Well structured program equivalence is highly undecidable

ROBERT GOLDBLATT, Victoria University of Wellington
MARCEL JACKSON, La Trobe University

We show that strict deterministic propositional dynamic logic with intersection is highly undecidable, solving a problem in the Stanford Encyclopedia of Philosophy. In fact we show something quite a bit stronger. We introduce the construction of program equivalence, which returns the value T precisely when two given programs are equivalent on halting computations. We show that virtually any variant of propositional dynamic logic has a Π_1^1-hard validity problem if it can express even just the equivalence of well-structured programs with the empty program skip. We also show, in these cases, that the set of propositional statements valid over finite models is not recursively enumerable, so there is not even an axiomatisation for finitely valid propositions.

Categories and Subject Descriptors: F.3 [**Logics and Meanings of Programs**]; F.4.1 [**Mathematical Logic and Formal Languages**]: Mathematical Logic

General Terms: PDL, undecidability

Additional Key Words and Phrases: Deterministic propositional dynamic logic, fixset, program equivalence, intersection

1. INTRODUCTION

Determinism has played an unusual role in the study of programs. While most actual algorithms are deterministic in nature, there has traditionally been a strong theme on modeling programs nondeterministically. Indeed the standard semantics for classic program logics such as dynamic logic, treat programs as binary relations on the state space of a computer, and (in the standard relational semantics) apply constructions such as program union and reflexive transitive closure, which fall outside of conventional programming languages. Of course, there are numerous good reasons for this: one is attempting to reason about programs more than reason from within them. Stating that "property α is true after some number of iterates of p" is a useful assertion to make and close to the kind of questions that need to be asked in applications such as formal program verification.

Another occasionally cited reason for the focus on nondeterminism is that logics based over deterministic programs (partial functions) are known to experience an unexpected explosion in complexity. In fact this is only half true. Satisfiability for strict

deterministic PDL (deterministic program variables, and program union and * replaced by only conventional constructions of structured programming: if-then-else and while-do) is only PSPACE-complete [Halpern and Reif 1983], while the full PDL (over nondeterministic programs), and even strict PDL has EXPTIME-complete complexity (see [D. Harel and Tiuryn 2000] for these and other similar results). However the introduction of program intersection produces enormous contrast. Standard (that is, nondeterministic) PDL with intersection is decidable [Danecki 1984], albeit doubly exponential time complete [Lange and Lutz 2005] (a result that has recently been extended to PDL with intersection and converse [S. Göller and Lutz 2009]) while Harel [Harel 1985] showed that *deterministic* PDL with intersection (DIPDL) has a Π_1^1-hard validity problem, at the first level of the analytical hierarchy!

Strangely, it seems unknown what happens between the relatively well behaved SD-PDL and the unimaginably badly behaved DIPDL. The decidability of strict deterministic propositional dynamic logic with intersection (SDIPDL) appears open and indeed is stated as such in the Winter 2008 edition of the Stanford Encyclopedia of Philosophy [Balbiani 2008]. While program intersection is not a conventionally encountered programming construction, it is easy to simulate the intersection of two actual programs p and q and return the result when and if they both halt and agree. Thus it is an available construct of conventional programming even if it is not expressible within the language of SDPDL.

Recently the second author (with Tim Stokes) has examined algebraic formulations of deterministic program logics [Jackson and Stokes 2009; 2011] and produced a very simple axiomatisation for the loop-free fragment of SDIPDL [Jackson and Stokes 2011]. The validity problem of this fragment is easily seen to be co-NP-complete (by guessing a finite invalidating model of size polynomial in the complexity of a given formula). The authors of [Jackson and Stokes 2011] were rather hopeful that despite Harel's famous negative result for DIPDL, the strict fragment might still be decidable. In the present article we show this is not the case: SDIPDL also suffers Π_1^1-hardness. In fact we show a more general result that concerns variants of PDL that are not necessarily deterministic. We identify a natural notion of "program equivalence" and show that this inevitably leads to Π_1^1-hardness when expressible in a variant of PDL, independently of the constraint of deterministic atomic programs. The Π_1^1-hardness of SDIPDL can be explained by the fact that in deterministic variants of PDL, intersection can be used to express program equivalence.

We also show that for variants of PDL capable of expressing program equivalence (such as SDIPDL) there is no axiomatisation possible for the propositions satisfiable on finite relational models.

2. PROGRAM CONSTRUCTIONS

We direct the reader to a text such as Harel, Kozen and Tiuryn [D. Harel and Tiuryn 2000] (particularly Sections 5.1 and 5.2) for a full formal definition of Propositional Dynamic Logic. PDL and its usual variants are two sorted, with variables consisting of atomic programs and atomic propositions (we use T for true and F for false). In the relational semantics, programs are binary relations on a set X (or partial maps in deterministic forms of PDL) and propositions are subsets of X. Propositions are given the usual operations of Propositional Logic (with the usual set theoretic semantics on X) and programs may be combined using ; (composition of relations), union $\cup$, and the unary operation * (reflexive transitive closure). There are also connectives enabling interaction between programs and propositions. Every program p and proposition α yields a new proposition by way of the modal necessity $[p]\alpha$ which has semantics $\{x \mid \forall y\, (x, y) \in p \Rightarrow y \models \alpha\}$; modal possibility is derived in the usual fashion as $\langle p \rangle \alpha := \neg[p]\neg\alpha$. Similarly, every proposition α yields a program via a "query operator"

?, with α? having semantics $\{(x,x) \mid x \models \alpha\}$. In *strict* (or *well structured*) forms of PDL, the operators of program union and * are replaced by the basic constructions of well-structured programming: aside from the already present composition, the constructions `while-do` and `if-then-else` are the standard choice (note that these constructions are expressible in full PDL). Note that the trivial program `skip` is T? while the the empty relation arises as F?. Many variants of PDL involve further constructions not described here. Of particular interest in this article are constructions relating to program intersection.

The usual semantics for program intersection is simply set-theoretic intersection of binary relations. Thus the program $p \cap q$ relates state s to state t provided that both p and q relate s to t. However even if $p \cap q$ relates state s to t, enacting p in state s might give rise to some t' outside of the range of the relation q. We consider a reasonable variant of intersection, which we refer to as *program equivalence*. For programs p, q, the proposition $p \bowtie q$ ("p tie q", or "p is equivalent to q") is true at a point a if p is equivalent to q at a: in the relational semantics, $p \bowtie q$ has truth set equal to

$$\{a \mid (\forall b)\, (a,b) \in p \leftrightarrow (a,b) \in q\}.$$

Program equivalence can be expressed in SDIPDL as $\langle p \cap q \rangle \mathsf{T} \vee \neg(\langle p \rangle \mathsf{T} \vee \langle q \rangle \mathsf{T})$. And, provided query is included, SDPDL with program equivalence can express intersection: $p \cap q = (p \bowtie q)? \,;\, p$.

Our main results will use a construction weaker than program equivalence. Consider the unary operation `Fix` acting on programs p to produce a proposition $\mathtt{Fix}(p)$ that asserts that halting computations of p act effectlessly. In the relational semantics,

$$\mathtt{Fix}(p) = \{a \mid (\forall b)\, (a,b) \in p \rightarrow a = b\}.$$

Our main results are expressed in terms of `Fix`, however in proofs it is more convenient to use a construction $\mathtt{fix}(p)$, which we define as $\mathtt{Fix}(p) \wedge \langle p \rangle \mathsf{T}$. Note that $\mathtt{Fix}(p) = \mathtt{fix}(p) \vee [p]\mathsf{F}$, so that `fix` and `Fix` are interdefinable in any reasonable variant of PDL. But also, `fix` (whence `Fix`) can be expressed in terms of program equivalence as $p \bowtie$ `skip` (hence it is expressible if $\cap$ is expressible in the deterministic case). On the other hand, $\bowtie$ cannot be expressed using `Fix` because one can find models of DPDL that are closed under `Fix` but not under program equivalence (we omit the details of this claim).

A key observation in this note is that expressions of the form $[x^*]\alpha$ are expressible in the language of well-structured programs (provided that x and α are): as $[\mathtt{while}\, \alpha\, \mathtt{do}\, x]\mathsf{F}$. Expressions of the form $[(x \cup y)^*]\alpha$ are fundamental to Harel's original proof of the high undecidability of DIPDL: they are used to interpret an infinite grid. Expressions of this form are not in general expressible in strict forms of PDL, however the presence of `fix` enables something similar to be done in enough cases to encode tiling problems.

3. TILINGS

The undecidability results are proved by encoding tiling problems as originally employed by Harel [Harel 1985]. A *finite set of square tiles* is a finite set $\mathcal{T} = \{T_0, \ldots, T_{k-1}\}$ of "*tiles*" endowed with a pair of binary "edge" relations $\sim_h$ (horizontal) and $\sim_v$ (vertical). We interpret $T_i \sim_h T_j$ to mean that tile T_i can be placed on the left of tile T_j in a horizontal row. Likewise $T_i \sim_v T_j$ is interpreted to mean that T_i can be placed beneath T_j in a vertical column. A natural and very standard geometric restriction is that if $T_i \sim_h T_j$ and $T_k \sim_h T_j$ and $T_k \sim_h T_\ell$, then $T_i \sim_H T_\ell$ also. We will not make use of this restriction, though assuming it does not affect the computational complexity of the tiling problems we consider.

Consider the non-negative integer lattice $\omega \times \omega$ endowed with relations $\sim_h$ and $\sim_v$ defined by $(i,j) \sim_h (i+1,j)$ and $(i,j) \sim_v (i,j+1)$ for all $i,j \geq 0$ (here of course, lattice

is referring to square grids rather than ordered sets). A *tiling of the positive quadrant of the plane* (henceforth, a *tiling of the plane*) is a function from $\omega \times \omega$ into $\mathcal{T}$ that preserves the relations $\sim_h$ and $\sim_v$. Tilings of $\mathbb{Z} \times \mathbb{Z}$ are defined analogously.

We use two fundamental facts on tiling the plane.

—**Tiling Fact 1.** The following problem is Σ_1^1-complete. Given a finite set of tiles $\mathcal{T}$ with distinguished subset $\mathcal{N}$ of "neon" tiles. Is there is a tiling of the plane τ in which $\tau(0,0) = T_0$ and that $\tau^{-1}(\mathcal{N}) \cap \{(i,i) \mid i \in \omega\}$ is infinite (that is, the diagonal contains infinitely many neon tiles)?

—**Tiling Fact 2.** The following sets S_{period} and S_{notiling} are recursively inseparable: S_{period} is the set of finite sets of square tiles that can tile $\mathbb{Z} \times \mathbb{Z}$ periodically; S_{notiling} is the set of finite sets of tiles that cannot tile the plane at all.

Tiling Fact 2 can be found in Böger, Grädel and Gurevich [E. Böger and Gurevich 1997, Theorem 3.1.7]: tiling periodically means that there is a tiling of $\mathbb{Z}_n \times \mathbb{Z}_m$, with the obvious toroidal adjacency constraints (work modulo n horizontally and modulo m vertically). Tiling Fact 1 is a minor variant of some well known tiling problems investigated by Harel; see [Harel 1986] or [D. Harel and Tiuryn 2000] for example. We now give a brief sketch of a proof of the Σ_1^1-completeness claim. In [Harel 1986, p. 233], Harel shows that the following problem is Σ_1^1-complete: given a nondeterministic Turing machine program T, with initial state q_0 and started on a one-way infinite blank tape, does T return to the state q_0 infinitely often? We now reduce this problem to the problem in Tiling Fact 1. We use a modification of the standard translation of Turing machines into tiles, as presented, say, by Robinson [Robinson 1971]. Using the nomenclature of Robinson's article, there are essentially four kinds of tile (aside from the blank tile which we will not need, as we're only tiling the positive quadrant): the initial tiles (including one designated start tile T_0), the merge tiles, the action tiles and the alphabet tiles. The action tiles are constructed according to the commands of the Turing machine program. Provided that T_0 is placed at the position $(0,0)$, the tiling can only be completed to the nth row if the program can run for n steps of computation without halting. Moreover, each successfully tiled row encodes the configuration of the Turing machine tape at the corresponding step of computation.

Now duplicate all tiles except initial tiles and action tiles. For each duplicated tile, we make the second copy "neon", and adjust the horizontal edge constraints to ensure that neon tiles can be placed horizontally adjacent only to other neon tiles (and even then, only if they additionally satisfy the original edge constraints). Vertical constraints are unchanged however. Now, replace every action tile that encodes a transition into the state q_0, by a neon copy. These tiles are not to be duplicated: they are only neon. Also, action tiles not involving a transition into q_0 are never neon. Then, in any tiling of the plane, a row containing a neon tile must contain only neon tiles. Since each successfully tiled row can contain precisely one action tile, the following are equivalent: there is a computation that revisits state q_0 infinitely often; there is a tiling of the plane starting from T_0 and in which infinitely many rows are neon; there is a tiling of the plane starting from T_0 and in which infinitely neon tiles are placed on the diagonal. As the first of these is Σ_1^1-complete, so the problem in Tiling Fact 1 is Σ_1^1-hard. Completeness follows in the usual way.

4. MAIN ARGUMENT

Let $\mathcal{T} = \{T_0, \ldots, T_{k-1}\}$ be some fixed finite set of tiles. For $i = 0, 1, \ldots, k-1$ we let α_i denote an atomic proposition variable which we think of as corresponding to the placement of tile T_i. In order to produce our $\omega \times \omega$ grid we introduce four atomic program variables: E, W, S and N. Squares of the grid will be created by asserting statements of

the form $\mathtt{fix}(\mathsf{N}\,;\mathsf{E}\,;\mathsf{S}\,;\mathsf{W})$. We first define the propositions required, then explain how these force a tiling.

Step 0. Defining a square. We need to be able to find squares in both clockwise and anti-clockwise directions. We encode the clockwise square by the following proposition:

$$\mathtt{fix}(\mathsf{N}\,;\mathsf{S}) \wedge [\mathsf{N}]\mathtt{fix}(\mathsf{E}\,;\mathsf{W}) \wedge [\mathsf{N}\,;\mathsf{E}]\mathtt{fix}(\mathsf{S}\,;\mathsf{N}) \wedge [\mathsf{N}\,;\mathsf{E}\,;\mathsf{S}]\mathtt{fix}(\mathsf{W}\,;\mathsf{E}) \wedge \mathtt{fix}(\mathsf{N}\,;\mathsf{E}\,;\mathsf{S}\,;\mathsf{W}).$$

The anticlockwise square is defined in the dual way, following partial paths through $\mathsf{E}\,;\mathsf{N}\,;\mathsf{W}\,;\mathsf{S}$. We denote the conjunction of the two square propositions by square.

Step 1. To define a grid we use the statement ρ_1:

$$[\mathsf{N}^*][\mathsf{E}^*]\mathsf{square}$$

which, as observed above, can be expressed using only modal operators and the language of well-structured programs (instead of *).

Step 2. To force a tiling, we first let α denote the proposition that asserts that precisely one of the α_i is true. Then, for each i, let β_i denote the disjunction of all the atomic tile propositions α_j for which $T_i \sim_h T_j$. Similarly, we let β^i denote the disjunction of the atomic tile propositions α_j for which $T_i \sim_v T_j$. Then, provided we have an $\omega \times \omega$ grid, a tiling can be forced by ρ_2:

$$[\mathsf{N}^*][\mathsf{E}^*]\left(\alpha \wedge \bigwedge_{i=0}^{k-1} \left(\alpha_i \Rightarrow ([\mathsf{E}]\beta_i \wedge [\mathsf{N}]\beta^j) \right) \right)$$

Step 3. To force infinitely many neon tiles in the diagonal, first let neon denote the disjunction of the atomic neon tile propositions. Then we use ρ_3:

$$[(\mathsf{N}\,;\mathsf{E})^*]\langle(\mathsf{N}\,;\mathsf{E})^*\rangle\mathsf{neon}.$$

THEOREM 4.1. *Fix any variation VPDL of PDL capable of expressing the usual connectives on propositions, program composition, $\mathtt{while\text{-}do}$, modal operators and $\mathtt{fix}$. The validity problem for VPDL is Π^1_1-hard, regardless of whether atomic programs are assumed to be deterministic or not.*

PROOF. For any set of tiles $\mathcal{T}$, with neon subset $\mathcal{N}$, let γ denote $\alpha_0 \wedge \rho_1 \wedge \rho_2 \wedge \rho_3$. We claim that the following are equivalent:

(1) $\mathcal{T}$ can tile the positive quadrant of the plane with infinitely many neon tiles on the diagonal and with T_0 in the $(0,0)$ position;
(2) γ can be satisfied in some relational model where all atomic programs are deterministic (even injective partial functions);
(3) γ can be satisfied in some relational model.

Implication $1 \Rightarrow 2$ is routine, while $2 \Rightarrow 3$ is trivial. Now assume that γ is satisfied at some point of a relational model. We label this point by $a_{0,0}$. Now by ρ_1 we have that square holds at $a_{0,0}$. Thus, the program $\mathsf{N}\,;\mathsf{E}$ is defined at $a_{0,0}$, because $a_{0,0}$ is fixed by $\mathsf{N}\,;\mathsf{E}\,;\mathsf{S}\,;\mathsf{W}$. Then by ρ_3, there is a nontrivial iterate of $\mathsf{N}\,;\mathsf{E}$ at which neon is true. Thus there is a path of edges from $a_{0,0}$ alternating N and E and leading to a position at which neon is true. We label the points visited along this path (after $a_{0,0}$) by $a_{0,1}$, $a_{1,1}$, $a_{1,2}$, $a_{2,2},\ldots$; see the left picture in Figure 1. We do not rule out the possibility that some points in the model are labelled more than once: to produce the tiling, we consider only the labels of the selected points

Now as square holds at $a_{0,0}$, we have that after $\mathsf{N}\,;\mathsf{E}$ it is necessary that $\mathtt{fix}(\mathsf{S}\,;\mathsf{N})$ hold. Hence, in particular there is a point $a_{1,0}$ that is reached by an application of S from the point $a_{1,1}$. Again applying square at $a_{0,0}$, we have that after applying $\mathsf{N}\,;\mathsf{E}\,;\mathsf{S}$ it is necessary that $\mathtt{fix}(\mathsf{W}\,;\mathsf{E})$. Thus in particular, there is a point $a'_{0,0}$ west of $a_{1,0}$.
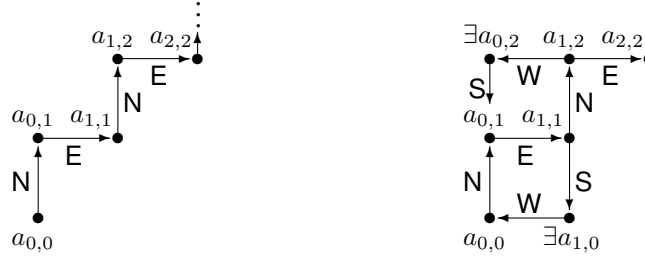
Fig. 1. Selecting the points $a_{i,j}$, and completing the $\omega \times \omega$ grid.

However $a'_{0,0}$ is reached by an application of $\mathsf{N}\,;\mathsf{E}\,;\mathsf{S}\,;\mathsf{W}$, which by square must fix $a_{0,0}$. Hence $a'_{0,0} = a_{0,0}$.

Similarly, ρ_1 ensures that square is true at the point $a_{0,1}$. We now construct a square anticlockwise through points $a_{0,1}, a_{1,1}, a_{1,2}$ and some new point $a_{0,2}$. The idea is essentially dual to the previous case: after applying $\mathsf{E}\,;\mathsf{N}$ (reaching $a_{1,2}$), it is necessary that $\mathtt{fix}(\mathsf{W}\,;\mathsf{E})$ be defined, thus we encounter some new point $a_{0,2}$. From here a further S is forced, and then as $\mathsf{E}\,;\mathsf{N}\,;\mathsf{W}\,;\mathsf{S}$ fixes $a_{0,1}$, we have the desired square.

So far we have not used all the power of the proposition square: in the right hand picture in Figure 1, the bottom left square has a different orientation to the square above it. However, each time we extended a new arrow from a point, we did so by way of propositions of the form $\mathtt{fix}(\mathsf{E}\,;\mathsf{W})$ (and so on): thus in fact every arrow drawn has an associated converse arrow labelled with the appropriate dual name (E switched with W and N switched with S). Once these edges are also drawn, both squares so far obtained are identical (two-way edges, with dual labels). So in fact, the process can be continued, working out outward from the central diagonal (with clockwise constructions below the horizontal and anti-clockwise constructions above) until a rectangular grid has been formed.

Then we apply ρ_3 a further time: extending the diagonal to a new point $a_{n,n}$ where neon is defined, and filling out the remaining pieces of a larger rectangle and so on.

In this way an infinite grid is interpreted, with neon tile propositions holding at infinitely many places on the diagonal. Furthermore, every position in this grid can now be visited by first iterating E and then iterating N. Now γ forces α_0 to be true at $a_{0,0}$. And then, working inductively outward from $a_{0,0}$, the proposition ρ_2 ensures that a tiling proposition holds at every one of the selected points and that neighbouring squares (horizontally or vertically) have tiling propositions that match the tiling constraints. Thus we interpreted a tiling of the positive quadrant of the plane in which neon tiles occur infinitely often along the diagonal. As the problem in Tiling Fact 1 is Σ_1^1-complete, thus satisfiability for VPDL is Σ_1^1-hard and validity is Π_1^1-hard. □

Recall that if atomic programs are deterministic, then intersection can be used to define $\mathtt{fix}$ on well-structured programs. This gives the following corollary.

COROLLARY 4.2. *Satisfiability for SDIPDL is* Π_1^1*-hard.*

Consider the operation of *program difference*:

$$p - q := \{(a, b) \mid (a, b) \in p \textbf{ and}(a, b) \notin q\}.$$

It is well known that standard PDL with program *complementation* is undecidable (see [D. Harel and Tiuryn 2000, Theorem 10.12]). Program difference can be expressed in terms of program complementation, but the reverse need not be true in the absence of

a universal program (that is, the universal relation in the relational semantics). As a second corollary, we show that standard PDL with program difference is Π_1^1-hard.

COROLLARY 4.3. *PDL with program difference* (*whence with program complementation*) *is* Π_1^1-*hard.*

PROOF. First observe that program intersection can be expressed from program difference: $p \cap q = p - (p - q)$. Now observe that $\mathtt{fix}(p) = (\langle p \rangle \mathsf{T}) \wedge ([p - (p \cap \mathtt{skip})]\mathsf{F})$. □

THEOREM 4.4. *Fix any variation VPDL of PDL capable of expressing the usual connectives on propositions, program composition,* `while-do`*, modal operators and* `fix`*. The set of VPDL propositions valid over finite relational models of VPDL is not recursively enumerable, whence there is no axiomatisation for VPDL over finite models.*

PROOF. Consider a finite set of tiles $\mathcal{T}$, and consider the proposition $\gamma_{\mathcal{T}} := \rho_1 \wedge \rho_2$. We first show that if $\gamma_{\mathcal{T}}$ is satisfied at some point $a_{0,0}$ in a model then $\mathcal{T}$ can tile the plane (whence $\mathcal{T} \notin \mathcal{S}_{\text{notiling}}$). The argument is similar to that used to prove Theorem 4.1, but we use ρ_1 to produce the diagonal (there are no neon tiles to consider). By ρ_1, the proposition square is true, which yields points $a_{0,1}$, $a_{1,1}$ and $a_{1,0}$, reached successively in following $\mathsf{N} \, ; \mathsf{E} \, ; \mathsf{S}$, with W taking $a_{1,0}$ back to $a_{0,0}$, and with $\mathsf{E} \, ; \mathsf{N} \, ; \mathsf{W} \, ; \mathsf{S}$ following through the points in reverse order. Now, by ρ_1 again, square is true at $a_{1,1}$. Thus we obtain points $a_{1,2}$, $a_{2,2}$ and $a_{2,1}$ forming the rest of a new square based at $a_{1,1}$. Now we can fill out these points to a 2×2 region using the same argument in the proof of Theorem 4.1. Then ρ_1 guarantees that square is true at $a_{2,2}$ and so on. Finally, once an $\omega \times \omega$ grid is interpreted, we can use ρ_2 to show that precisely one tiling proposition is true at $a_{0,0}$, and then force a tiling as in the proof of Theorem 4.1.

Now observe that if $\mathcal{T}$ can tile periodically: that is, can tile the torus $\mathbb{Z}_n \times \mathbb{Z}_m$, then $\gamma_{\mathcal{T}}$ can be satisfied in some finite model based on the nm points of $\mathbb{Z}_n \times \mathbb{Z}_m$.

Thus the set $\mathbb{S}$ of finitely satisfiable propositions contains $\{\gamma_{\mathcal{T}} \mid \mathcal{T} \in S_{\text{period}}\}$ and is disjoint from $\{\gamma_{\mathcal{T}} \mid \mathcal{T} \in S_{\text{notiling}}\}$. Now $\mathbb{S}$ is recursively enumerable (simply search for a finite satisfying model). But it cannot be recursive, because S_{period} and S_{notiling} are recursively inseparable. Hence $\mathbb{S}$ is not coRE. Whence the propositions valid over finite models of VPDL is not RE. □

We mention that in order to express `Fix` in terms of program equivalence we invoked the program `skip`. In the absence of `skip` (whence also query, as `skip = T?`), it is unclear if Theorem 4.1 and Theorem 4.4 hold (replacing `fix` by program equivalence). However all of the arguments relating to the encoding of tilings can be routinely adapted to the program equivalence situation, with some simplification. As a sketch: work with only N and E, and replace the proposition square by statements of the form $(\mathsf{N} \, ; \mathsf{E}) \bowtie (\mathsf{E} \, ; \mathsf{N})$.

REFERENCES

BALBIANI, P. 2008. Propositional dynamic logic. In *The Stanford Encyclopedia of Philosophy (Winter 2008 Edition)*, E. N. Zalta, Ed. URL = http://plato.stanford.edu/archives/win2008/entries/logic-dynamic/.

D. HAREL, D. K. AND TIURYN, J. 2000. *Dynamic Logic*. Foundations of Computer Science. MIT Press.

DANECKI, R. 1984. Nondeterministic propositional dynamic logic with intersection is decidable. In *Proceedings of the 5th Symposium on Computation Theory (Zaborów, Poland)*. Number 208. LNCS, 34–53.

E. BÖGER, E. G. AND GUREVICH, Y. 1997. *The Classical Decision Problem*. Springer.

FEARNLEY-SANDER, D. AND STOKES, T. 1997. Equality algebras. *Bull. Aust. Math. Soc. 56*, 177–191.

FEARNLEY-SANDER, D. AND STOKES, T. 2003. Varieties of equality structures. *Internat. J. Algebra Comput. 13*, 463–480.

HALPERN, J. AND REIF, J. 1983. The propositional dynamic logic of deterministic, well-structured programs. *Theoret. Comput. Sci. 27*, 127–165.

HAREL, D. 1985. Recurring dominoes: making the highly undecidable highly understandable. *Ann. Disc. Math. 24*, 51–72.

HAREL, D. 1986. Effective transformations on infinite trees with applications to high undecidability, dominoes and fairness. *J. ACM 33*, 224–248.

JACKSON, M. AND STOKES, T. 2009. Semigroups with `if-then-else` and halting programs. *Internat. J. Algebra Comput. 19*, 937–961.

JACKSON, M. AND STOKES, T. 2011? Modal restriction semigroups: toward and algebra of deterministic programs. *Internat. J. Algebra Comput., to appear*.

LANGE, M. AND LUTZ, C. 2005. 2-ExpTime lower bounds for propositional dynamic logics with intersection. *J. Symbolic Logic 70*, 1072–1086.

ROBINSON, R. 1971. Undecidability and nonperiodicity for tilings of the plane. *Inventiones Math. 12*, 177–209.

S. GÖLLER, M. L. AND LUTZ, C. 2009. PDL with intersection and converse: satisfiability and infinite-state model checking. *J. Symbolic Logic 74*, 279–314.

STOKES, T. 2006. On EQ-monoids. *Acta Sci. Math. (Szeged) 72*, 481–506.