
2. $\forall v u_1 u_2(v \neq c_1 \wedge E(u_1,v) \wedge E(u_2,v) \rightarrow u_1 = u_2)$
3. $c_0 \neq c_1 \wedge \forall v \neg E(v,c_0)$

(1) says that every vertex besides c_1 has an edge to c_1 but not a DTC path to c_1, so it must have outdegree greater than 1; (2) says that every vertex besides c_1 has in-degree at most one; and (3) says that c_0 has in-degree 0. Thus there must be an infinite chain of edges starting at c_0.

These formulas are satisfied by a model that contains the natural numbers plus a new point called c_1, with edges $E(n,c_1)$ and $E(n,n+1)$, for $n = 0,1,\ldots$. □

Theorem 6.2 $\forall(\mathrm{DTC}^+[E], \mathrm{DTC}^-[E])$ *is undecidable.*

Proof: We take as our starting point the undecidability proof of Theorem 5.4. Our new idea is to remove all of the non-boldface E's in Fig. 5 and to replace them by a gadget of new green vertices, satisfying the unary relation symbol, G, and associated edges. The existence of the green vertices and their associated edges will be implied by the "not DTC trick" described in Proposition 6.1, together with some universal first-order statements that make sure that the vertical edges continue to be attached appropriately.

Just as in the proof of Theorem 5.4, we express the existence of a tiling. Since we have removed the non-boldface E's we can now simply express the path from the first tile to the last as $\mathrm{DTC}[E](a,d)$.

To get our gadget we add two new constants, b, for the top, rightmost tile, and c_1 for the top, rightmost green vertex, just below it. The green path proceeds in the opposite direction of the non-green, tile path directly above it, see Fig. 6.

We make the following assertions. These all concern the green row below each R, i.e., right-going, row of tiles. For simplicity, we skip the analogous case below each left-going row of tiles.

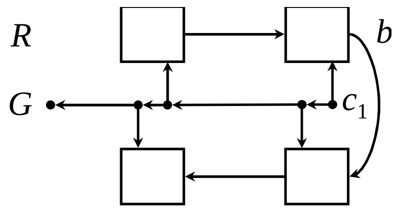

Figure 6. Gadget used in Theorem 6.2.

1. $G(c_1) \wedge E(c_1, b) \wedge \forall ux(E(c_1, x) \wedge G(x) \wedge E(b, u) \rightarrow E(x, u))$
2. $\forall x((\neg G(x) \leftrightarrow \text{DTC}[E](x, d)) \wedge (\neg G(x) \leftrightarrow \text{DTC}[E](a, x)))$
3. $\forall xyz\big(G(x) \wedge E(x, y) \wedge E(x, z) \wedge y \neq z \rightarrow (G(y) \leftrightarrow \neg G(z))\big)$
4. $\forall uvxyz\big(\neg G(u) \wedge \neg G(v) \wedge G(x) \wedge G(y) \wedge G(z) \wedge R(u) \wedge R(v) \wedge E(v, u) \wedge E(x, u) \wedge E(x, y) \wedge E(y, z) \rightarrow E(z, v)\big)$
5. $\forall uvxyz\big(\neg G(u) \wedge \neg G(v) \wedge G(x) \wedge G(y) \wedge G(z) \wedge \neg R(u) \wedge \neg R(v) \wedge E(u, v) \wedge E(x, u) \wedge E(x, y) \wedge E(y, z) \rightarrow E(z, v)\big)$
6. $\forall u, v, x, y\big(\neg G(u) \wedge \neg G(v) \wedge G(x) \wedge G(y) \wedge R(u) \wedge \neg R(v) \wedge E(x, u) \wedge E(x, y) \wedge E(y, v) \rightarrow \text{Vert}(u, v)\big)$

(1) starts us out by saying that c_1 is green, has an edge to b, and its green successor has an edge to the tile directly below b. (2) says that green vertices do not have DTC paths to d, but all non-green vertices do; it also says that all the non-green edges occur on the DTC-path from a to d. (3) says that if the outdegree of a green vertex is at least 2, then it has a green and a non-green successor. We will assure later, inductively, that each green vertex has an edge to a non-green vertex. Since the non-green vertex has a DTC-path to d, but the green vertex does not, this assures that the green vertex has outdegree 2. (4) is an inductive condition which says that if x, y, and z are consecutive green nodes, and if x points up to a non-green node, u, then z points up to u's predecessor, v. (5) is the similar condition for the edges going down.

Finally, condition (6) asserts that these green gadgets transmit the vertical information between the non-green, i.e., tile, nodes as desired. □

Theorem 6.2 leaves open the question of the decidability of $\forall(\text{DTC}^-[E])$. It would seem that the positive use of DTC was crucial in the statement $\text{DTC}[E](a, d)$. However, even this can be replaced by the "not DTC trick". (The positive uses of DTC in formula (2) of the proof of Theorem 6.2 can easily be removed.) The conclusion is that $\forall(\text{DTC}^-[E])$ is undecidable.

Theorem 6.3 $\forall(\text{DTC}^-[E])$ *is undecidable.*

Proof: This amounts to modifying the proof of Theorem 6.2. We remove the assertion $\text{DTC}[E](a, d)$ and replace it using the "not DTC trick". More explicitly, we add another unary predicate B true of the tiles, and we add another constant, c_0. Then we make the following additional assertions:

1. $B(a) \wedge \forall x(B(x) \wedge x \neq d \rightarrow E(x, c_0) \wedge \neg\text{DTC}[E](x, c_0))$
2. $\forall xy(B(x) \wedge y \neq c_0 \wedge E(x, y) \rightarrow B(y))$
3. The in-degree for B-vertices from B-vertices is at most one, and it is zero for a.

(1) and (2) together assert that each B-vertex besides d has an edge to another B-vertex. It follows that either $\text{DTC}[E](a, d)$ holds, or there is an infinite path. Thus, the formula is finitely satisfiable iff the corresponding tiling problem has a solution. □

7 Complexity of the Decision Procedure

In this section, we study the complexity of the decision procedure for $\exists\forall(\text{DTC}[E])$. The first thing we do is look more carefully at the proof of Theorem 3.4, and show that our lower bound is tight, matching the $2^{O(n^2)}$ upper bound of Theorem 3.2.

Lemma 7.1 *The formula φ_n in the proof of Theorem 3.4 may be written in length $O(n)$.*

Proof: The only difficulty in keeping φ_n to total size $O(m)$ is in writing the formulas $\text{Suc}_h(x, y)$ and $\text{Suc}_v(x, y)$. These are nearly identical and we will restrict our attention to $\text{Suc}_h(x, y)$. Recall that $\text{Suc}_h(x, y)$ means that the horizontal position of y is one greater than the horizontal position of x. This can most simply be written as follows:

$$\begin{aligned} \text{Suc}_h(x, y) \quad \equiv \quad & \bigvee_{i=1}^{n} \Big[\bigwedge_{j>i} (H_j(x) \wedge \neg H_j(y)) \\ & \wedge \quad (\neg H_i(x) \wedge H_i(y)) \\ & \wedge \quad \bigwedge_{j<i} (H_j(x) \leftrightarrow H_j(y)) \Big] \end{aligned}$$

However, the length of the above formula is $O(n^2)$. We can decrease the size by keeping track of the position i in the above formula. We do this by adding $2n$ more unary relation symbols, G_j, K_j, $1 \leq j \leq n$. The intuitive meaning of $K_i(x)$ is that it is bit i of the horizontal number that will be incremented as we go from x to its successor. This means that $\neg H_i(x)$, and for all $j > i$, $H_j(x)$; i.e., there is a "0" in position i, and a "1" in each position to the right of i.

(We are thinking of the bit positions as 1 to n with 1 being the high-order bit, and n the low-order bit.)

The intuitive meaning of $G_j(x)$ is that $j > i$ where $K_i(x)$. We also use the abbreviation $L_j(x) \equiv \neg(K_j(x) \vee G_j(x))$. (The mnemonic is that G holds for elements in positions "greater" than the K position; L holds for elements in "lesser" positions.)

The advantage of having these new relations around is that we can now reduce the length of $\text{Suc}_h(x, y)$ as follows:

$$\begin{aligned} \text{Suc}_h(x,y) \quad \equiv \quad & \bigwedge_{j=1}^{n} \Big[(G_j(x) \wedge H_j(x) \wedge \neg H_j(y)) \\ & \vee \quad (K_j(x) \wedge \neg H_j(x) \wedge H_j(y)) \\ & \vee \quad (L_j(x) \wedge (H_j(x) \leftrightarrow H_j(y))) \Big] \end{aligned}$$

Finally, we must write down several more conditions. The conjunction of the following conditions assures that the new relations G_i and K_i are defined correctly.

1. $\forall x(K_1(x) \vee K_2(x) \vee \cdots \vee K_n(x) \vee (H_1(x) \wedge H_2(x) \cdots H_n(x)))$
2. $\forall x(\bigwedge_{i=1}^{n-1} (K_i(x) \to G_{i+1}(x)))$
3. $\forall x(\bigwedge_{i=1}^{n-1} (K_{i+1}(x) \to L_i(x)))$
4. $\forall x(\bigwedge_{i=1}^{n-1} (L_{i+1}(x) \to L_i(x)))$
5. $\forall x(\bigwedge_{i=1}^{n-1} (G_i(x) \to G_{i+1}(x)))$
6. $\forall x(\bigwedge_{i=1}^{n} \neg(K_i(x) \wedge G_i(x))$
7. $\forall x(\bigwedge_{i=1}^{n} ((G_i(x) \to H_i(x)) \wedge (K_i(x) \to \neg H_i(x))))$

□

It follows from Lemma 7.1 and the proof of Theorem 3.4 that we can write a sequence of formulas $\varphi_n \in \exists\forall(\text{DTC}[E])$, $n = 1, 2, \ldots$ such that $|\varphi_n| = O(n)$, φ_n has only two variables, and yet φ_n's smallest model is of size $2^{\Omega(n)}$. This is the best possible with only two variables. To match the $2^{O(n^2)}$ upper bound of Theorem 3.2, we need a formula with n variables.

We can count up to 2^{n^2} using a sequence of n consecutive vertices, each with a number between 1 and 2^n. We will add n more unary relation symbols, C_i, $1 \le i \le n$. A tile will then be encoded by n vertices as follows:

$$\begin{array}{llll} [C_1, h_1, v_1, t] & [C_2, h_2, v_2, t] & \cdots & [C_n, h_n, v_n, t] \\ [C_1, h'_1, v'_1, t'] & [C_2, h'_2, v'_2, t'] & \cdots & [C_n, h'_n, v'_n, t'] \end{array}$$

That is, the first n vertices hold tile t with its (collective) horizontal and vertical numbers $\langle h_1, \ldots, h_n \rangle$ and $\langle v_1, \ldots, v_n \rangle$ having values between 1 and 2^{n^2}, the next n vertices hold tile t' with the successor number, etc. Using very similar ideas to the proof of Lemma 7.1 we can prove,

Lemma 7.2 *Given any tiling problem, $\mathcal{T}$, we can write a sequence of formulas φ'_n of length $O(n)$, $n = 1, 2, \ldots$, such that φ_n is satisfiable iff there is a solution to $\mathcal{T}$ that is a 2^{n^2} by 2^{n^2} square.*

It follows that

Corollary 7.3 *The $2^{O(n^2)}$ upper bound of Theorem 3.2 is optimal.*

8 Conclusions

We have introduced the language $\exists\forall(\text{DTC}[E])$ which is a decidable transitive closure logic that goes beyond trees. We have shown that all the reasonable extensions of $\exists\forall(\text{DTC}[E])$ that we could think of are undecidable. Uses of $\exists\forall(\text{DTC}[E])$ exist but how useful it might be remains to be seen. The following questions are worth considering:

– Unlike our other undecidability proofs which only required two variables, our proof of the undecidability of $\forall(\text{DTC}^-[E])$ used five variables. We suspect that this can be improved.

– We showed that the satisfiability of $\exists\forall(\text{DTC}[E])$ is NEXPTIME complete. The lower bound depended on a formula that describes an exponentially long sequence of colors. We suspect that in practice the formulas one encounters would have much, much shorter sequences of color types. We suspect that techniques related to Ehrenfeucht-Fraïssé games can automatically find the relevant color sequences. These might lead to a satisfiability algorithm that is feasible in practice.

References

[1] E. Börger, E. Grädel, and Y. Gurevich. *The Classical Decision Problem*. Springer-Verlag, 1996.

[2] G. Dong and J. Su. Space-bounded foies. In *Principles of Database Systems*, pages 139–150. ACM Press, 1995.

[3] E. Allen Emerson and Charanjit S. Jutla. The complexity of tree automata and logics of programs. In *Proc. 29th IEEE Symposium on Foundations of Computer Science*, pages 328–337. IEEE Computer Society Press, 1988.

[4] E. Grädel, Ph. Kolaitis, and M. Vardi. On the decision problem for two-variable first-order logic. *Bulletin of Symbolic Logic*, 3:53–69, 1997.

[5] E. Grädel, M. Otto, and E. Rosen. Undecidability results on two-variable logics. *Archive of Math. Logic*, 38:313–354, 1999.

[6] E. Grädel and I. Walukiewicz. Guarded fixed point logic. In *Proc. 14th IEEE Symposium on Logic in Computer Science*, pages 45–54. IEEE Computer Society Press, 1999.

[7] J.G. Henriksen, J. Jensen, M. Jørgensen, N. Klarlund, B. Paige, T. Rauhe, and A. Sandholm. Mona: Monadic second-order logic in practice. In *Tools and Algorithms for the Construction and Analysis of Systems, First International Workshop, TACAS '95, LNCS 1019*, 1995.

[8] N. Immerman. *Descriptive Complexity*. Springer-Verlag, 1999.

[9] N. Immerman, A. Rabinovich, T. Reps, M. Sagiv, and G. Yorsh. Verification via structure simulation. To appear in CAV'04, 2004.

[10] M. Mortimer. On languages with two variables. *Zeitschr. f. math. Logik u. Grundlagen d. Math*, 21:135–140, 1975.

[11] C. Papadimitriou. *Computational Complexity*. Addison–Wesley, 1994.

[12] M. Rabin. Decidability of second-order theories and automata on infinite trees. *Trans. Amer. Math. Soc.*, 141:1–35, 1969.

[13] M. Sagiv, T. Reps, and R. Wilhelm. Parametric shape analysis via 3-valued logic. *Trans. on Prog. Lang. and Syst.*, pages 217–298, 2002.

[14] G. Yorsh, T. Reps, and M. Sagiv. Symbolically computing most-precise abstract operations for shape analysis. To appear in TACAS, 2004.