Program Equivalence is Highly Undecidable in Dynamic Logic

Rob Goldblatt

Victoria University of Wellington

12th Asian Logic Conference

Wellington, December 15-20, 2011

Joint work with Marcel Jackson

‘Well structured program equivalence is highly undecidable’.
ACM Transactions on Computational Logic

Equivalent Programs

if $(p \vee \neg p)$ **then do** α**, else do** β

is equivalent to α

Equivalent Programs

if $(p \vee \neg p)$ **then do** α**, else do** β

is equivalent to α

```
begin {a > 0, b ⩾ 0}
      x := a; y := b;
      while y ≠ 0 do {gcd(a;b) = gcd(x,y)}
      begin r := x mod y;
            x := y;
            y := r
      end
      {x = gcd(a,b)}
end
```

```
{(x > 0) ∧ (y > 0)}
begin a := x; b := y;
{(a > 0) ∧ (b > 0) ∧ (gcd(a,b) = gcd(x,y))}
      repeat
            while a > b do a := a − b;
            while b > a do b := b − a
      until a = b
      {a = b = gcd(x,y)}
end
```

```
begin x := u; y := v;
      k := 1;
      while even(x) ∧ even(y) do
      begin x := x div 2; y := y div 2;
            k := k * 2
      end; {gcd(u,v) = k * gcd(x,y)}
repeat {odd(x) ∨ odd(y)}
       while even(x) do x := x div 2;
       while even(y) do y := y div 2;
       {odd(x) ∧ odd(y)}
       if x > y then x := x − y else y := y − x
until (x = 0) ∨ (y = 0)
if x = 0 then gcd := y * k else gcd := x * k
end {gcd = gcd(u,v)}
```

The “Tie” Connective

$$\alpha \bowtie \beta$$

“α is equivalent to β”

Tim Stokes:
what happens if we add this to PDL ?
Is the logic axiomatizable ? decidable?

The “Tie” Connective

$$\alpha \bowtie \beta$$

“α is equivalent to β”

Tim Stokes:
what happens if we add this to PDL ?
Is the logic axiomatizable ? decidable?

Propositional Dynamic Logic (PDL)

Atomic formulas: p
Atomic programs: π
Formulas: A
Programs: α

$$A ::= p \mid \bot \mid A_1 \to A_2 \mid [\alpha]A$$
$$\alpha ::= \pi \mid \alpha_1; \alpha_2 \mid \alpha_1 \cup \alpha_2 \mid \alpha^* \mid A?$$

$[\alpha]A$ after α, A.
$\alpha_1; \alpha_2$ do α_1 and then α_2 (*composition*).
$\alpha_1 \cup \alpha_2$ do either α_1 or α_2 non-deterministically (*alternation*).
α^* repeat α some finite number (≥ 0) of times (*iteration*).
$A?$ *test* A: continue if A is true, otherwise "fail".

Propositional Dynamic Logic (PDL)

Atomic formulas: p
Atomic programs: π
Formulas: A
Programs: α

$$A ::= p \mid \bot \mid A_1 \to A_2 \mid [\alpha]A$$
$$\alpha ::= \pi \mid \alpha_1; \alpha_2 \mid \alpha_1 \cup \alpha_2 \mid \alpha^* \mid A?$$

$[\alpha]A$	after α, A.
$\alpha_1; \alpha_2$	do α_1 and then α_2 (*composition*).
$\alpha_1 \cup \alpha_2$	do either α_1 or α_2 non-deterministically (*alternation*).
α^*	repeat α some finite number (≥ 0) of times (*iteration*).
$A?$	*test* A: continue if A is true, otherwise "fail".

Some definitions

if A **then** α **else** β	is	$(A?;\alpha) \cup (\neg A?;\beta)$
while A **do** α	is	$(A?;\alpha)^*;\neg A?$
repeat α **until** A	is	$\alpha;(\neg A?;\alpha)^*$
skip	is	$(p \vee \neg p)?$
$\langle\alpha\rangle A$	is	$\neg[\alpha]\neg A$

PDL-Models

$$\mathcal{M} = (S, \{R_\alpha : \alpha \text{ a program}\}, \models),$$

where

- S is a set (states)
- R_α is a binary relation on S (input/output pairs)
- $\models$ is a satisfaction relation between states and formulas.
- $\mathcal{M}, s \models [\alpha]A$ iff for all t, $sR_\alpha t$ implies $\mathcal{M}, t \models A$.
- $\mathcal{M}, s \models \langle\alpha\rangle A$ iff for some t, $sR_\alpha t$ and $\mathcal{M}, t \models A$.
- Standard model conditions:
  - $R_{\alpha;\beta} = R_\alpha \circ R_\beta = \{(s,t) : \exists u(sR_\alpha u \text{ and } uR_\beta t)\}$.
  - $R_{\alpha\cup\beta} = R_\alpha \cup R_\beta$.
  - $R_{\alpha^*} =$ the reflexive-transitive closure of R_α.
  - $R_{A?} = \{(s,s) : \mathcal{M}, s \models A\}$.

Adding $\bowtie$

$$\mathcal{M}, s \models \alpha \bowtie \beta \text{ iff } \text{ for all } t \in S,\ sR_\alpha t \text{ iff } sR_\beta t.$$

Definition

A formula A is valid if $\mathcal{M}, s \models A$ for every s in every $\mathcal{M}$.

Theorem

The set of valid formulas is highly undecidable.

Adding $\bowtie$

$$\mathcal{M}, s \models \alpha \bowtie \beta \text{ iff } \text{ for all } t \in S,\ sR_\alpha t \text{ iff } sR_\beta t.$$

Definition

A formula A is valid if $\mathcal{M}, s \models A$ for every s in every $\mathcal{M}$.

Theorem

The set of valid formulas is highly undecidable.

Adding $\bowtie$

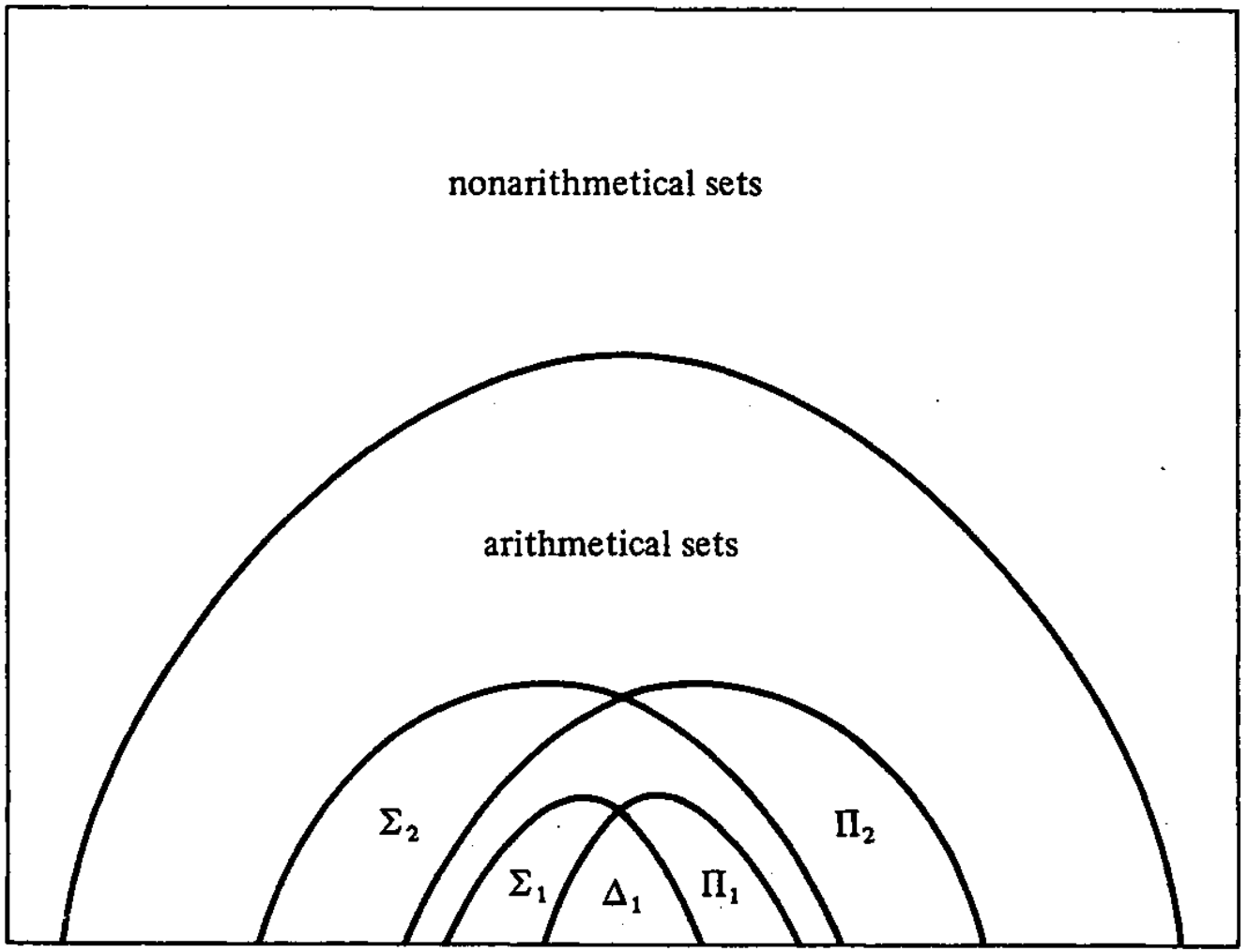
$$\mathcal{M}, s \models \alpha \bowtie \beta \text{ iff } \text{ for all } t \in S,\ sR_\alpha t \text{ iff } sR_\beta t.$$

Definition

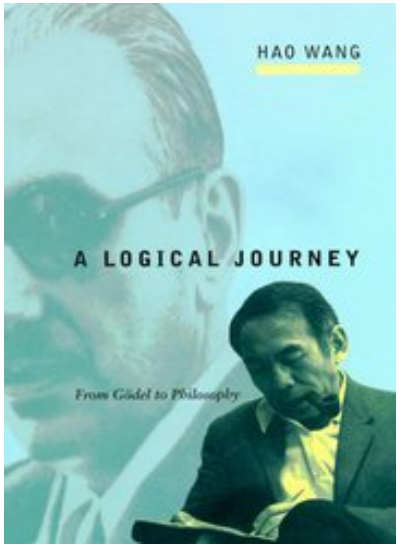
A formula A is valid if $\mathcal{M}, s \models A$ for every s in every $\mathcal{M}$.

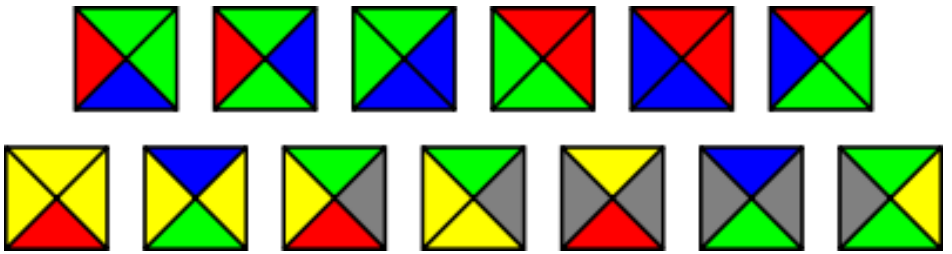
Theorem

The set of valid formulas is highly undecidable.

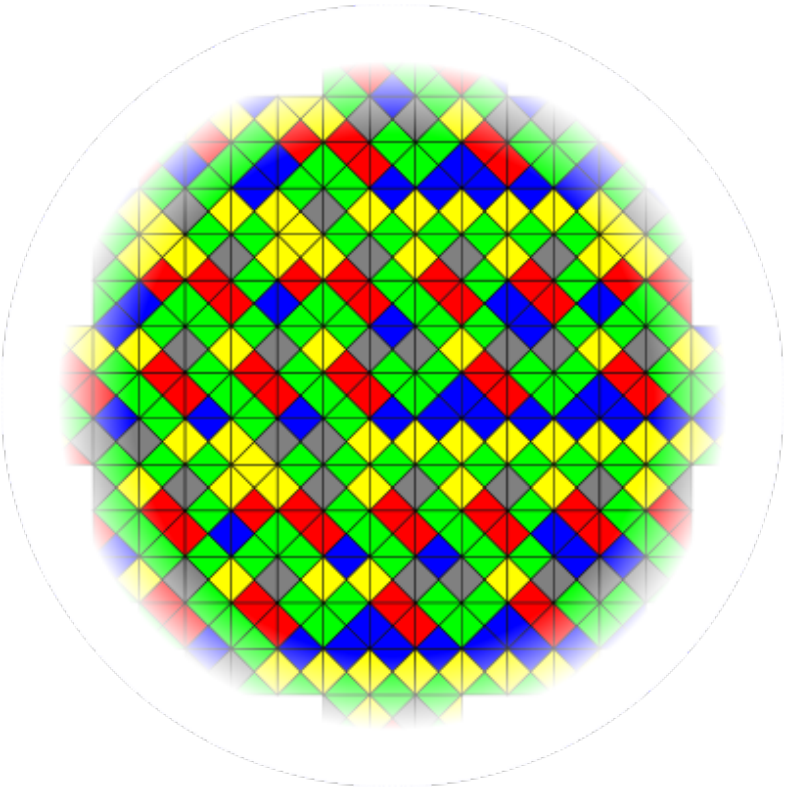

nonarithmetical sets
arithmetical sets
Σ_2
Π_2
Σ_1
Δ_1
Π_1

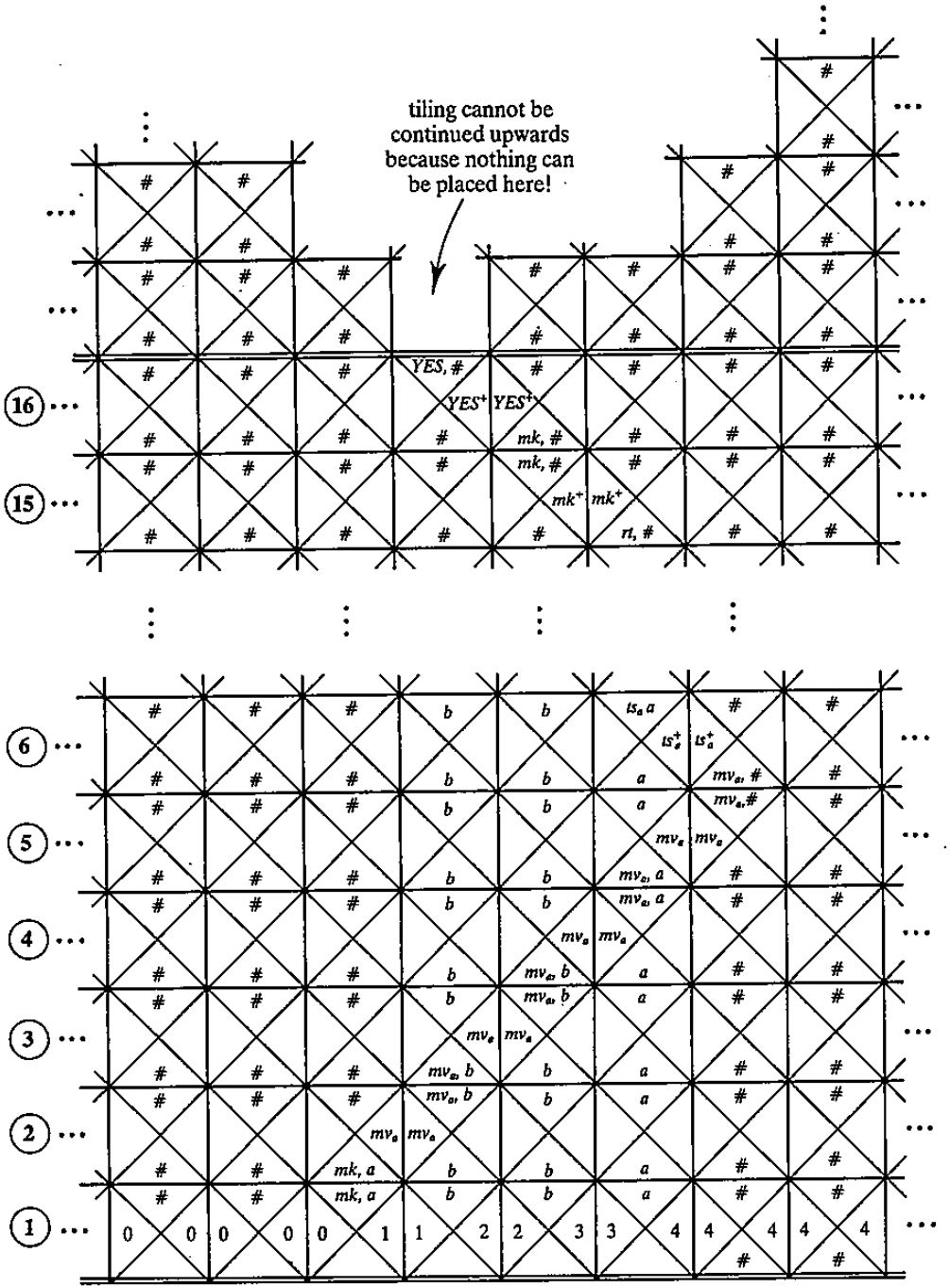
Dominoes (Wang tiles)





Tiling a plane area




tiling cannot be continued upwards because nothing can be placed here!
16
15
6
5
4
3
2
1
0 0 0 0 0 1 1 2 2 3 3 4 4 4 4 4

Tiling – formally:

A domino system $\mathcal{D} = (D, V, H)$ comprises

- a finite set D (dominoes).
- binary relations V (vertical) and H (horizontal) on D.

A tiling of the plane by $\mathcal{D}$ is a function $\tau : \mathbb{Z} \times \mathbb{Z} \to D$ such that

- $\tau(x, y)\, V\, \tau(x + 1, y)$
- $\tau(x, y)\, H\, \tau(x, y + 1)$

Using $\tau : \omega \times \omega \to D$ gives a tiling of the positive quadrant

Tiling – formally:

A domino system $\mathcal{D} = (D, V, H)$ comprises

- a finite set D (dominoes).
- binary relations V (vertical) and H (horizontal) on D.

A tiling of the plane by $\mathcal{D}$ is a function $\tau : \mathbb{Z} \times \mathbb{Z} \to D$ such that

- $\tau(x, y)\, V\, \tau(x + 1, y)$
- $\tau(x, y)\, H\, \tau(x, y + 1)$

Using $\tau : \omega \times \omega \to D$ gives a tiling of the positive quadrant

Tiling – formally:

A domino system $\mathcal{D} = (D, V, H)$ comprises

- a finite set D (dominoes).
- binary relations V (vertical) and H (horizontal) on D.

A tiling of the plane by $\mathcal{D}$ is a function $\tau : \mathbb{Z} \times \mathbb{Z} \to D$ such that

- $\tau(x, y)\, V\, \tau(x + 1, y)$
- $\tau(x, y)\, H\, \tau(x, y + 1)$

Using $\tau : \omega \times \omega \to D$ gives a tiling of the positive quadrant

Recurring Domino Problems (David Harel)

- Given a domino system, can it tile the plane with some designated tile or colour occurring infinitely often?
- Or, can it tile the positive quadrant with a designated tile occurring infinitely often in the first column, or at least once in each column?
- Given a non-deterministic Turing machine started on a blank tape, does it have a computation that re-enters its start state infinitely often ? This is Σ_1^1-complete.

Recurring Domino Problems (David Harel)

- Given a domino system, can it tile the plane with some designated tile or colour occurring infinitely often?
- Or, can it tile the positive quadrant with a designated tile occurring infinitely often in the first column, or at least once in each column?
- Given a non-deterministic Turing machine started on a blank tape, does it have a computation that re-enters its start state infinitely often ? This is Σ_1^1-complete.

Recurring Domino Problems (David Harel)

- Given a domino system, can it tile the plane with some designated tile or colour occurring infinitely often?
- Or, can it tile the positive quadrant with a designated tile occurring infinitely often in the first column, or at least once in each column?
- Given a non-deterministic Turing machine started on a blank tape, does it have a computation that re-enters its start state infinitely often ? This is Σ_1^1-complete.

The highly undecidable problem we use:

Given a domino system $\mathcal{D}$, with a designated subset of "neon" tiles, is there a tiling of the positive quadrant in which

- some specified domino d_0 occurs at the origin, and
- the diagonal contains infinitely many neon tiles?

Theorem

There is a formula $A_{\mathcal{D}}$ of $\mathrm{PDL}{+}\bowtie$ that is satisfiable iff such a tiling exists.

The highly undecidable problem we use:

Given a domino system $\mathcal{D}$, with a designated subset of "neon" tiles, is there a tiling of the positive quadrant in which

- some specified domino d_0 occurs at the origin, and
- the diagonal contains infinitely many neon tiles?

Theorem

There is a formula $A_{\mathcal{D}}$ of $\mathrm{PDL}+\bowtie$ that is satisfiable iff such a tiling exists.

Defining $A_{\mathcal{D}}$

Atomic formulas: the members d of D

Atomic programs: N and E (for North and East)

neon = $\bigvee\{d : d \text{ is neon}\}$

Models: $\mathcal{M} = (S, R_{\mathsf{N}}, R_{\mathsf{E}}, \models)$

for example: $(\omega \times \omega, R_V, R_H, \dots)$

Defining $A_{\mathcal{D}}$

Atomic formulas: the members d of D

Atomic programs: N and E (for North and East)

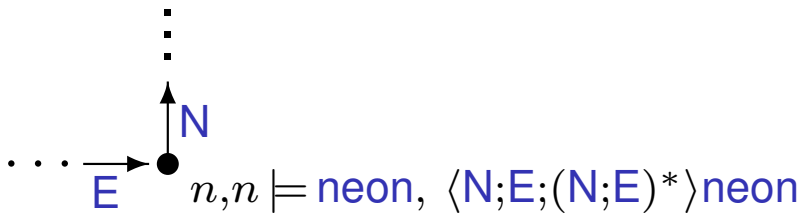
neon = $\bigvee\{d : d \text{ is neon}\}$

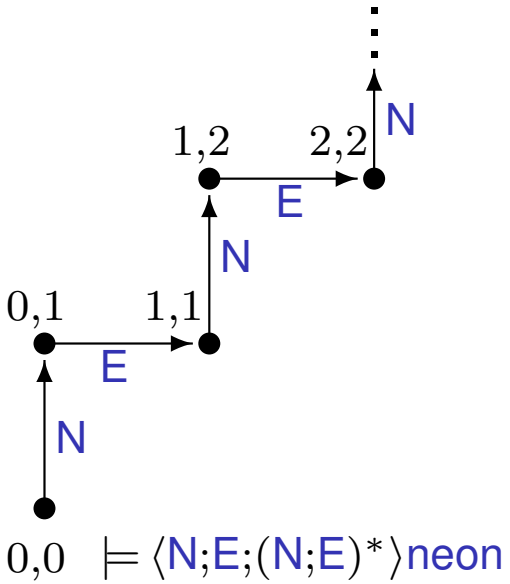
Models: $\mathcal{M} = (S, R_{\mathsf{N}}, R_{\mathsf{E}}, \models)$

for example: $(\omega \times \omega, R_V, R_H, \dots)$

Neon infinitely-often on the diagonal:

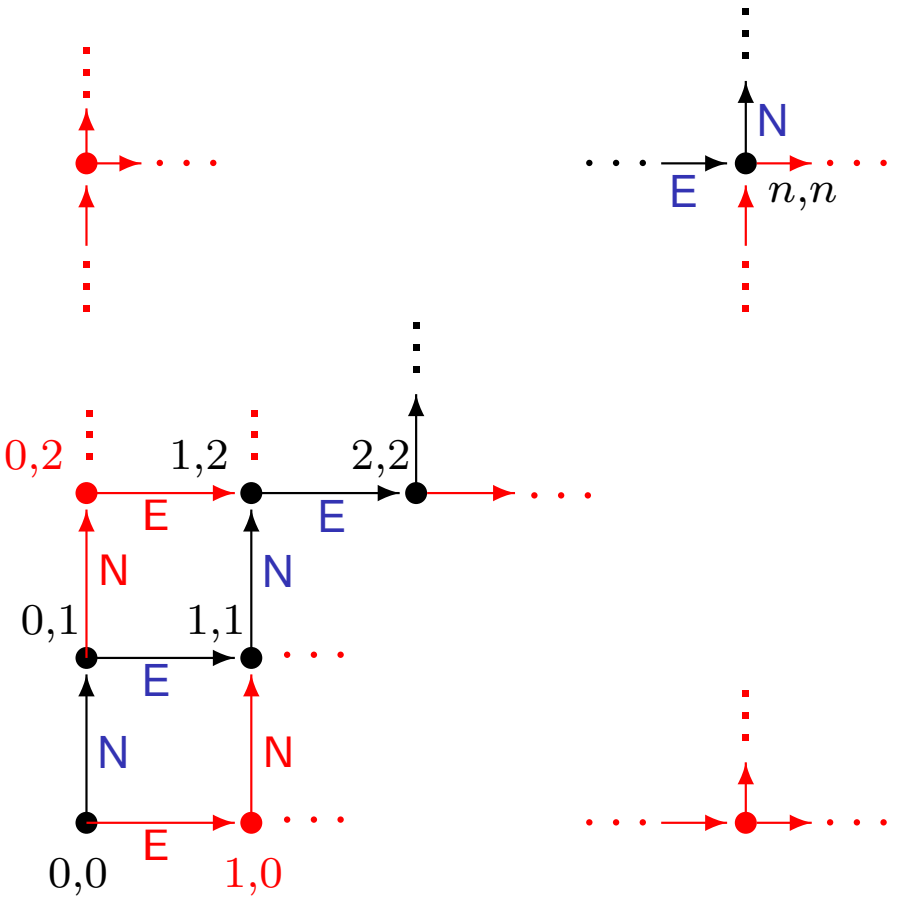
A_{neon} : $[(\mathsf{N};\mathsf{E})^*]\langle \mathsf{N};\mathsf{E};(\mathsf{N};\mathsf{E})^*\rangle \text{neon}$





Using $\bowtie$ to fill out the grid:

A_{grid} : $[\mathsf{N}^*][\mathsf{E}^*](\mathsf{N}; \mathsf{E} \bowtie \mathsf{E}; \mathsf{N})$



Forcing a tiling:

Want

$$\tau(x, y) = d \quad \text{iff} \quad d \text{ is true at the point labelled } x, y.$$

A_{tiling} is the conjunction of

$$\bigvee\{d : d \in D\};$$
$$\neg(d \wedge d'), \qquad \text{for all } d \neq d';$$
$$d \to [\mathsf{N}](\bigvee\{d' : d\, V\, d'\}) \qquad \text{for all } d;$$
$$d \to [\mathsf{E}](\bigvee\{d' : d\, H\, d'\}) \qquad \text{for all } d.$$

$A_{\mathcal{D}}$ is

$$d_0 \wedge A_{\text{neon}} \wedge A_{\text{grid}} \wedge [\mathsf{N}^*][\mathsf{E}^*]A_{\text{tiling}}$$

Forcing a tiling:

Want

$$\tau(x, y) = d \quad \text{iff} \quad d \text{ is true at the point labelled } x, y.$$

A_{tiling} is the conjunction of

$$\bigvee\{d : d \in D\};$$
$$\neg(d \wedge d'), \qquad \text{for all } d \neq d';$$
$$d \to [\mathsf{N}](\bigvee\{d' : d\, V\, d'\}) \qquad \text{for all } d;$$
$$d \to [\mathsf{E}](\bigvee\{d' : d\, H\, d'\}) \qquad \text{for all } d.$$

$A_{\mathcal{D}}$ is

$$d_0 \wedge A_{\text{neon}} \wedge A_{\text{grid}} \wedge [\mathsf{N}^*][\mathsf{E}^*]A_{\text{tiling}}$$

Forcing a tiling:

Want

$$\tau(x, y) = d \quad \text{iff} \quad d \text{ is true at the point labelled } x, y.$$

A_{tiling} is the conjunction of

$$\bigvee\{d : d \in D\};$$
$$\neg(d \wedge d'), \qquad \text{for all } d \neq d';$$
$$d \to [\mathsf{N}](\bigvee\{d' : d\, V\, d'\}) \qquad \text{for all } d;$$
$$d \to [\mathsf{E}](\bigvee\{d' : d\, H\, d'\}) \qquad \text{for all } d.$$

$A_{\mathcal{D}}$ is

$$d_0 \wedge A_{\text{neon}} \wedge A_{\text{grid}} \wedge [\mathsf{N}^*][\mathsf{E}^*]A_{\text{tiling}}$$

$$[\alpha^*]A \quad \text{iff} \quad [\textbf{while } A \textbf{ do } \alpha]\bot$$

High undecidability holds for any variant of PDL that expresses

$$\textbf{while } A \textbf{ do } \alpha$$
$$\alpha; \beta$$
$$\alpha \bowtie \mathbf{skip}$$

regardless of whether atomic programs are deterministic or non-deterministic.

This uses four atomic programs N, E, S, W, with

$$\mathsf{N}; \mathsf{E}; \mathsf{S}; \mathsf{W} \bowtie \mathbf{skip}$$

etc.

$$[\alpha^*]A \quad \text{iff} \quad [\textbf{while } A \textbf{ do } \alpha]\bot$$

High undecidability holds for any variant of PDL that expresses

$$\textbf{while } A \textbf{ do } \alpha$$
$$\alpha; \beta$$
$$\alpha \bowtie \mathbf{skip}$$

regardless of whether atomic programs are deterministic or non-deterministic.

This uses four atomic programs N, E, S, W, with

$$\mathsf{N}; \mathsf{E}; \mathsf{S}; \mathsf{W} \bowtie \mathbf{skip}$$

etc.

$$[\alpha^*]A \quad \text{iff} \quad [\textbf{while } A \textbf{ do } \alpha]\bot$$

High undecidability holds for any variant of PDL that expresses

$$\textbf{while } A \textbf{ do } \alpha$$
$$\alpha; \beta$$
$$\alpha \bowtie \mathbf{skip}$$

regardless of whether atomic programs are deterministic or non-deterministic.

This uses four atomic programs N, E, S, W, with

$$\mathsf{N}; \mathsf{E}; \mathsf{S}; \mathsf{W} \bowtie \mathbf{skip}$$

etc.


That's all Folks!