Basic Research in Computer Science

BRICS RS-99-1 Jurdziński & Nielsen: Hereditary History Preserving Simulation is Undecidable

Hereditary History Preserving Simulation is Undecidable

Marcin Jurdziński
Mogens Nielsen

BRICS Report Series **RS-99-1**

ISSN 0909-0878 **January 1999**

Copyright © 1999, Marcin Jurdziński & Mogens Nielsen
BRICS, Department of Computer Science
University of Aarhus. All rights reserved.

Reproduction of all or part of this work
is permitted for educational or research use
on condition that this copyright notice is
included in any copy.

See back inner page for a list of recent BRICS Report Series publications.
Copies may be obtained by contacting:

BRICS
Department of Computer Science
University of Aarhus
Ny Munkegade, building 540
DK–8000 Aarhus C
Denmark
Telephone: +45 8942 3360
Telefax: +45 8942 3255
Internet: BRICS@brics.dk

BRICS publications are in general accessible through the World Wide Web and anonymous FTP through these URLs:

`http://www.brics.dk`
`ftp://ftp.brics.dk`
This document in subdirectory `RS/99/1/`

Hereditary History Preserving Simulation Is Undecidable

Marcin Jurdziński* Mogens Nielsen*
BRICS†
Department of Computer Science
University of Aarhus

January 1999

Abstract

We show undecidability of *hereditary history preserving simulation* for finite asynchronous transition systems by a reduction from the *halting problem* of deterministic Turing machines. To make the proof more transparent we introduce an intermediate problem of *deciding the winner* in *domino snake games*. First we reduce the halting problem of deterministic Turing machines to domino snake games. Then we show how to model a domino snake game by a *hereditary history simulation game* on a pair of finite *asynchronous transition systems*.

1 Domino snake games

A *tiling system* $\mathcal{D} = (D, V, H)$ consists of a set D of *dominoes*, and two relations: $V \subseteq D^2$, called *vertical compatibility*, and $H \subseteq D^2$, called *horizontal compatibility*. Intuitively, one can think of dominoes as unit squares with coloured sides (and with an orientation, *i.e.*, the dominoes cannot be rotated.) In this metaphor the meaning of the vertical and horizontal compatibility relations can be described as follows: for a pair of dominoes $d, d' \in D$, we have

- $(d, d') \in V$, if the top side of d has the same color as the bottom side of d',
- $(d, d') \in H$, if the right-hand side of d has the same color as the left-hand side of d'.

*Address: BRICS, Department of Computer Science, University of Aarhus, Ny Munkegade, Building 540, 8000 Aarhus C, Denmark. Emails: {mju,mn}@brics.dk.

†**B**asic **R**esearch in **C**omputer **S**cience,
Centre of the Danish National Research Foundation.

Definition 1 (Domino snake game)
Let $\mathcal{D} = (D, V, H)$ be a *tiling system*, and let $\mathbb{N}_+^2$, the positive quadrant of the integer grid, be the set of *locations*. Pairs $(l, d) \in \mathbb{N}_+^2 \times D$ are called *configurations*. Locations $l = (i, j)$ and $l' = (i', j')$ are *neighbouring*, if $|i - i'| + |j - j'| = 1$. Configurations (l, d) and (l', d') are *consistent*, if their locations l and l' are neighbouring, and the adjecent sides have the same color, *e.g.*: if $i' = i + 1$ and $j' = j$, then $(d, d') \in H$; if $i' = i$ and $j' = j - 1$, then $(d', d) \in V$; *etc.*
The *domino snake game* $\Gamma_{\mathrm{ds}}(\mathcal{D})$ is played by two players *Tiler* and *Challenger*.

1. First the players fix an *initial configuration*: Challenger chooses a location $l \in \mathbb{N}_+^2$, and then Tiler responds by picking a domino $d \in D$; the pair (l, d) becomes the *current configuration*.

2. In each *move* of the game the players change the current configuration (l, d): first Challenger chooses a location $l' \in L$ neighbouring with l, and then Tiler responds by picking a domino d', so that configurations (l, d) and (l', d') are consistent.

A *play* is a *maximal* sequence of configurations formed by players making moves in the fashion described above. Challenger *wins* a play if after a finite number of moves Tiler gets stuck, *i.e.*, he cannot complete a move. Otherwise the play is infinite and Tiler is the winner.

Let $d^{\mathrm{init}} \in D$ be a domino. In the *origin constrained domino snake game* $\Gamma_{\mathrm{ds}}(\mathcal{D}, d^{\mathrm{init}})$ Tiler's responses in the "origin" location $(1, 1)$ are restricted only to domino d^{init}; we refer to $d^{\mathrm{init}} \in D$ as the *origin constraint* of the game $\Gamma_{\mathrm{ds}}(\mathcal{D}, d^{\mathrm{init}})$. [Definition 1] □

A strategy for Tiler in $\Gamma_{\mathrm{ds}}(\mathcal{D}, d^{\mathrm{init}})$ is a partial function $\Theta : \mathbb{N}_+^2 \cup (\mathbb{N}_+^2 \times D \times \mathbb{N}_+^2) \rightharpoonup D$, such that:

1. if $\Theta\big((1, 1)\big)$, or $\Theta\big(l, d, (1, 1)\big)$ are defined, then they are equal to d^{init},

2. if $\Theta(l, d, l') = d'$, and locations l and l' are neighbouring, then configurations (l, d) and (l', d') are consistent.

A play $\big\langle (l_0, d_0), (l_1, d_1), (l_2, d_2), \dots \big\rangle$ is consistent with a strategy Θ, if $d_0 = \Theta(l_0)$, and $d_{i+1} = \Theta(l_i, d_i, l_{i+1})$ for all $i \geq 0$. A strategy Θ is *winning* for Tiler if all plays consistent with Θ are infinite, *i.e.*, winning for Tiler. The notion of a (winning) strategy for Challenger can be defined appropriately.
We say that a map $\Theta : \mathbb{N}_+^2 \to \wp(D)$ is *closed* for $\Gamma_{\mathrm{ds}}(\mathcal{D}, d^{\mathrm{init}})$ if:

1. $\Theta(l) \neq \emptyset$ for all $l \in \mathbb{N}_+^2$, and $\Theta\big((1, 1)\big) = \{d^{\mathrm{init}}\}$,

2. if $d \in \Theta(l)$, then for all locations l' neighbouring with l, there is $d' \in \Theta(l')$, so that configurations (l, d) and (l', d') are consistent.

Proposition 2 Tiler has a winning strategy in $\Gamma_{\mathrm{ds}}(\mathcal{D}, d^{\mathrm{init}})$ *if and only if* there is a closed map for $\Gamma_{\mathrm{ds}}(\mathcal{D}, d^{\mathrm{init}})$.

Domino snake games are easily seen to be determined, *i.e.*, either of the two players has a *winning strategy*. The problem of *deciding the winner* in an origin constrained domino snake game is the following: given a tiling system $\mathcal{D} = (D, V, H)$, and an origin constraint $d^{\mathrm{init}} \in D$, decide which of the players has a winning strategy in the game $\Gamma_{\mathrm{ds}}(\mathcal{D}, d^{\mathrm{init}})$.

Theorem 3 (Undecidability of domino snake games)
The problem of deciding the winner in origin constrained *domino snake games is undecidable.*

The proof is a reduction from the halting problem for deterministic inputless Turing machines. For a deterministic inputless Turing machine M we define a tiling system $\mathcal{D}_M = (D_M, V_M, H_M)$, and an origin constraint $d_M \in D_M$, enjoing the following property.

Proposition 4 Machine M halts *if and only if* Challenger has a winning strategy in the origin constrained domino snake game $\Gamma_{\mathrm{ds}}(\mathcal{D}_M, d_M)$.

Let $M = (Q, q^{\mathrm{init}}, q^{\mathrm{halt}}, \Sigma, \delta)$ be a 1-tape, deterministic Turing machine, where Q is the finite set of *states*, $q^{\mathrm{init}} \in Q$ is the *initial state*, $q^{\mathrm{halt}} \in Q$ is the *halting state*, Σ is the *tape alphabet*, and $\delta : Q \times \Sigma \to Q \times \Sigma \times \{-1, 1\}$ is the *transition function*. The semantics is standard:

- the machine starts working at time $t = 1$ and in state q^{init} scanning cell $c = 1$ of an empty tape (containing the special *blank symbol* $\sqcup \in \Sigma$ in all cells),
- if the machine at time t is in state q and scans cell c containing symbol σ, then it writes symbol σ' into cell c, and at time $t + 1$ it is in state q' and scans cell $c + d$, where $\delta(q, \sigma) = (q', \sigma', d)$.

Without loss of generality we may assume that the machine never moves its head to the left of the first cell of the tape.

For technical convenience we assume that the transition function δ of machine M is total; this makes machine M loop forever in fact. We say that machine M *halts* if its computation reaches the halting state in a finite number of steps; otherwise we say that it *loops*. Clearly, the question if a deterministic inputless Turing machine *halts* in this sense is undecidable.

By $\Gamma = (Q \times \Sigma) \cup \Sigma \cup \{\bot\}$ we denote the set of *cell contents* of machine M, extended with a special symbol $\bot \notin \Sigma$. Let $C_M : \mathbb{N}^2 \to \Gamma$ be the *unbounded*

$\perp$				
$\perp$	$(q^{\text{init}}, \sqcup)$	$\sqcup$	$\sqcup$	
$\perp$	q^{init}	$\perp$	$\perp$	$\perp$

Figure 1: The initial fragment of the computation table of an inputless Turing machine.

computation table of M (see Figure 1):

$$C_M(c,t) = \begin{cases} q^{\text{init}} & \text{if } c = 1 \text{ and } t = 0, \\ \perp & \text{if } c = 0, \text{ or } t = 0 \text{ and } c \geq 2, \\ \text{the contents of cell } c \text{ at time } t & \text{otherwise.} \end{cases}$$

It is an easy and standard observation (see [Pap94], page 168) that if M is deterministic, then for $c, t \geq 1$ it holds that

$C_M(c,t)$ is uniquely determined by

- $C_M(c-1,t-1)$, $C_M(c,t-1)$, and $C_M(c+1,t-1)$, if $c \geq 2$, and (1)
- $C_M(c,t-1)$, and $C_M(c+1,t-1)$, if $c = 1$.

Let $\Delta : \Gamma^3 \rightharpoonup \Gamma$ be a (partial) function yielding this unique value. For technical reasons (see the proof of Claim 7) we require that

$$\text{if } \beta = \perp \text{ and } \Delta(\alpha,\beta,\gamma) \text{ is defined, then } \gamma = \perp. \tag{2}$$

The function Δ can be easily computed from the transition function δ of machine M.

We adopt the following notational convention: a lower case Greek letter with a bar denotes a pair of triples of cell contents

$$\bar{\gamma} = (\gamma_{-1}, \gamma_0) \in (\Gamma^3)^2, \text{ where } \gamma_t = (\gamma_{i,-1}, \gamma_{i,0}, \gamma_{i,1}) \in \Gamma^3, \text{ for } i \in \{-1,0\}.$$

We say that $\bar{\gamma} \in (\Gamma^3)^2$ is a *peephole*, if

$$\gamma_{0,0} = \Delta(\gamma_{-1});$$

we denote the set of peepholes by Π. A peephole γ is *halting* if $\gamma_{i,d} = (q^{\text{halt}}, \sigma)$ for some $i \in \{-1,0\}$, $d \in \{-1,0,1\}$, and $\sigma \in \Sigma$, otherwise it is *live*; we denote the set of live peepholes by Λ. The *peephole table* $P_M : \mathbb{N}_+^2 \to \Pi$ of machine M is defined for all $(c,t) \in \mathbb{N}_+^2$ by

$$P_M(c,t) = \bar{\gamma}, \text{ where } \gamma_{i,d} = C_M(c+d, t+i),$$

for $i \in \{-1,0\}$, and $d \in \{-1,0,1\}$, *i.e.*, its value in location (c,t) is a 3×2 fragment of the computation table to the left, to the right, and below (c,t).

Definition 5 (Peephole snake game $\Gamma_{\rm ds}(\mathcal{D}_M, d_M)$)
The *peephole tiling system* $\mathcal{D}_M$ is defined to be the triple (Λ, V_M, H_M), where

- $(\bar{\tau}, \bar{\gamma}) \in V_M$ if $\tau_0 = \gamma_{-1}$, and
- $(\bar{\tau}, \bar{\gamma}) \in H_M$ if $\tau_{i,d+1} = \gamma_{i,d}$ for all $i, d \in \{-1, 0\}$.

In other words, the *dominoes* of the peephole tiling system are the live peepholes, and the *vertical* and *horizontal compatibility* relations ensure, that two peepholes are compatible if their "overlapping" parts coincide. The *origin constraint* $d_M \in D$ is the value of the peephole table for the "origin" location $(1,1)$, *i.e.*, $d_M = \Big((\bot, q^{\rm init}, \bot), (\bot, (q^{\rm init}, \sqcup), \sqcup)\Big)$; see Figure 1. [Definition 5] □

Proof (of Proposition 4)

Tiler wins if M loops: If machine M does not halt then its computation table C_M does not contain the halting state, hence its peephole table P_M contains only live peepholes. Tiler has then a winning strategy consisting in choosing always the value of the peephole table $P_M(c,t)$ for the current location (c,t).

Challenger wins if M halts: We say that a configuration $\big((c,t), \bar{\gamma}\big)$ is *incorrect*, if for some $d \in \{-1,0,1\}$ we have $\gamma_{-1,d} \neq C_M(c+d, t-1)$ and $c+d \geq 1$.

If machine M halts then there is a location (c,t) with $c \geq 1$, so that $C_M(c,t) = (q^{\rm halt}, \sigma)$ for some $\sigma \in \Sigma$. Challenger picks $(c, t+1)$ as the initial location, and so the initial configuration must be incorrect since Tiler can only respond with live peepholes. The following two claims give Challenger a strategy to maintain an incorrect configuration as an invariant, while progressing towards the "origin" location $(1,1)$. This clearly gives a winning strategy for Challenger, because the origin constraint allows only the value of the peephole table $P_M(1,1)$ in location $(1,1)$, which clearly is not incorrect.

Claim 6 If the current configuration $\big((c,t), \bar{\gamma}\big)$ with $t \geq 2$ is incorrect, then Challenger can in no more than two moves either make Tiler stuck, or force the play to an incorrect configuration $\big((c', t-1), \bar{\gamma}'\big)$.

Proof: By the definition of an incorrect configuration we have that $\gamma_{-1,d} \neq C_M(c+d, t-1)$ for some $d \in \{-1,0,1\}$, so that $c+d \geq 1$. Challenger in no more than two moves makes Tiler stuck or forces $\big((c+d, t-1), \bar{\gamma}'\big)$ to be a new configuration. By the definition of V_M and H_M we have $\gamma'_{0,0} = \gamma_{-1,d} \neq C_M(c+d, t-1)$, hence from (1) and the definition of a peephole it follows that the new configuration must be incorrect. [Claim 6] ■

Claim 7 If the current configuration $\big((c,1), \bar{\gamma}\big)$ with $c \geq 2$ is incorrect, then Challenger by moving to location $(c-1, 1)$ either makes Tiler stuck, or forces the new configuration $\big((c-1,1), \bar{\gamma}'\big)$ to be incorrect.

Proof: If $\gamma_{-1,-1} \neq C_M(c-1,0)$, or $\gamma_{-1,0} \neq C_M(c,0)$, then by the definition of H_M the next configuration $\big((c-1,1),\bar{\gamma}'\big)$ must be incorrect. Otherwise $\gamma_{-1,0} = \perp$, hence (2) implies that $\gamma_{-1,1} = \perp$, but this is impossible because $\big((c,1),\bar{\gamma}\big)$ is incorrect. [Claim 7] ■ [Proposition 4] ■

2 Hereditary history preserving simulation

Definition 8 (Labelled asynchronous transition system)
A *labelled asynchronous transition system* is a tuple $A = (S, s^{\text{init}}, E, \rightarrow, L, \lambda, I)$, where S is its set of *states*, $s^{\text{init}} \in S$ is the *initial state*, E is the set of *events*, $\rightarrow \subseteq S \times E \times S$ is the set of *transitions*, L is the set of labels, and $\lambda : E \rightarrow L$ is the *labelling function*, and $I \subseteq E^2$ is the *independence relation* which is irreflexive and symmetric. We often write $s \stackrel{e}{\rightarrow} s'$, instead of $(s,e,s') \in \rightarrow$. Moreover, the following conditions have to be satisfied:

1. if $s \stackrel{e}{\rightarrow} s'$, and $s \stackrel{e}{\rightarrow} s''$, then $s' = s''$,
2. if $(e,e') \in I$, $s \stackrel{e}{\rightarrow} s'$, and $s' \stackrel{e'}{\rightarrow} t$, then $s \stackrel{e'}{\rightarrow} s''$, and $s'' \stackrel{e}{\rightarrow} t$ for some $s'' \in S$.

[Definition 8] □

This definition may seem to be quite liberal, in the sense that it requires an asynchronous transition system to satisfy very few properties related to its independence relation. For example, labelled asynchronous transition systems arising from 1-safe Petri nets form a proper subclass. We want to stress, however, that we have chosen this liberal definition only for technical convenince. In fact, the proof of the undecidability result that follows goes through even for 1-safe Petri nets.

Let $A = (S, s^{\text{init}}, E, \rightarrow, L, \lambda, I)$ be a labelled asynchronous transition system. A sequence of events $\bar{e} = \langle e_1, e_2, \ldots, e_k \rangle \in E^k$ is a *path* in A if there are states $s_1, s_2, \ldots, s_{k+1} \in S$, such that $s_1 = s^{\text{init}}$, and for all $i \in [k]$ we have $t_i = (s_i, e_i, s_{i+1})$ for some $t_i \in T$. We denote the set of paths in A by $\text{Path}(A)$. For every sequence of events $\bar{e} \in E^k$ the independence relation induces an E-labelled partial order $\pi(\bar{e}) = ([k], \trianglelefteq, \varepsilon)$, where $\varepsilon : [k] \rightarrow E$ is the labelling function. For all $i \in [k]$ we set $\varepsilon(i) = e_i$. For $i,j \in [k]$ we define $i \lessdot j$ to hold, if $i \leq j$, and $(e_i, e_j) \notin I$. We get $\trianglelefteq$ as the transitive closure of $\lessdot$.

For $\bar{e}, \bar{e'} \in \text{Path}(A)$ we define $\bar{e} \cong \bar{e'}$ to hold, if the corresponding labelled partial orders $\pi(\bar{e})$ and $\pi(\bar{e'})$ are isomorphic. This is clearly an equivalence relation.

Definition 9 (Unfolding)
Let $A = (S, s^{\text{init}}, E, \rightarrow, L, \lambda, I)$ be a labelled asynchronous transition system. The *unfolding* of A is the labelled asynchronous transition system $U(A) = \big(S_{U(A)}, s^{\text{init}}_{U(A)}, E, \rightarrow_{U(A)}, L, \lambda, I\big)$, defined as follows:

- the set of states $S_{U(A)}$ is defined to be $\mathrm{Path}(A)/_{\cong}$, *i.e.*, the set of ($E$-labelled) partial order behaviours of A,
- the initial state $s^{\mathrm{init}}_{U(A)}$ is $[\epsilon]_{\cong}$, *i.e.*, the empty (E-labelled) partial order,
- we define $\tau \xrightarrow{e}_{U(A)} \tau'$ to hold, if $\bar{e} \in \tau$, and $\bar{e} \cdot e \in \tau'$, for some path $\bar{e} \in \mathrm{Path}(A)$.

Definition 10 (Hereditary history preserving simulation game) Let $A_i = (S_i, s_i, E_i, \rightarrow_i, L, \lambda_i, I_i)$ for $i = 1, 2$ be labelled asynchronous transition systems. The *hereditary history preserving simulation game* $\Gamma_{\mathrm{hhps}}(A_1, A_2)$ is played by two players *Spoiler* and *Simulator* on the arena based on the unfoldings $U(A_1)$, and $U(A_2)$.

A $\tau_i \in S_{U(A_i)}$ can be seen an E_i-labelled partial order $\big(|\tau_i|, \trianglelefteq_i, \varepsilon_i\big)$, where $|\tau_i|$ is the underlying set of τ_i, and $\varepsilon_i : |\tau_i| \to E_i$ is the labelling function. By $\lambda_i(\tau_i)$ we denote the L-labelled partial order $\big(|\tau_i|, \trianglelefteq_i, \lambda_i \circ \varepsilon_i\big)$. The *arena* of $\Gamma_{\mathrm{hhps}}(A_1, A_2)$ is the labelled transition system

$$\big(S_{(A_1,A_2)}, (\emptyset, \emptyset, \emptyset), E_1 \times E_2, \rightarrow_{(A_1,A_2)}\big),$$

defined as follows:

- the set of states $S_{(A_1,A_2)}$ is the set of triples (τ_1, Ξ, τ_2), where $\tau_1 \in S_{U(A_1)}$, $\tau_2 \in S_{U(A_2)}$, and $\Xi : |\tau_1| \to |\tau_2|$ is an isomorphism of L-labelled partial orders $\lambda_1(\tau_1)$ and $\lambda_2(\tau_2)$,
- we define $(\tau_1, \Xi, \tau_2) \xrightarrow{(e_1,e_2)}_{(A_1,A_2)} (\tau_1', \Xi', \tau_2')$ to hold, if $\tau_1 \xrightarrow{e_1}_{U(A_1)} \tau_1'$, and $\tau_2 \xrightarrow{e_2}_{U(A_2)} \tau_2'$, and $\Xi \subseteq \Xi'$, *i.e.*, Ξ' is the unique extension of Ξ obtained by mapping the latest occurrence of the e_1 event in τ_1' to the latest occurrence of the e_2 event in τ_2'.

We call the elements of $S_{(A_1,A_2)}$ *configurations* of the game. The initial state $(\emptyset, \emptyset, \emptyset)$ is the *initial configuration*. In every move the players change the *current configuration* (τ_1, Ξ, τ_2) into the *next configuration* (τ_1', Ξ', τ_2') in one of the following ways.

1. First Spoiler picks an $e_1 \in E_1$ so that $\tau_1' \xrightarrow{e_1}_{U(A_1)} \tau_1$. Then Simulator has to respond with the $e_2 \in E_2$, such that $\tau_2' \xrightarrow{e_2}_{U(A_2)} \tau_2$, and $(\tau_1', \Xi', \tau_2') \xrightarrow{(e_1,e_2)}_{(A_1,A_2)} (\tau_1, \Xi, \tau_2)$.

Note that Simulator has no choice here: his response is uniquely determined as the label of $\Xi(p)$ in τ_2, where $p \in |\tau_1|$ is the latest occurrence of an e_1 event in τ_1.

2. First Spoiler picks an $e_1 \in E_1$ so that $\tau_1 \stackrel{e_1}{\rightarrow}_{U(A_1)} \tau_1'$. Then Simulator has to respond with an $e_2 \in E_2$, such that $\tau_2 \stackrel{e_2}{\rightarrow}_{U(A_2)} \tau_2'$, and $(\tau_1, \Xi, \tau_2) \stackrel{(e_1,e_2)}{\longrightarrow}_{(A_1,A_2)} (\tau_1', \Xi', \tau_2')$.

A play is a *maximal* sequence of configurations formed by players making moves in the fashion described above. Spoiler *wins* a play if after a finite number of moves Simulator gets stuck, *i.e.*, he cannot complete a move. Otherwise the play is infinite and Simulator is the winner. [Definition 10] □

Note that if $C \stackrel{(e_1,e_2)}{\longrightarrow}_{(A_1,A_2)} C'$, then configuration C' is uniquely determined by C, e_1, and e_2; and vice versa, configuration C is uniquely determined from C', e_1, and e_2. Based on this we define partial operations $\oplus : S_{(A_1,A_2)} \times (E_1 \times E_2) \rightharpoonup S_{(A_1,A_2)}$, and $\ominus : S_{(A_1,A_2)} \times (E_1 \times E_2) \rightharpoonup S_{(A_1,A_2)}$,

- $C \oplus (e_1, e_2) = C'$, if $C \stackrel{(e_1,e_2)}{\longrightarrow}_{(A_1,A_2)} C'$,
- $C \ominus (e_1, e_2) = C'$, if $C' \stackrel{(e_1,e_2)}{\longrightarrow}_{(A_1,A_2)} C$.

A *strategy* for Simulator in $\Gamma_{\mathrm{hhps}}(A_1, A_2)$ is a function $\Sigma : S_{(A_1,A_2)} \times E_1 \rightarrow \wp(E_2)$, such that if $e_2 \in \Sigma(C, e_1)$, then $C \oplus (e_1, e_2)$ is defined. Let $C_i = (\tau_1^i, \Xi^i, \tau_2^i)$ for $i \geq 0$ be configurations of $\Gamma_{\mathrm{hhps}}(A_1, A_2)$. A play $\langle C_0, C_1, C_2, \ldots \rangle$ is *consistent* with a strategy Σ, if for all $i \geq 0$ such that $C_{i+1} = C_i \oplus (e_1, e_2)$ (*i.e.*, the i-th move is of type 2.), we have $e_2 \in \Sigma(C_i, e_1)$. A strategy Σ is *winning* for Simulator if all plays consistent with Σ are infinite, *i.e.*, winning for Simulator.

We say that a strategy $\Sigma : S_{(A_1,A_2)} \times E_1 \rightarrow \wp(E_2)$ is *defined* in a configuration (τ_1, Ξ, τ_2), if $\Sigma\big((\tau_1, \Xi, \tau_2), e_1\big) \neq \emptyset$ for all $e_1 \in E_1$, such that $\tau_1 \stackrel{e_1}{\rightarrow}_{U(A_1)} \tau_1'$. We say that a strategy Σ is *closed* if it satisfies the following conditions:

1. Σ is defined in the initial configuration $(\emptyset, \emptyset, \emptyset)$,
2. if Σ is defined in C, and $C \oplus (e_1, e_2)$ is defined, then Σ is defined in $C \oplus (e_1, e_2)$,
3. if Σ is defined in C, and $C \ominus (e_1, e_2)$ is defined, then Σ is defined in $C \ominus (e_1, e_2)$.

The following follows easily from the definitions.

Proposition 11 Every closed strategy is winning for Simulator. Moreover, if Simulator has a winning strategy then he also has a closed one.

Theorem 12 (Undecidability of hhp-simulation)
Hereditary history preserving simulation for finite labelled asynchronous transition systems is undecidable.

The proof is a reduction from the problem of deciding the winner in origin constrained domino snake games. The method is essentially due to Madhusudan and Thiagarajan [MT98]; we use a slightly modified version of a gadget invented by them.

Let $\Gamma_{\mathrm{ds}}(\mathcal{D}, d^{\mathrm{init}})$ be an origin constrained domino snake game, where $\mathcal{D} = (D, V, H)$ is a tiling system, and $d^{\mathrm{init}} \in D$ is an origin constraint. We define a pair of finite labelled asynchronous transition systems: A_{C} "modelling" Challenger, and A_{T} "modelling" Tiler, so that the following property holds.

Proposition 13 Simulator has a winning strategy in $\Gamma_{\mathrm{hhps}}(A_{\mathrm{C}}, A_{\mathrm{T}})$ *if and only if* Tiler has a winning strategy in the origin constrained domino snake game $\Gamma_{\mathrm{ds}}(\mathcal{D}, d^{\mathrm{init}})$. □

Labelled asynchronous transition system

$A_{\mathrm{C}} = (S_{\mathrm{C}}, s_{\mathrm{C}}^{\mathrm{init}}, E_{\mathrm{C}}, \rightarrow_{\mathrm{C}}, L_{\mathrm{C}}, \lambda_{\mathrm{C}}, I_{\mathrm{C}})$

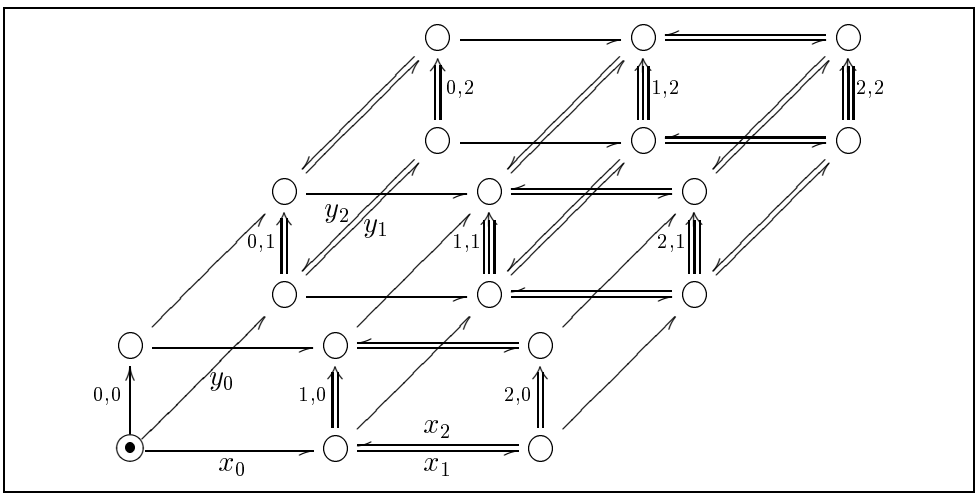
The states, events, and transitions of A_{C} can be read from Figure 2; we briefly explain below how to do it. The initial state is denoted by a solid circle (see Figure 2(a)). The set of events of A_{C} is defined as:

$$E_{\mathrm{C}} = \{\, x_i, y_i \,:\, i \in \{0,1,2\} \,\} \cup \{\, ?_{ij}, ¿_{ij} \,:\, i,j \in \{0,1,2\} \,\}.$$

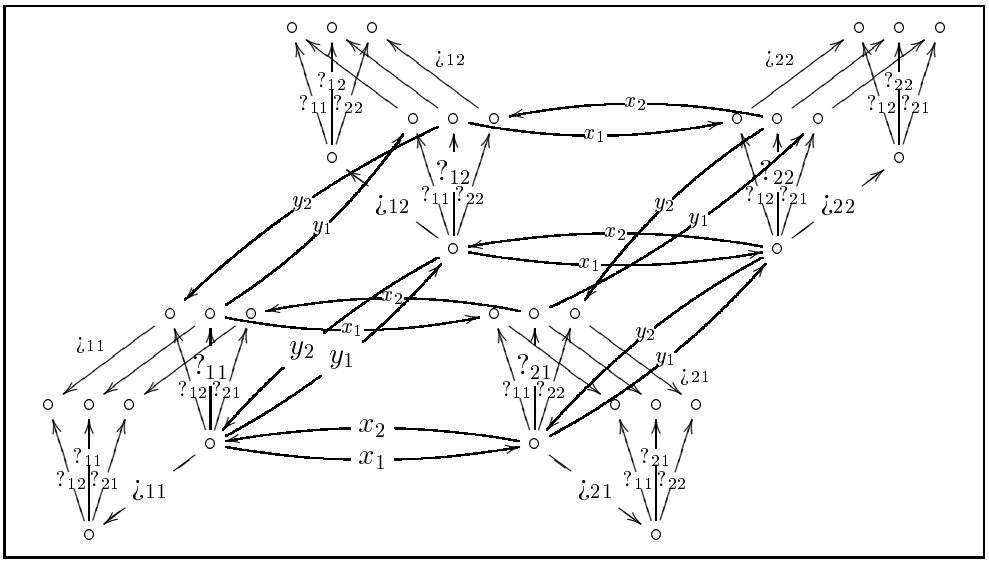
In Figure 2(b) we provide the detailed view of the upper-right cube of the schematic picture of A_{C} from Figure 2(a). The Reader can reconstruct the details of the remaining three cubes in Figure 2(a) by taking appropriately prunned and relabelled copies of the graph in Figure 2(b). For example, there are only two "question" events (instead of four as typical in Figure 2(b)), sticking out from the initial state, namely $¿_{00}$, and $?_{00}$. From a state which is at the bottom of a "double" vertical arrow labelled with "$i, 0$" for $i \in \{1,2\}$, there are three events sticking out, namely: $¿_{i0}$, $?_{i0}$, and $?_{(i\ominus 1)0}$, where $(i \ominus 1) \stackrel{\mathrm{df}}{=} 2 - (i \bmod 2)$. Similarly, from a state which is at the bottom of a "double" vertical arrow labelled with "$0, j$" for $j \in \{1,2\}$, there are three events sticking out, namely: $¿_{0j}$, $?_{0j}$, and $?_{0(j\ominus 1)}$. Finally, we obtain the whole transition graph of A_{C} by gluing together the appropriate faces of the four cubes. We have decided not to draw the whole picture in detail as it would be hard to digest. For example, as the result of the merge, there are six "question" events sticking out of the state at the bottom of the "triple" arrow labelled with "$1, 1$", namely: $¿_{11}$, $?_{11}$, $?_{01}$, $?_{10}$, $?_{21}$, and $?_{12}$.

As the set of labels L_{C} we take the set of events E_{C}; we set the identity function on E_{C} as the labelling λ_{C}. As the independence relation I_{C} we take the symmetric closure of the set:

$$\begin{array}{l} \{\, (x_i, y_j), (x_i, ?_{ij}), (y_j, ?_{ij}) \,:\, i,j \in \{0,1,2\} \,\} \;\cup \\ \{\, (?_{ij}, ¿_{ij}), (?_{ij}, ¿_{(i\oplus 1)j}), (?_{ij}, ¿_{i(j\oplus 1)}) \,:\, i,j \in \{0,1,2\} \,\}, \end{array} \tag{3}$$



(a) The structure of A_{C} and A_{T} in the large.



(b) The structure of the upper-right cube of A_{C} in detail.

Figure 2: The structure of asynchronous transition systems A_{C} and A_{T}.

where $(i \oplus 1) \stackrel{\text{df}}{=} 2 - \big((i+1) \bmod 2\big)$. Note, that this implies that all the "diamonds" in the transition graph of A_{C} are in fact independence squares.

Labelled asynchronous transition system

$A_{\mathrm{T}} = (S_{\mathrm{T}}, s_{\mathrm{T}}^{\mathrm{init}}, E_{\mathrm{T}}, \rightarrow_{\mathrm{T}}, L_{\mathrm{T}}, \lambda_{\mathrm{T}}, I_{\mathrm{T}})$

The overall structure of the transition graph of A_{T} is very similar to A_{C}. Indeed, Figure 2(a) serves as its schematic picture for both A_{C} and A_{T}. The set of events of A_{T} is defined as:

$$\begin{aligned} E_{\mathrm{T}} \;&=\; \{\, x_i, y_i \;:\; i = 0,1,2 \,\} \\ &\cup\; \big\{\, !_{ij}^{d}, ¡_{ij}^{d} \;:\; i,j \in \{0,1,2\}, d \in D, \text{ such that } (i,j) \neq (0,0), \text{ or } d = d^{\mathrm{init}} \,\big\}. \end{aligned}$$

The notable difference with A_{C} is that every "question" event is replaced by $|D|$ copies of a corresponding "answer" events, one for each element d of the set of dominoes D. As for the transition graph of A_{T}, it very closely mimicks the transition graph of A_{C}; every "question" event transition in A_{C} has its $|D|$ "answer" counterparts in A_{T}. The only exception is the initial state of A_{T}, from which only two events stick out, namely: $¡_{00}^{d^{\mathrm{init}}}$, and $!_{00}^{d^{\mathrm{init}}}$. This is how the origin constraint of the domino snake game $(\mathcal{D}, d^{\mathrm{init}})$ is encoded in A_{T}.

As the set of labels L_{T} we take again the set E_{C} of events of A_{C}. The labelling function maps the "answer" events to their "question" counterparts:

$$\lambda_{\mathrm{T}}(e) = \begin{cases} e & \text{if } e \in \{\, x_i, y_i \;:\; i \in \{0,1,2\} \,\}, \\ ?_{ij} & \text{if } e = !_{ij}^{d}, \text{ for some } d \in D, \\ ¿_{ij} & \text{if } e = ¡_{ij}^{d}, \text{ for some } d \in D. \end{cases}$$

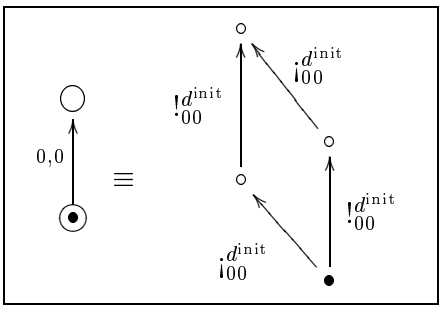
The independence relation I_{T} is defined as the symmetric closure of the set:

$$\begin{aligned} &\big\{\, (x_i, y_j), (x_i, !_{ij}^{d}), (y_j, !_{ij}^{d}) \;:\; i,j \in \{0,1,2\}, \text{ and } d \in D \,\big\} \;\cup \\ &\big\{\, (!_{ij}^{d}, ¡_{ij}^{d}) \;:\; i,j \in \{0,1,2\}, \text{ and } d \in D \,\big\} \;\cup \\ &\big\{\, (!_{ij}^{b}, ¡_{i(j\oplus 1)}^{d}) \;:\; i,j \in \{0,1,2\},\ b,d \in D, \text{ and } (b,d) \in V \,\big\} \;\cup \\ &\big\{\, (!_{ij}^{c}, ¡_{(i\oplus 1)j}^{d}) \;:\; i,j \in \{0,1,2\},\ c,d \in D, \text{ and } (c,d) \in H \,\big\}. \end{aligned} \tag{4}$$

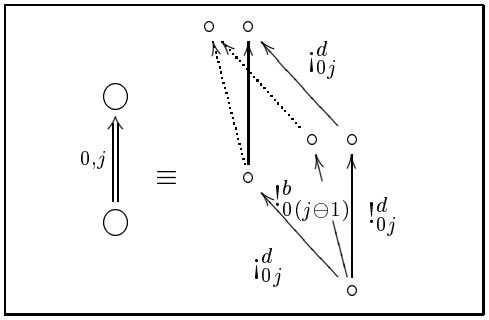
Note how the vertical, and horizontal compatibility relations V, and H of the tiling system $\mathcal{D}$ are encoded in the independence relation I_{T} of the asynchronous transition system A_{T}. Figures 3(a–c) contain some close-ups of the fine structure of the transition graph of A_{T}. We adopt the convention that the dotted arrows in Figures 3(b–c) exist if and only if the corresponding events are independent according to I_{T}.

Proof (of Proposition 13)

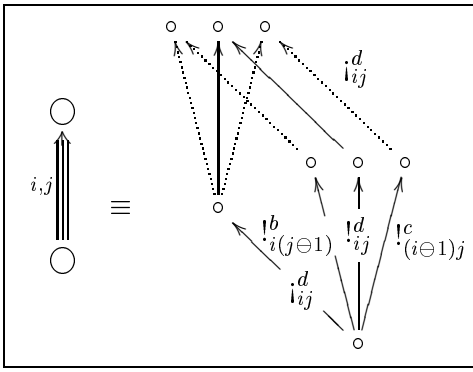
The idea of the proof is to show that winning strategies for Simulator in



(a) The single arrow "$0, 0$".



(b) A double arrow "$0, j$", for $j \in \{1, 2\}$.



Variables b, c, and d range the set of dominoes D.

$$(!^{d}_{ij}, \downarrow^{d}_{ij}) \in I_{\mathrm{T}} \quad \text{for all } d \in D$$

$$(!^{b}_{i(j\ominus 1)}, \downarrow^{d}_{ij}) \in I_{\mathrm{T}} \quad \text{iff} \quad (b, d) \in V$$
$$(!^{c}_{(i\ominus 1)j}, \downarrow^{d}_{ij}) \in I_{\mathrm{T}} \quad \text{iff} \quad (c, d) \in H$$

where $(i \ominus 1) \stackrel{\text{df}}{=} (i \bmod 2) + 1$.

(c) A triple arrow "i, j", seen as a part of the upper-right cube in Figure 2(a).

Figure 3: Details of the asynchronous transitions system A_{T}.

$\Gamma_{\mathrm{hhps}}(A_{\mathrm{C}}, A_{\mathrm{T}})$, and winning strategies for Tiler in $\Gamma_{\mathrm{ds}}(\mathcal{D}, d^{\mathrm{init}})$ can be mutually simulated. This idea is captured in the two translations below, one yielding a closed map for $\Gamma_{\mathrm{ds}}(\mathcal{D}, d^{\mathrm{init}})$, given a closed strategy for Simulator in $\Gamma_{\mathrm{hhps}}(A_{\mathrm{C}}, A_{\mathrm{T}})$, and the other yielding a closed strategy for Simulator in $\Gamma_{\mathrm{hhps}}(A_{\mathrm{C}}, A_{\mathrm{T}})$, given a closed map for $\Gamma_{\mathrm{ds}}(\mathcal{D}, d^{\mathrm{init}})$. By Propositions 11 and 2, and by determinacy of both domino snake, and hereditary history preserving simulation games, these translations suffice to establish Proposition 13.

By $E_?$ we denote the set $\{\ ?_{ij}, ¿_{ij}\ :\ i,j \in \{0,1,2\}\ \}$, and by $E_!$ the set $\{\ !^d_{ij}, ¡^d_{ij}\ :\ i,j \in \{0,1,2\},$ and $d \in D\ \}$. Let $\pi_T = (|\pi_T|, \trianglelefteq_T, \varepsilon_T)$ be a state of $U(A_{\mathrm{T}})$; we write

- $\#_x(\pi_T)$ for $\left|\varepsilon_T^{-1}\big(\{x_0, x_1, x_2\}\big)\right|$,
- $\#_y(\pi_T)$ for $\left|\varepsilon_T^{-1}\big(\{y_0, y_1, y_2\}\big)\right|$.

It is not hard to see that if (π_C, Ξ, π_T) is a configuration of $\Gamma_{\mathrm{hhps}}(A_{\mathrm{C}}, A_{\mathrm{T}})$, then

1. π_C is uniquely determined by π_T, (in fact, π_C is isomorphic to $\lambda_{\mathrm{T}}(\pi_T)$),
2. π_T is uniquely determined by $\#_x(\pi)$, $\#_y(\pi)$, and by $\varepsilon_T \circ \varepsilon_T^{-1}(E_!) \subseteq E_!$,
3. $\left|\varepsilon_T^{-1}(E_!)\right| = \left|\varepsilon \circ \varepsilon_T^{-1}(E_!)\right| \leq 2$.

Properties 1. and 2. amount to saying that the number of occurrences of "x" events, the number of occurrences of "y" events, and the set of occurrences of "!" events in the E_{T}-labelled partial order $\pi_T \in S_{U(A_{\mathrm{T}})}$, uniquely determine the configuration $(\pi_C, \Xi, \pi_T) \in S_{(A_{\mathrm{C}}, A_{\mathrm{T}})}$. Property 3. implies that if $\left|\varepsilon_T^{-1}(E_!)\right| = 2$, then (π_C, Ξ, π_T) is a maximal configuration. It follows that a strategy for Simulator in $\Gamma_{\mathrm{hhps}}(A_{\mathrm{C}}, A_{\mathrm{T}})$ can be represented as a function:

$$\Omega : \mathbb{N}_+^2 \times \big(\{\emptyset\} \cup E_!\big) \times E_? \to \wp(E_!).$$

For notational convenience, if $l = (m, n) \in \mathbb{N}_+^2$, then we write $?_l$ to denote $?_{ij}$, and $¿_l$ to denote $¿_{ij}$, where $i = (m \bmod 2)$, and $j = (n \bmod 2)$. Similarly, we write $!^d_l$ for $!^d_{ij}$, and $¡^d_l$ for $¡^d_{ij}$, where $i = 2 - (m \bmod 2)$, and $j = 2 - (n \bmod 2)$.

Tiler wins $\Gamma_{\mathrm{ds}}(\mathcal{D}, d^{\mathrm{init}})$ if Simulator wins $\Gamma_{\mathrm{hhps}}(A_{\mathrm{C}}, A_{\mathrm{T}})$:
Suppose that Simulator has a closed strategy $\Omega : \mathbb{N}_+^2 \times \big(\{\emptyset\} \cup E_!\big) \times E_? \to \wp(E_!)$ in $\Gamma_{\mathrm{hhps}}(A_{\mathrm{C}}, A_{\mathrm{T}})$. We define a map $\Theta : \mathbb{N}_+^2 \to \wp(D)$ in the following way:

$$\Theta(l) = \{\ d \in D\ :\ !^d_l \in \Omega(l, \emptyset, ?_l)\ \}$$

for all $l \in \mathbb{N}_+^2$. It can be verified that Θ is a closed map for $\Gamma_{\mathrm{ds}}(\mathcal{D}, d^{\mathrm{init}})$.

Simulator wins $\Gamma_{\mathrm{hhps}}(A_{\mathrm{C}}, A_{\mathrm{T}})$ if Tiler wins $\Gamma_{\mathrm{ds}}(\mathcal{D}, d^{\mathrm{init}})$:
Suppose that there is a closed map $\Theta : \mathbb{N}_+^2 \to \wp(D)$ for $\Gamma_{\mathrm{ds}}(\mathcal{D}, d^{\mathrm{init}})$. We define a strategy $\Omega : \mathbb{N}_+^2 \times \big(\{\emptyset\} \cup E_!\big) \times E_? \to \wp(E_!)$ for Simulator in the following way:

- $\Omega\big(l, \emptyset, ?_l\big) = \big\{\ !_l^d\ :\ d \in \Theta(l)\ \big\}$, and $\Omega\big(l, \emptyset, ¿_l\big) = \big\{\ ¡_l^d\ :\ d \in \Theta(l)\ \big\}$,
- $\Omega\big(l, \emptyset, ?_{(l-\delta)}\big) = \big\{\ !_{(l-\delta)}^d\ :\ d \in \Theta(l-\delta)\ \big\}$,
- $\Omega\big(l, ¡_l^d, ?_{(l-\delta)}\big) = \big\{\ !_{(l-\delta)}^d\ :\ d \in \Theta(l-\delta)\ \big\}$, and $\Omega\big(l, !_{l-\delta}^d, ¿_l\big) = \big\{\ ¡_l^d\ :\ d \in \Theta(l)\ \big\}$.

for $l \in \mathbb{N}_+^2$, and $\delta \in \big\{(0,1),(1,0)\big\}$. We skip the verification that Ω indeed gives rise to a closed strategy for Simulator in $\Gamma_{\mathrm{hhps}}(A_{\mathrm{C}}, A_{\mathrm{T}})$. [Proposition 13] ■

The definition and some results on the hereditary history preserving bisimulation can be found in [Bed91, NC94, JNW96].

Definition 14 (Hereditary history preserving bisimulation game)
The *hereditary history preserving bisimulation game* $\Gamma_{\mathrm{hhpb}}(A_1, A_2)$ is played by two players: *Spoiler* and *Bisimulator*. The only differences with respect to the hereditary history preserving *simulation* game $\Gamma_{\mathrm{hhps}}(A_1, A_2)$ are that Simulator is replaced by Bisimulator, and an extra kind of move is allowed.

3. First Spoiler picks an $e_2 \in E_2$ so that $\tau_2 \stackrel{e_2}{\rightarrow}_{U(A_2)} \tau_2'$. Then Simulator has to respond with an $e_1 \in E_1$, such that $\tau_1 \stackrel{e_1}{\rightarrow}_{U(A_1)} \tau_1'$, and $(\tau_1, \Xi, \tau_2) \stackrel{(e_1,e_2)}{\longrightarrow}_{(A_1,A_2)} (\tau_1', \Xi', \tau_2')$.

We get the (plain) *history preserving bisimulation game* [RT88, vGG89] by allowing only the use of moves of type 2. and 3. [Definition 14] □

Plain history preserving bisimulation is known to be decidable [Vog91, JM96].

Problem 1
Is the problem of deciding the winner in hereditary history preserving bisimulation games decidable?

References

[Bed91] Marek A. Bednarczyk. Hereditary history preserving bisimulations or what is the power of the future perfect in program logics. Technical report, Instytut Podstaw Informatyki PAN, filia w Gdańsku, April 1991. Available electronically at `http://www.ipipan.gda.pl/~marek`.

[JM96] Lalita Jategaonkar and Albert R. Meyer. Deciding true concurrency equivalences on safe, finite nets. *Theoretical Computer Science*, 154:107–143, 1996.

[JNW96] André Joyal, Mogens Nielsen, and Glynn Winskel. Bisimulation from open maps. *Information and Computation*, 127(2):164–185, 1996.

[MT98] P. Madhusudan and P. S. Thiagarajan. Controllers for discrete event systems via morphisms. In Davide Sangiorgi and Robert de Simone, editors, *CONCUR'98, Concurrency Theory, 9th International Conference, Proceedings*, volume 1466 of *LNCS*, pages 18–33, Nice, France, September 1998. Springer.

[NC94] Mogens Nielsen and Christian Clausen. Bisimulations, games and logic. Technical Report RS-94-6, Basic Research in Computer Science, Department of Computer Science, University of Aarhus, April 1994.

[Pap94] Christos H. Papadimitriou. *Computational Complexity*. Addison Wesley, 1994.

[RT88] A. Rabinovich and B. Trakhtenbrot. Behaviour structures and nets of processes. *Fundamenta Informaticae*, 11:357–404, 1988.

[vGG89] Rob van Glabbek and Ursula Goltz. Equivalence notions for concurrent sysmtes and refinement of actions (Extended abstract). In A. Kreczmar and G. Mirkowska, editors, *Mathematical Foundations of Computer Science 1989*, volume 379 of *LNCS*, pages 237–248, Porąbka-Kozubnik, Poland, August/September 1989. Springer-Verlag.

[Vog91] Walter Vogler. Deciding history preserving bisimilarity. In Javier Leach Albert, Burkhard Monien, and Mario Rodríguez-Artalejo, editors, *Auotamata, Languages and Programming, 18th International Colloqium, ICALP'91*, volume 510 of *LNCS*, pages 493–505, Madrid, Spain, 8–12 July 1991. Springer-Verlag.

Recent BRICS Report Series Publications

RS-99-1 **Hereditary History Preserving Simulation is Undecidable.** ***Nielsen, Mogens and Jurdziński, Marcin*****. January 1999. 15 pp.**

RS-98-55 **Andrew D. Gordon, Paul D. Hankin, and Søren B. Lassen.** ***Compilation and Equivalence of Imperative Objects (Revised Report)*****. December 1998. iv+75 pp. This is a revision of Technical Report 429, University of Cambridge Computer Laboratory, June 1997, and the earlier BRICS report RS-97-19, July 1997. Appears in Ramesh and Sivakumar, editors,** ***Foundations of Software Technology and Theoretical Computer Science: 17th Conference*****, FST&TCS '97 Proceedings, LNCS 1346, 1997, pages 74–87.**

RS-98-54 **Olivier Danvy and Ulrik P. Schultz.** ***Lambda-Dropping: Transforming Recursive Equations into Programs with Block Structure*****. December 1998. 55 pp. To appear in** ***Theoretical Computer Science*****.**

RS-98-53 **Julian C. Bradfield.** ***Fixpoint Alternation: Arithmetic, Transition Systems, and the Binary Tree*****. December 1998. 20 pp.**

RS-98-52 **Josva Kleist and Davide Sangiorgi.** ***Imperative Objects and Mobile Processes*****. December 1998. 22 pp. Appears in Gries and de Roever, editors,** ***IFIP Working Conference on Programming Concepts and Methods*****, PROCOMET '98 Proceedings, 1998, pages 285–303.**

RS-98-51 **Peter Krogsgaard Jensen.** ***Automated Modeling of Real-Time Implementation*****. December 1998. 9 pp. Appears in** ***The 13th IEEE Conference on Automated Software Engineering*****, ASE '98 Doctoral Symposium Proceedings, 1998, pages 17–20.**

RS-98-50 **Luca Aceto and Anna Ingólfsdóttir.** ***Testing Hennessy-Milner Logic with Recursion*****. December 1998. 15 pp. Appears in Thomas, editor,** ***Foundations of Software Science and Computation Structures: Second International Conference*****, FoSSaCS '99 Proceedings, LNCS 1578, 1999, pages 41–55.**