On a Semantic Definition of Data Independence*

Ranko Lazić[1]** and David Nowak[2]

[1] Department of Computer Science, University of Warwick, UK
Ranko.Lazic@dcs.warwick.ac.uk
[2] LSV, CNRS & ENS Cachan, France
David.Nowak@lsv.ens-cachan.fr

Abstract. A variety of results which enable model checking of important classes of infinite-state systems are based on exploiting the property of data independence. The literature contains a number of definitions of variants of data independence which are by syntactic restrictions in particular formalisms. More recently, data independence was defined for labelled transition systems using logical relations, enabling results about data independent systems to be proved without reference to a particular syntax. In this paper, we show that the semantic definition is sufficiently strong for this purpose. More precisely, it was known that any syntactically data independent symbolic LTS denotes a semantically data independent family of LTSs, but here we show that the converse also holds.
Keywords: data independence, definability, logical relations, nondeterminism

1 Introduction

Informally, a system is data independent with respect to a data type X when, apart from input, output and storage, the only operation that is performed on values of type X is testing a pair of them for equality. The stronger variant of data independence where equality on X is not available is also studied in the literature, as are weaker variants such as allowing constants of type X and unary predicates on X.

A variety of results which enable model checking [5] of important classes of infinite-state systems are based on exploiting data independence (e.g. [23, 12, 10, 20, 6, 14, 22, 19]). For most of those results, although their proofs are in terms of semantics, they are based on definitions of data independence which are by means of syntactic restrictions in particular formalisms.

The first semantic definition of data independence which accommodates the variants with or without equality, constants and predicates was given in [16].

* We acknowledge support from the EPSRC Standard Research Grant 'Exploiting Data Independence', GR/M32900. A part of this research was done at the Oxford University Computing Laboratory.

** Also affiliated to the Mathematical Institute, Belgrade. This author was supported in part by a grant from the Intel Corporation, a Junior Research Fellowship from Christ Church, Oxford, and previously by a scholarship from Hajrija & Boris Vukobrat and Copechim France SA.

The semantic entities used are families of labelled transition systems (LTSs), which consist of an LTS per instantiation of a signature. A signature is a set of type variables and a term context. Logical relations [21] are used to define when a family of LTSs is parametric, and the definition of data independence is the special case when the term context of the signature consists of only equality predicates, constants and unary predicates. It is shown in [16] that the semantics of any syntactically data independent UNITY program [4] is a data independent family of LTSs. The same paper also proves a theorem based on the semantic definition which enables the problem of model checking a data independent system for all instantiations of X to be reduced to model checking for a finite number of finite instantiations. Since it is proved from the semantic definition, the theorem applies to any formalism which can be given semantics by LTSs in which transition labels record values of transition parameters.

Although the definition in [16] was sufficiently restrictive to prove the particular reduction theorem, it was not known whether that was an accident. In other words, it was not known whether the collection of all data independent families of LTSs was equal to or strictly larger than the collection of those which arise as semantics of syntactically data independent systems. In this paper, we show that it is equal.

More precisely, we show that, in the absence of inductive types, any parametric family of LTSs whose signature consists of equality predicates, uninterpreted predicates of arbitrary arity, and uninterpreted constants, and whose types of states and transition labels do not contain functions, is definable by a symbolic LTS with the same signature. Data independence as in [16] is the special case when the uninterpreted predicates are only unary. Inductive types are not considered for simplicity, because they are orthogonal to definability of families constrained by logical relations, which is the topic of the paper. Functions within states or transition labels are also excluded in the reduction theorems in [16]. Symbolic LTSs are a basic formalism which combines simply typed λ-calculus and nondeterminism. They can be seen as a graphical variant of UNITY, and are a generalisation of first-order Kripke structures [2].

Compared with the literature on definability in models based on logical relations (e.g. [18, 13, 1, 7]), we consider only first-order computation which can use equality testing and predicates of arbitrary arity, but the novelty in our result is that it applies to nondeterministic computation.

Section 2 is devoted to preliminaries. In Section 3, we define families of values, sets, and LTSs, and when they are regarded parametric. In Section 4, we define symbolic LTSs, and observe that their semantics is parametric. Section 5 is devoted to the definability result. In Section 6, we conclude and remark about future work.

2 Preliminaries

We fix notation for the syntax and set-theoretic semantics of the simply typed λ-calculus with product and sum types, for binary logical relations, and for

LTSs. Terms of the λ-calculus will be used to form symbolic LTSs. In addition to typing the terms, types will be used to structure the semantic entities we shall consider, such as families of LTSs. Logical relations will serve to constrain families indexed by signature instantiations, such as in the semantic definition of data independence.

λ-calculus syntax. We assume $TypeVars$ is an infinite set of names for type variables.

The syntax of types T is as follows:

$$T \ ::= \ X \in TypeVars \mid T_1 \times \cdots \times T_n \mid T_1 + \cdots + T_n \mid T_1 \rightarrow T_2$$

where n ranges over nonnegative integers.

For any type T, we write $Free(T)$ for the set of free type variables of T.

We assume $TermVars$ is an infinite set of names for term variables.

A *type context* Γ is a sequence of the form

$$\langle x_1 : T_1, \ldots, x_n : T_n \rangle$$

where the x_i are distinct term variables.

We write $\Gamma(x)$ for the type associated to the term variable x in Γ. $\Gamma \setminus x$ is the type context obtained by removing $x : \Gamma(x)$ from Γ if $x \in Dom(\Gamma)$, otherwise it is Γ. $\Gamma\Gamma'$ is the concatenation of Γ and Γ' provided their domains are disjoint.

A *signature* is an ordered pair (Υ, Γ), where

- Υ is a finite subset of $TypeVars$, and
- Γ is a type context such that $Free(\Gamma(x)) \subseteq \Upsilon$ for each $x \in Dom(\Gamma)$.

Terms-in-context $\Upsilon, \Gamma \vdash t : T$ are built in the standard well-typed fashion from term variables, tuple formation, projection, injection, matching, λ-

abstraction, and application.

$$\Upsilon, \Gamma \vdash x : T \text{ if } \Gamma(x) = T$$

$$\frac{\Upsilon, \Gamma \vdash t_i : T_i \text{ for } i = 1, \dots, n}{\Upsilon, \Gamma \vdash \langle t_1, \dots, t_n \rangle : T_1 \times \cdots \times T_n}$$

$$\frac{\Upsilon, \Gamma \vdash t : T_1 \times \cdots \times T_n}{\Upsilon, \Gamma \vdash \pi_i(t) : T_i} \text{ for } i = 1, \dots, n$$

$$\frac{\Upsilon, \Gamma \vdash t : T_i}{\Upsilon, \Gamma \vdash in_i^{T_1 + \cdots + T_n}(t) : T_1 + \cdots + T_n} \text{ for } i = 1, \dots, n$$

$$\frac{\Upsilon, \Gamma \vdash t : T_1 + \cdots + T_n \quad \Upsilon, (\Gamma \setminus x)\langle x : T_i \rangle \vdash t_i : T \text{ for } i = 1, \dots, n}{\Upsilon, \Gamma \vdash \mathit{match}\; t \;\mathit{with}\; in_1(x) \Rightarrow t_1 \;[]\; \cdots \;[]\; in_n(x) \Rightarrow t_n : T}$$

$$\frac{\Upsilon, (\Gamma \setminus x)\langle x : T_1 \rangle \vdash t : T_2}{\Upsilon, \Gamma \vdash \lambda x : T_1.t : T_1 \to T_2}$$

$$\frac{\Upsilon, \Gamma \vdash t_1 : T_1 \to T_2 \quad \Upsilon, \Gamma \vdash t_2 : T_1}{\Upsilon, \Gamma \vdash t_1\; t_2 : T_2}$$

We write $\mathit{Free}(t)$ for the set of free term variables of t.

λ-calculus semantics. A *set map* is a partial map from $\mathit{TypeVars}$ to sets, whose domain is finite.

For any type T, and any set map δ such that $\mathit{Free}(T) \subseteq \mathit{Dom}(\delta)$, we write $[\![T]\!]_\delta$ for the denotational semantics of T with respect to δ:

$$\begin{aligned}
[\![X]\!]_\delta &= \delta[\![X]\!] \\
[\![T_1 \times \cdots \times T_n]\!]_\delta &= [\![T_1]\!]_\delta \times \cdots \times [\![T_n]\!]_\delta \\
[\![T_1 + \cdots + T_n]\!]_\delta &= \{1\} \times [\![T_1]\!]_\delta \cup \cdots \cup \{n\} \times [\![T_n]\!]_\delta \\
[\![T_1 \to T_2]\!]_\delta &= [\![T_1]\!]_\delta \longrightarrow [\![T_2]\!]_\delta
\end{aligned}$$

A *value map* is a map whose domain is a finite subset of $\mathit{TermVars}$.

We write $\eta[x \mapsto v]$ for the value map such that $\eta[x \mapsto v][\![x]\!] = v$, and $\eta[x \mapsto v][\![y]\!] = \eta[\![y]\!]$ if $x \neq y$.

Given a type context Γ and a set map δ such that $\mathit{Free}(\Gamma(x)) \subseteq \mathit{Dom}(\delta)$ for each $x \in \mathit{Dom}(\Gamma)$, we say that a value map η is *with respect to Γ and δ* iff

- $\mathit{Dom}(\eta) = \mathit{Dom}(\Gamma)$, and
- $\eta[\![x]\!] \in [\![\Gamma(x)]\!]_\delta$ for each $x \in \mathit{Dom}(\Gamma)$.

An *instantiation* of a signature (Υ, Γ) is an ordered pair (δ, η) such that

- δ is a set map whose domain is Υ, and

– η is a value map with respect to Γ and δ.

For any term $\Upsilon, \Gamma \vdash t : T$ and any instantiation (δ, η), we write $[\![\Upsilon, \Gamma \vdash t : T]\!]_{(\delta,\eta)}$ for the denotational semantics of $\Upsilon, \Gamma \vdash t : T$ with respect to (δ, η). This is defined by

$$[\![\Upsilon, \Gamma \vdash x : T]\!]_{(\delta,\eta)} = \eta[\![x]\!]$$

$$[\![\Upsilon, \Gamma \vdash \langle t_1, \ldots t_n\rangle : T_1 \times \cdots \times T_n]\!]_{(\delta,\eta)} = \\ \quad ([\![\Upsilon, \Gamma \vdash t_1 : T_1]\!]_{(\delta,\eta)}, \ldots, [\![\Upsilon, \Gamma \vdash t_n : T_n]\!]_{(\delta,\eta)})$$

$$[\![\Upsilon, \Gamma \vdash \pi_i(t) : T]\!]_{(\delta,\eta)} = v_i \text{ if } [\![\Upsilon, \Gamma \vdash t : T]\!]_{(\delta,\eta)} = (v_1, \ldots, v_n)$$

$$[\![\Upsilon, \Gamma \vdash in_i^{T_1+\cdots+T_n}(t) : T_1 + \cdots + T_n]\!]_{(\delta,\eta)} = (i, [\![\Upsilon, \Gamma \vdash t : T_i]\!]_{(\delta,\eta)})$$

$$[\![\Upsilon, \Gamma \vdash \mathit{match}\ t\ \mathit{with}\ in_1(x) \Rightarrow t_1 \,[\!]\, \cdots \,[\!]\, in_n(x) \Rightarrow t_n : T]\!]_{(\delta,\eta)}) = \\ \quad [\![\Upsilon, (\Gamma \setminus x)\langle x : T_i\rangle \vdash t_i : T]\!]_{(\delta,\eta[x\mapsto v])} \\ \text{if } [\![\Upsilon, \Gamma \vdash t : T_1 + \cdots + T_n]\!]_{(\delta,\eta)} = (i, v)$$

$$[\![\Upsilon, \Gamma \vdash \lambda x : T_1.t : T_1 \to T_2]\!]_{(\delta,\eta)} = f \text{ where} \\ \quad f(v) = [\![\Upsilon, (\Gamma \setminus x)\langle x : T_1\rangle \vdash t : T_2]\!]_{(\delta,\eta[x\mapsto v])}$$

$$[\![\Upsilon, \Gamma \vdash t_1\ t_2 : T_2]\!]_{(\delta,\eta)} = [\![\Upsilon, \Gamma \vdash t_1 : T_1 \to T_2]\!]_{(\delta,\eta)}([\![\Upsilon, \Gamma \vdash t_2 : T_1]\!]_{(\delta,\eta)})$$

Some abbreviations. We define some standard types:

$$\begin{aligned} Unit &= \textit{the empty product type} \\ Bool &= Unit + Unit \\ Enum_k &= \underbrace{Unit + \cdots + Unit}_{k} \end{aligned}$$

We also define terms and values:

$$\begin{array}{ll} \mathit{false} = in_1^{Bool}(\langle\rangle) & \mathit{true} = in_2^{Bool}(\langle\rangle) \\ \overline{\mathit{false}} = (1, ()) & \overline{\mathit{true}} = (2, ()) \end{array}$$

so that we have:

$$[\![Bool]\!]_{\{\}} = \{\overline{\mathit{false}}, \overline{\mathit{true}}\} \qquad [\![Enum_k]\!]_{\{\}} = \{(1, ()), \ldots, (k, ())\}$$

Logical relations. A *relation map* is a triple (ρ, δ, δ') such that

– ρ is a partial map from $TypeVars$ to relations, whose domain is finite,
– δ and δ' are set maps with the same domain as ρ, and
– $\rho[\![X]\!]$ is a relation between $\delta[\![X]\!]$ and $\delta'[\![X]\!]$, for each $X \in Dom(\rho)$.

Given a relation map (ρ, δ, δ'), it determines a logical relation [21] indexed by the types T such that $Free(T) \subseteq Dom(\rho)$. For any such type T, we write $[\![T]\!]_{(\rho,\delta,\delta')}$ for the component at T of the logical relation. This is a relation between the sets $[\![T]\!]_\delta$ and $[\![T]\!]_{\delta'}$, and is defined by

$$
\begin{aligned}
&[\![X]\!]_{(\rho,\delta,\delta')} = \rho[\![X]\!] \\
&[\![T_1 \times \cdots \times T_n]\!]_{(\rho,\delta,\delta')} = \{(\underline{a}, \underline{a'}) \mid \forall i \in \{1, \ldots, n\}. \pi_i(\underline{a})[\![T_i]\!]_{(\rho,\delta,\delta')}\pi_i(\underline{a'})\} \\
&[\![T_1 + \cdots + T_n]\!]_{(\rho,\delta,\delta')} = \{((i, a), (i', a')) \mid i = i' \ \wedge \ a[\![T_i]\!]_{(\rho,\delta,\delta')}a'\} \\
&[\![T_1 \to T_2]\!]_{(\rho,\delta,\delta')} = \{(f, f') \mid a[\![T_1]\!]_{(\rho,\delta,\delta')}a' \ \Rightarrow \ f(a)[\![T_2]\!]_{(\rho,\delta,\delta')}f'(a')\}
\end{aligned}
$$

Labelled transition systems. An *LTS* is a tuple $S = (A, B, I, \longrightarrow)$ such that A and B are sets, $I \subseteq A$ and $\longrightarrow \subseteq A \times B \times A$.

We say that A is the *set of states*, B is the *set of transition labels*, I is the *set of initial states*, and $\longrightarrow$ is the *transition relation*. We write $a_1 \xrightarrow{b} a_2$ for $(a_1, b, a_2) \in \longrightarrow$.

3 Parametric families

If (Υ, Γ) is a signature, then the semantics of a term or program which uses (Υ, Γ), with respect to a class $\mathcal{I}$ of instantiations of (Υ, Γ), can be seen as a family of semantic elements which is indexed by $\mathcal{I}$. We now define three kinds of such families, namely those of values, sets and LTSs. In the first two cases, there is a type T such that the family member whose index is (δ, η) is an element/subset of $[\![T]\!]_\delta$. In the case of LTSs, there are two types T and U which determine the sets of states and transition labels of family members.

Definition 1 (families). *A* family *of values, sets, or LTSs (respectively) is of the form* $(\Upsilon, \Gamma, T, \mathcal{I}, \underline{v})$, $(\Upsilon, \Gamma, T, \mathcal{I}, \underline{N})$, *or* $(\Upsilon, \Gamma, T, U, \mathcal{I}, \underline{S})$.

(Υ, Γ) *is a signature,* T *and* U *are types such that* $Free(T) \subseteq \Upsilon$ *and* $Free(U) \subseteq \Upsilon$, *and* $\mathcal{I}$ *is a class of instantiations of* (Υ, Γ).

For each $(\delta, \eta) \in \mathcal{I}$, *we have* $v_{(\delta,\eta)} \in [\![T]\!]_\delta$, $N_{(\delta,\eta)} \subseteq [\![T]\!]_\delta$, *and* $S_{(\delta,\eta)}$ *is an LTS with set of states* $[\![T]\!]_\delta$ *and set of transition labels* $[\![U]\!]_\delta$. □

Logical relations can be used as follows to define when a family is parametric. We shall see below that families arising as semantics of λ-calculus terms or of symbolic LTSs have this property.

The details are as in [16], except that here we treat families of values and sets explicitly, because they will be used later in the paper. The definitions of when two sets/LTSs are related can be seen as liftings of logical relations to powerset/LTS types, although we do not give such types first-class status. A more general treatment of such liftings of logical relations can be found in [8].

Definition 2 (related sets). *Suppose:*

- P *is a relation between* N *and* N';
- $M \subseteq N$ *and* $M' \subseteq N'$.

We say that P relates M and M' iff

$$\begin{array}{l} \forall x \in M \ \cdot \exists x' \in M' \cdot (x, x') \in P \\ \forall x' \in M' \cdot \ \exists x \in M \ \cdot (x, x') \in P \quad \square \end{array}$$

Definition 3 (universal partial R-bisimulation). *Suppose:*

- *P is a relation between A and A';*
- *R is a relation between B and B';*
- *$S = (A, B, I, \longrightarrow)$ and $S' = (A', B', I', \longrightarrow')$ are LTSs.*

We say that P is a universal partial R-bisimulation *between S and S' iff*

(i) whenever aPa' then $a \in I$ iff $a' \in I'$, and

(ii) whenever $a_1 P a_1'$ and bRb', then P relates $\{a_2 \mid a_1 \xrightarrow{b} a_2\}$ and $\{a_2' \mid a_1' \xrightarrow{b'}{}' a_2'\}$. $\square$

Definition 4 (parametric families). *A family $(\Upsilon, \Gamma, T, \mathcal{I}, \underline{v})$, $(\Upsilon, \Gamma, T, \mathcal{I}, \underline{N})$, or $(\Upsilon, \Gamma, T, U, \mathcal{I}, \underline{S})$ (respectively) is* parametric *iff, for any $(\delta, \eta), (\delta', \eta') \in \mathcal{I}$, and any relation map (ρ, δ, δ') such that*

$$\forall x \in Dom(\Gamma) \cdot \eta[\![x]\!][\![\Gamma(x)]\!]_{(\rho,\delta,\delta')}\eta'[\![x]\!]$$

we have

- *$v_{(\delta,\eta)}[\![T]\!]_{(\rho,\delta,\delta')}v_{(\delta',\eta')}$,*
- *$[\![T]\!]_{(\rho,\delta,\delta')}$ relates $N_{(\delta,\eta)}$ and $N_{(\delta',\eta')}$, or*
- *$[\![T]\!]_{(\rho,\delta,\delta')}$ is a universal partial $[\![U]\!]_{(\rho,\delta,\delta')}$-bisimulation between $S_{(\delta,\eta)}$ and $S_{(\delta',\eta')}$.* $\square$

We can now state the Basic Lemma of logical relations [21] in the following way.

Proposition 1. *For any term $\Upsilon, \Gamma \vdash t : T$ and class $\mathcal{I}$ of instantiations of (Υ, Γ), the family $(\Upsilon, \Gamma, T, \mathcal{I}, [\![t]\!])$ is parametric.* $\square$

Signatures which consist of equality predicates, uninterpreted predicates, and uninterpreted constants will be important later in the paper, as will types which are sums of products of type variables, and classes of instantiations whose only restriction is that equality predicates are interpreted as expected. These will determine the kind of parametric families of LTSs which our main result applies to.

Terminology 1. We say that a signature (Υ, Γ) is *EPC* iff Γ is of the form $\Gamma_E \Gamma_P \Gamma_C$ such that

- any $\Gamma_E(e)$ is of the form $(X \times X) \to Bool$,
- any $\Gamma_P(p)$ is of the form $T \to Enum_k$ where T is a product of type variables, and

– any $\Gamma_C(c)$ is a product of type variables.

A type T is *SP* iff it is a sum of products of type variables.

A class $\mathcal{I}$ of instantiations of an EPC-signature (Υ, Γ) is *full* iff it consists of all (δ, η) such that $\eta[\![e]\!]$ is the equality predicate on $\delta[\![X]\!]$ for any $e : ((X \times X) \rightarrow Bool)$ in Γ_E. □

Data independence was defined semantically in [16] as parametricity of families of LTSs whose signatures are EPC with only unary uninterpreted predicates, and whose classes of instantiations are full.

Example 1. Consider a family of LTSs defined as follows. The signature

$$\begin{aligned}\Upsilon &= \{X\}\\ \Gamma &= \langle p : X \rightarrow Bool, q : (X \times X) \rightarrow Bool\rangle\end{aligned}$$

consists of type variable X, unary uninterpreted predicate p on X, and binary uninterpreted predicate q on X.

The type of states $T = X + (X \times X)$ can be seen as two control states, the first with one data item of type X, the second with two data items of type X. The type of transition labels $U = X$ means that any transition has a parameter of type X.

$\mathcal{I}$ consists of all instantiations of (Υ, Γ), and any $S_{(\delta,\eta)}$ is such that

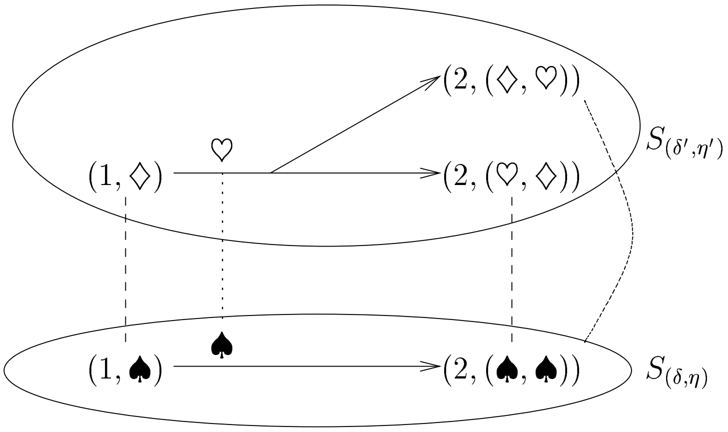
– a state is initial iff it is the first control state together with data u which satisfies p, and
– a transition is from the first control state to the second provided either the parameter w satisfies p and the two target data items v_1 and v_2 are set to w and the source data u, or (u, w) satisfies q and v_1, v_2 are set to u, w.

More formally:

$$\begin{aligned}I_{(\delta,\eta)} &= \{(1, u) \mid \eta[\![p]\!](u)\}\\ \longrightarrow_{(\delta,\eta)} &= \{((1, u), w, (2, (v_1, v_2))) \mid\\ &\qquad (\eta[\![p]\!](w) \wedge v_1 = w \wedge v_2 = u) \vee\\ &\qquad (\eta[\![q]\!](u, w) \wedge v_1 = u \wedge v_2 = w)\}\end{aligned}$$

It is straightforward to check that this family is parametric. In fact, as we shall see in Example 2, it is the semantics of a symbolic LTS.

Let $\delta[\![X]\!] = \{\spadesuit, \clubsuit\}$ and $\delta'[\![X]\!] = \{\diamondsuit, \heartsuit, \triangle\}$. Let η and η' be such that $\eta[\![p]\!]$ holds only on $\spadesuit$, $\eta'[\![p]\!]$ holds on $\diamondsuit$ and $\heartsuit$, and $\eta[\![q]\!]$ and $\eta'[\![q]\!]$ hold on all pairs. Define $\rho[\![X]\!]$ by $\spadesuit\rho[\![X]\!]\diamondsuit$ and $\spadesuit\rho[\![X]\!]\heartsuit$. Then $[\![T]\!]_{(\rho,\delta,\delta')}$ is a universal partial $[\![U]\!]_{(\rho,\delta,\delta')}$ bisimulation between $S_{(\delta,\eta)}$ and $S_{(\delta,\eta)}$, and the following figure is a simple illustration.



□

4 Symbolic labelled transition systems

The notion of SLTSs we define below is a formalism for expressing nondeterministic reactive systems, which is based on the simply typed λ-calculus introduced in Section 2.

An SLTS $\mathbf{S}$ computes on types built from type variables from Υ, using operations from Γ, where (Υ, Γ) is a signature. $\mathbf{S}$ has a set $\mathbf{A}$ of symbolic states, and a set $\mathbf{B}$ of symbolic labels. The elements of $\mathbf{A}$ and $\mathbf{B}$ have associated type contexts $\Phi(\mathbf{a})$ and $\Psi(\mathbf{b})$. Symbolic states can be thought of as control states, where $\Phi(\mathbf{a})$ are data variables at $\mathbf{a}$. Similarly, symbolic labels can be thought of as kinds of transitions, so that $\Psi(\mathbf{b})$ are parameters for transitions of kind $\mathbf{b}$.

The initial states of $\mathbf{S}$ are given as a set of pairs $(\mathbf{a}, t)$, where t is a λ-calculus term of type $Bool$ which specifies which data values associated with the symbolic state $\mathbf{a}$ form initial states.

The transitions of $\mathbf{S}$ are given by symbolic transitions. A symbolic transition has source and target symbolic states, a symbolic label, a guard, and an assignment. The guard is a λ-calculus term of type $Bool$ which determines when the symbolic transition is enabled, in which case the action of the symbolic transition is to set each data variable at the target symbolic state according to the assignment. In particular, the lifetime of data variables and transition parameters is one transition. Nondeterminism is present when $\mathbf{S}$ has more than one symbolic transition with the same source symbolic state and the same symbolic label.

The fact that SLTSs allow different sets of data variables at different symbolic states can be used e.g. to model data which is local to a part of the system. Observe also that a portion of data in a system (such as data which is not treated in a data independent manner) can be modelled non-symbolically by regarding it as part of control.

SLTSs are similar to a number of formalisms in the literature, e.g. [9, 2, 3]. They can also be seen as a graphical variant of UNITY [4].

Definition 5 (SLTS). $\mathbf{S}$ *is an* SLTS *iff it is a tuple*

$$(\Upsilon, \Gamma, \mathbf{A}, \Phi, \mathbf{B}, \Psi, \mathbf{I}, \mathbf{R})$$

such that:

- (Υ, Γ) *is a signature.*
- $\mathbf{A}$ *and* $\mathbf{B}$ *are sets. We call elements of* $\mathbf{A}$ symbolic states, *and elements of* $\mathbf{B}$ symbolic labels.
- Φ *and* Ψ *are such that, for any* $\mathbf{a} \in \mathbf{A}$ *and* $\mathbf{b} \in \mathbf{B}$, *we have that* $(\Upsilon, \Phi(\mathbf{a}))$ *and* $(\Upsilon, \Psi(\mathbf{b}))$ *are signatures, and that* $Dom(\Phi(\mathbf{a}))$ *and* $Dom(\Psi(\mathbf{b}))$ *are disjoint from* $Dom(\Gamma)$.
- $\mathbf{I}$ *is a set of ordered pairs* $(\mathbf{a}, t)$, *where* $\mathbf{a} \in \mathbf{A}$ *and* $\Upsilon, \Gamma\Phi(\mathbf{a}) \vdash t : Bool$ *is a term. We say that* t *is an* initial condition, *and that elements of* $\mathbf{I}$ *are* symbolic initial states.
- $\mathbf{R}$ *is a set of tuples of the form* $(\mathbf{a}_1, \mathbf{b}, g, E, \mathbf{a}_2)$ *where* $\mathbf{a}_1, \mathbf{a}_2 \in \mathbf{A}$, $\mathbf{b} \in \mathbf{B}$, $Dom(\Phi(\mathbf{a}_1))$ *and* $Dom(\Psi(\mathbf{b}))$ *are disjoint,* $\Upsilon, \Gamma\Phi(\mathbf{a}_1)\Psi(\mathbf{b}) \vdash g : Bool$ *is a term, and* E *is such that, for any* $x \in Dom(\Phi(\mathbf{a}_2))$, $\Upsilon, \Gamma\Phi(\mathbf{a}_1)\Psi(\mathbf{b}) \vdash E(x) : \Phi(\mathbf{a}_2)(x)$ *is a term. We say that* $\mathbf{a}_1$ *is the* symbolic source state, $\mathbf{a}_2$ *is the* symbolic target state, g *is the* guard, E *is the* assignment, $\mathbf{R}$ *is the* symbolic transition relation *and its elements are* symbolic transitions. *We write* $\mathbf{a}_1 \,[\mathbf{b} : g \hookrightarrow E\rangle_{\mathbf{R}}\, \mathbf{a}_2$ *for* $(\mathbf{a}_1, \mathbf{b}, g, E, \mathbf{a}_2) \in \mathbf{R}$. □

Example 2. The following SLTS is illustrated in the figure.

- $\Upsilon = \{X\}$;
- $\Gamma = \langle p : X \to Bool, q : (X \times X) \to Bool \rangle$;
- $\mathbf{A} = \{\mathbf{a}_1, \mathbf{a}_2\}$;
- $\Phi(\mathbf{a}_1) = \langle x : X \rangle$, $\Phi(\mathbf{a}_2) = \langle y_1 : X, y_2 : X \rangle$;
- $\mathbf{B} = \{\mathbf{b}\}$;
- $\Psi(\mathbf{b}) = \langle z : X \rangle$;
- $\mathbf{I} = \{(\mathbf{a}_1, p\, x)\}$;
- $\mathbf{a}_1 \,[\mathbf{b} : p\, z \hookrightarrow y_1, y_2{:=}z, x\rangle\, \mathbf{a}_2$ and $\mathbf{a}_1 \,[\mathbf{b} : q\, x\, z \hookrightarrow y_1, y_2{:=}x, z\rangle\, \mathbf{a}_2$ are the symbolic transitions.

$$\begin{tikzcd}
\begin{array}{c}\mathbf{a}_1\\ \langle x : X\rangle\end{array} \arrow[rr, bend left, "\begin{array}{c}\mathbf{b}\\ \langle z : X\rangle\\ p\ z \hookrightarrow y_1, y_2{:=}z, x\end{array}"] \arrow[rr, bend right, "\begin{array}{c}\mathbf{b}\\ \langle z : X\rangle\\ q\ x\ z \hookrightarrow y_1, y_2{:=}x, z\end{array}"'] & & \begin{array}{c}\mathbf{a}_2\\ \langle y_1 : X, y_2 : X\rangle\end{array}
\end{tikzcd}$$

□

Given an SLTS $\mathbf{S}$ and an instantiation (δ, η) of its signature (Υ, Γ), we can define a concrete LTS $[\![\mathbf{S}]\!]_{(\delta,\eta)}$. Provided the sets of symbolic states and symbolic labels of $\mathbf{S}$ are finite, and given a class of instantiations of (Υ, Γ), the concrete LTSs $[\![\mathbf{S}]\!]_{(\delta,\eta)}$ form a family of LTSs which is always parametric.

Notation 1. Given a signature (Υ, Δ), where $\Delta = \langle x_1 : T_1, \ldots, x_n : T_n\rangle$, and a set map δ such that $\Upsilon \subseteq Dom(\delta)$, let $[\![\Delta]\!]_\delta$ be defined by

$$[\![\Delta]\!]_\delta = \{(v_1, \ldots, v_n) \mid \forall i \ \cdot \ v_i \in [\![T_i]\!]_\delta\}$$

Given a type context $\Delta = \langle x_1 : T_1, \ldots, x_n : T_n\rangle$, a value map ζ such that $Dom(\zeta) \cap Dom(\Delta) = \{\}$, and a tuple $\underline{v} = (v_1, \ldots, v_n)$, let $\zeta \oplus_\Delta \underline{v}$ be the map ζ extended by $x_i \mapsto v_i$ for all $x_i \in Dom(\Delta)$. □

Definition 6 (semantics of SLTSs). *Given an SLTS*

$$\mathbf{S} = (\Upsilon, \Gamma, \mathbf{A}, \Phi, \mathbf{B}, \Psi, \mathbf{I}, \mathbf{R})$$

and an instantiation (δ, η) of (Υ, Γ), let $[\![\mathbf{S}]\!]_{(\delta,\eta)}$ be the LTS $(A, B, I, \longrightarrow)$ defined as follows:

- $A = \{(\mathbf{a}, \underline{v}) \mid \mathbf{a} \in \mathbf{A} \ \wedge \ \underline{v} \in [\![\Phi(\mathbf{a})]\!]_\delta\}$.
- $B = \{(\mathbf{b}, \underline{w}) \mid \mathbf{b} \in \mathbf{B} \ \wedge \ \underline{w} \in [\![\Psi(\mathbf{b})]\!]_\delta\}$.
- $I = \bigcup_{(\mathbf{a},t)\in\mathbf{I}} [\![(\mathbf{a}, t)]\!]_{(\delta,\eta)}$ *where*

$$[\![(\mathbf{a}, t)]\!]_{(\delta,\eta)} = \{(\mathbf{a}, \underline{v}) \in A \mid [\![\Upsilon, \Gamma\Phi(\mathbf{a}) \vdash t : Bool]\!]_{(\delta,\eta\oplus_{\Phi(\mathbf{a})}\underline{v})} = \overline{true}\}.$$

- *The transition relation $\longrightarrow$ is the set of triples*

$$((\mathbf{a}_1, \underline{v^1}), (\mathbf{b}, \underline{w}), (\mathbf{a}_2, \underline{v^2})) \in A \times B \times A$$

such that for some g and E we have

- $(\mathbf{a}_1, \mathbf{b}, g, E, \mathbf{a}_2) \in \mathbf{R}$,
- $[\![\Upsilon, \Gamma \Phi(\mathbf{a}_1)\Psi(\mathbf{b}) \vdash g : Bool]\!]_{(\delta,(\eta \oplus_{\Phi(\mathbf{a}_1)} \underline{v^1}) \oplus_{\Psi(\mathbf{b})} \underline{w})} = \overline{true}$, *and*
- *for all* $x_i \in Dom(\Phi(\mathbf{a}_2))$,

$$[\![\Upsilon, \Gamma \Phi(\mathbf{a}_1)\Psi(\mathbf{b}) \vdash E(x_i) : \Phi(\mathbf{a}_2)(x_i)]\!]_{(\delta,(\eta \oplus_{\Phi(\mathbf{a}_1)} \underline{v^1}) \oplus_{\Psi(\mathbf{b})} \underline{w})} = v_i^2$$

where x_i *is the* i*th component of* $Dom(\Phi(\mathbf{a}_2))$. □

Proposition 2. *Suppose* $\mathbf{S} = (\Upsilon, \Gamma, \mathbf{A}, \Phi, \mathbf{B}, \Psi, \mathbf{I}, \mathbf{R})$ *is an SLTS such that* $\mathbf{A}$ *and* $\mathbf{B}$ *are finite. Let* $T = \sum_{\mathbf{a} \in \mathbf{A}} \prod_{x \in \Phi(\mathbf{a})} \Phi(\mathbf{a})(x)$ *and* $U = \sum_{\mathbf{b} \in \mathbf{B}} \prod_{y \in \Psi(\mathbf{b})} \Psi(\mathbf{b})(y)$. *For any class* $\mathcal{I}$ *of instantiations of* (Υ, Γ), *we have that* $(\Upsilon, \Gamma, T, U, \mathcal{I}, [\![\mathbf{S}]\!])$ *is a parametric family of LTSs.*

Proof. This is a straightforward consequence of Proposition 1. The most interesting part is when, in notation of Definition 3, $a_1 [\![T]\!]_{(\rho,\delta,\delta')} a_1'$, $b [\![U]\!]_{(\rho,\delta,\delta')} b'$, and $a_1 \xrightarrow{b} a_2$. Then any symbolic transition which gave rise to a_2 can be used to show that there exists a_2' such that $a_1' \xrightarrow{b'}{}' a_2'$ and $a_2 [\![T]\!]_{(\rho,\delta,\delta')} a_2'$.

In the statement, the $[\![\mathbf{S}]\!]_{(\delta,\eta)}$ are regarded as members of the family of LTSs by identifying $\mathbf{A}$ and $\mathbf{B}$ with $\{1, \ldots, |\mathbf{A}|\}$ and $\{1, \ldots, |\mathbf{B}|\}$. □

When restricted to EPC signatures with only unary uninterpreted predicates and to full classes of instantiations, Proposition 2 states that the semantics of any syntactically data independent SLTS is data independent according to the semantic definition in [16].

Example 3. The SLTS in Example 2 yields the parametric family of LTSs in Example 1, for the class of all instantiations. □

5 Definability

This section contains the main result of the paper, namely that any parametric family of LTSs whose signature is EPC, whose types of states and transition labels are SP, and whose class of instantiations is full, is definable by an SLTS. In particular, this shows that the semantic definition of data independence in [16] is sufficiently strong. The SP assumption is equivalent to assuming absence of the function-type construct, which is done in the reduction theorems in [16].

Before the theorem, we present a proposition and a lemma which are used in its proof.

Proposition 3. *For any parametric family of values*

$$(\Upsilon, \Gamma, T, \mathcal{I}, \underline{v})$$

such that (Υ, Γ) *is EPC,* T *is SP, and* $\mathcal{I}$ *is full, there exists a term*

$$\Upsilon, \Gamma \vdash s : T$$

such that, for any $(\delta,\eta) \in \mathcal{I}$, $[\![s]\!]_{(\delta,\eta)} = v_{(\delta,\eta)}$, and such that s is of the form

$$\mathit{match}\ h\ \mathit{with}\ [\!]_{i=1}^{H}\ \ in_i(x)\ \Rightarrow\ in_{R_i}^{T}(r_i)$$

where $H \in \mathbb{N}$, $\Upsilon, \Gamma \vdash h : \mathit{Enum}_H$ is a term, $T = \sum_{i=1}^{n} T_i$, and for each i, $R_i \in \{1,\ldots,n\}$ and $\Upsilon, \Gamma_C \vdash r_i : T_{R_i}$ is a term.

Proof outline. The class $\mathcal{I}$ can be split into finitely many subclasses according to the results of all possible applications of the equality predicates and the uninterpreted predicates to the uninterpreted constants. The subclasses have the property that two instantiations can be related by a relation map iff they belong to the same subclass.

The term s can be defined by letting H be the number of subclasses. Each R_i and r_i are defined by considering an instantiation from the corresponding subclass which, for any $X \in \Upsilon$ without an equality predicate in Γ_E, instantiates any two uninterpreted constant components of type X by distinct values. □

Example 4. This example due to Plotkin shows that Proposition 3 cannot be straightforwardly extended to second order. It is a parametric family of values which is not definable.

The signature consists of one type variable X, predicate $p : X \to \mathit{Bool}$, operation $s : X \to X$, and constant $z : X$. The type of the family is Bool, and for any instantiation (δ,η) of the signature, the member $v_{(\delta,\eta)}$ is defined to be $\overline{\mathit{true}}$ iff, for all $n \in \mathbb{N}$, the result of applying n times $\eta[\![s]\!]$ to $\eta[\![z]\!]$ satisfies $\eta[\![p]\!]$. □

Terminology 2. We say that a family of sets is *deterministic* iff each set of the family is either the empty set or a singleton. □

Notation 2. We write $(\Upsilon,\Gamma,T,\mathcal{I},\underline{N}) \sqsubseteq (\Upsilon',\Gamma',T',\mathcal{I}',\underline{N'})$ iff we have $\Upsilon = \Upsilon'$, $\Gamma = \Gamma'$, $T = T'$, $\mathcal{I} = \mathcal{I}'$, and $N_{(\delta,\eta)} \subseteq N'_{(\delta,\eta)}$ for all $(\delta,\eta) \in \mathcal{I}$.

We write $(\Upsilon,\Gamma,T,\mathcal{I},\underline{N}) \sqcup (\Upsilon,\Gamma,T,\mathcal{I},\underline{N'})$ for the family of sets $(\Upsilon,\Gamma,T,\mathcal{I},\underline{M})$ where, for any $(\delta,\eta) \in \mathcal{I}$, $M_{(\delta,\eta)} = N_{(\delta,\eta)} \cup N'_{(\delta,\eta)}$. □

Lemma 1. *Given any parametric family of sets*

$$\mathcal{N} = (\Upsilon,\Gamma,T,\mathcal{I},\underline{N})$$

such that (Υ,Γ) is EPC, T is SP, and $\mathcal{I}$ is full, there are parametric families of sets $\mathcal{M}_1, \ldots, \mathcal{M}_m$ such that:

(i) *$\mathcal{M}_i \sqsubseteq \mathcal{N}$ for each i;*
(ii) *$\mathcal{M}_i$ is deterministic for each i;*
(iii) *given any parametric family of sets $\mathcal{M}'$ such that $\mathcal{M}' \sqsubseteq \mathcal{N}$ and $\mathcal{M}'$ is deterministic, it is equal to $\mathcal{M}_i$ for some i;*
(iv) *$\bigsqcup_{i=1}^{m} \mathcal{M}_i = \mathcal{N}$.* □

The proof of Lemma 1 has similar structure to the proof of Proposition 3, which means that the definability of families of sets can be shown somewhat more directly than by combining the two results. However, Lemma 1 is of wider interest, since it shows that definability of parametric nondeterministic families can be reduced to definability of finitely many parametric deterministic families.

Example 5. Recall the SLTS in Example 2 and its semantics in Example 1.

Take $\mathcal{N}$ to be the family of sets corresponding to the nondeterministic computation at symbolic state $\mathbf{a}_1$ and symbolic label $\mathbf{b}$. Its signature is $\Upsilon = \{X\}$ and

$$\Gamma = \langle p : X \to Bool, q : (X \times X) \to Bool, x : X, z : X \rangle$$

The type of $\mathcal{N}$ is $X \times X$, and the class consists of all instantiations of (Υ, Γ). For any (δ, η), $N_{(\delta,\eta)}$ is the set of all outcomes of the two symbolic transitions when p, q, x and z have the values given by η. If neither symbolic transition is enabled, $N_{(\delta,\eta)} = \{\}$.

Similarly, we can let $\mathcal{M}_1$ and $\mathcal{M}_2$ be families of sets corresponding to the two symbolic transitions respectively.

Parametricity of these families of sets follows from parametricity of the family of LTSs. Also, $\mathcal{M}_1$ and $\mathcal{M}_2$ are deterministic, and $\mathcal{M}_1 \sqcup \mathcal{M}_2 = \mathcal{N}$. Therefore, $\mathcal{M}_1$ and $\mathcal{M}_2$ are two of the families corresponding to $\mathcal{N}$ in the statement of Lemma 1. There are others, e.g. the empty family. □

Theorem 1. *For any parametric family of LTSs $(\Upsilon, \Gamma, T, U, \mathcal{I}, \underline{S})$ such that (Υ, Γ) is EPC, T and U are SP, and $\mathcal{I}$ is full, there exists a finite SLTS $\mathbf{S}$ with the same signature and such that, for any $(\delta, \eta) \in \mathcal{I}$, $[\![\mathbf{S}]\!]_{(\delta,\eta)} = S_{(\delta,\eta)}$.*

Proof outline. Symbolic states and symbolic labels of $\mathbf{S}$ are defined to correspond to the sum components of T and U.

For each symbolic state $\mathbf{a}$, the sets of initial states of $S_{(\delta,\eta)}$ restricted to $\mathbf{a}$ form a parametric family of predicates, so that Proposition 3 can be applied to obtain the initial condition at $\mathbf{a}$.

For each symbolic state $\mathbf{a}$ and symbolic label $\mathbf{a}$, the transitions of the concrete LTSs $S_{(\delta,\eta)}$ form a parametric family of sets whose type is T. Lemma 1 can be applied to this family to yield a finite number of deterministic families. Symbolic transitions of $\mathbf{S}$ are then obtained by applying Proposition 3 to families of values which correspond to the deterministic families of sets. □

Example 6. Theorem 1 applies to the family of LTSs in Example 1. We already saw that this family is definable by the SLTS in Example 2. □

6 Conclusions

This paper answers negatively the question whether there are any data independent families of LTSs [16] which do not arise as semantics of any syntactically data independent system. Thus we confirm that the semantic definition of data independence is suitable for reasoning about data independent systems without being tied to a particular syntax.

More precisely, we showed that any parametric family of LTSs whose signature consists of equality predicates, uninterpreted predicates of arbitrary arity, and uninterpreted constants, and whose types of states and transition labels do not contain functions, is definable by a symbolic LTS. Data independence is the special case with only unary uninterpreted predicates.

From another point of view, we demonstrated that when binary logical relations are extended to nondeterministic computation by means of bisimulation, they can be used to ensure that any parametric family is definable.

Future work should investigate definability of parametric families in settings where powerset types have first-class status [17, 11].

Acknowledgements

We are grateful to Samson Abramsky, Brian Dunphy, Andrew Pitts, Gordon Plotkin, Uday Reddy, Bill Roscoe and Alex Simpson for useful discussions.

References

1. M. Alimohamed. A characterization of lambda definability in categorical models of implicit polymorphism. *Theoretical Computer Science*, 146:5–23, 1995.
2. J. Bohn, W. Damm, O. Grumberg, H. Hungar, and K. Laster. First-order-CTL model checking. In *Foundations of Software Technology and Theoretical Computer Science (FST&TCS'98)*, volume 1530 of *Lecture Notes in Computer Science*, pages 283–294. Springer-Verlag, 1998.
3. M. Calder and C. Shankland. A symbolic semantics and bisimulation for full LOTOS. In *International Conference on Formal Description Techniques for Networked and Distributed Systems (FORTE'01)*, pages 184–200. Kluwer Academic Publishers, 2001.
4. K. M. Chandy and J. Misra. *Parallel Program Design: A Foundation*. Addison-Wesley, 1988.
5. E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. MIT Press, 1999.
6. D. Dill, R. Hojati, and R.K. Brayton. Verifying linear temporal properties of data intensive controllers using finite instantiations. In *Hardware Description Languages and their Applications (CHDL '97)*. Chapman and Hall, 1997.
7. M. Fiore and A. Simpson. Lambda definability with sums via Grothendieck logical relations. In *Proceedings of the 4th International Conference on Typed Lambda Calculi and Applications (TLCA'99)*, volume 1581 of *Lecture Notes in Computer Science*, pages 147–161. Springer-Verlag, 1999.
8. J. Goubault-Larrecq, S. Lasota, and D. Nowak. Logical relations for monadic types. In *Proceedings of the 11th Annual Conference of the European Association for Computer Science Logic (CSL'02)*, volume 2471 of *Lecture Notes in Computer Science*, pages 553–568. Springer-Verlag, 2002.
9. M. Hennessy and H. Lin. Symbolic bisimulations. *Theoretical Computer Science*, 138(2):353–389, 1995.

10. C. N. Ip and D. L. Dill. Better verification through symmetry. *Formal Methods in System Design: An International Journal*, 9(1/2):41–75, 1996.
11. A. Jeffrey. A fully abstract semantics for a higher-order functional language with nondeterministic computation. *Theoretical Computer Science*, 228:105–150, 1999.
12. B. Jonsson and J. Parrow. Deciding bisimulation equivalences for a class of non-finite-state programs. *Information and Computation*, 107(2):272–302, 1993.
13. A. Jung and J. Tiuryn. A new characterization of lambda definability. In *Proceedings of the 1st International Conference on Typed Lambda Calculi and Applications (TLCA'93)*, volume 664 of *Lecture Notes in Computer Science*, pages 245–257. Springer-Verlag, 1993.
14. R. Lazić. *A Semantic Study of Data Independence with Applications to Model Checking*. DPhil thesis, Oxford University Computing Laboratory, 1999.
15. R. Lazić and D. Nowak. On a semantic definition of data independence. Research report, Department of Computer Science, University of Warwick. http://www.dcs.warwick.ac.uk.
16. R. Lazić and D. Nowak. A unifying approach to data-independence. In *Proceedings of the 11th International Conference on Concurrency Theory (CONCUR 2000)*, volume 1877 of *Lecture Notes in Computer Science*, pages 581–595. Springer-Verlag, 2000. Also available as Technical report PRG-TR-4-00, Oxford University Computing Laboratory.
17. T. Nipkow. Non-deterministic data types: models and implementations. *Acta Informatica*, 22(6):629–661, 1986.
18. G. D. Plotkin. Lambda-definability in the full type hierarchy. In *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 363–373. Academic Press, 1980.
19. S. Qadeer. Verifying sequential consistency on shared-memory multiprocessors by model checking. Research Report 176, Compaq, 2001.
20. R. Hojati and R. K. Brayton. Automatic datapath abstraction in hardware systems. In *Proceedings of the 7th International Conference On Computer Aided Verification*, volume 939 of *Lecture Notes in Computer Science*, pages 98–113. Springer Verlag, 1995.
21. J. C. Reynolds. Types, abstraction and parametric polymorphism. In *Proceedings of the 9th IFIP World Computer Congress (IFIP'83)*, pages 513–523. North-Holland, 1983.
22. A. W. Roscoe and P. J. Broadfoot. Proving security protocols with model checkers by data independence techniques. *Journal of Computer Security, Special Issue on the 11th IEEE Computer Security Foundations Workshop (CSFW11)*, pages 147–190, 1999.
23. P. Wolper. Expressing interesting properties of programs in propositional temporal logic. In *Conference Record of the 13th Annual ACM Symposium on Principles of Programming Languages*, pages 184–193. ACM, 1986.

A Proofs

Proof (Proposition 3). Without loss of generality, we can assume Γ_C is of the form

$$\langle c_j^i : Z_i \mid i = 1, \ldots, l \wedge j = 1, \ldots, l'_i \rangle$$

where the Z_i are mutually distinct and $\Upsilon = \{Z_1, \ldots, Z_l\}$.

Let us say that two instantiations (δ, η) and (δ', η') in $\mathcal{I}$ are related by $\mathcal{R}$ iff they are related by some relation map (ρ, δ, δ'). It is straightforward to check that $\mathcal{R}$ is an equivalence relation.

Let I be the set of all $(\delta, \eta) \in \mathcal{I}$ such that:

- if there is an equality predicate on Z_i in $\varGamma_E$, then $\delta[\![Z_i]\!]$ is the set of equivalence classes of an equivalence relation on $\{c^i_j \mid j = 1, \ldots, l'_i\}$;
- otherwise, $\delta[\![Z_i]\!] = \{\{c^i_j\} \mid j = 1, \ldots, l'_i\}$;
- for any $p \in Dom(\varGamma_P)$, $\eta[\![p]\!]$ is arbitrary;
- $\eta[\![c^i_j]\!]$ is the equivalence class of c^i_j.

It is routine to show that any equivalence class of $\mathcal{R}$ contains exactly one member of I.

Since I is a finite set, let H be its cardinality, and let f be a bijection from $\{1, \ldots, H\}$ to I. It is straightforward to define a term

$$\varUpsilon, \varGamma \vdash h : \mathit{Enum}_H$$

such that, for any $(\delta, \eta) \in \mathcal{I}$ and $i \in \{1, \ldots, H\}$,

$$[\![h]\!]_{(\delta,\eta)} = (i, ()) \;\Leftrightarrow\; (\delta, \eta)\mathcal{R}f(i)$$

For any $i \in \{1, \ldots, H\}$, let

$$(R_i, (w^i_1, \ldots, w^i_{n'_{R_i}})) = v_{f(i)}$$

and define r_i as $(d^i_1, \ldots, d^i_{n'_{R_i}})$, where $d^i_j \in w^i_j$ for all j.

We have now defined H, h, and for each $i \in \{1, \ldots, H\}$, R_i and r_i, which provides a definition of s. Given $(\delta, \eta) in \mathcal{I}$, by considering $i \in \{1, \ldots, H\}$ such that $(\delta, \eta)\mathcal{R}f(i)$, it follows that $[\![s]\!]_{(\delta,\eta)} = v_{(\delta,\eta)}$. □

Proof (Lemma 1). Let us fix notation for components of T by

$$T = \sum_{i=1}^{n} \prod_{j=1}^{n'_i} X_{i,j}$$

We use the same assumption about $\varGamma_C$ as in the proof of Proposition 3, without loss of generality.

We define $\mathcal{R}$, I, H and f as in the proof of Proposition 3.

For any $i \in \{1, \ldots, H\}$, $N_{f(i)}$ is of the form

$$\{(R_{i,j}, (w^{i,j}_1, \ldots, w^{i,j}_{n'_{R_{i,j}}})) \mid j \in \{1, \ldots, H'_i\}\}$$

Let G be the set of all maps g on $\{1, \ldots, H\}$ such that, for any i, $g(i) \in \{0, 1, \ldots, H'_i\}$.

We define m as the cardinality of G, and for any $g \in G$, we define $\mathcal{M}_g$ as follows. Suppose $(\delta, \eta) \in \mathcal{I}$, let $i \in \{1, \ldots, H\}$ be such that $(\delta, \eta)\mathcal{R}f(i)$, and let

$$\rho[\![X]\!] = \{(\eta[\![c]\!], \eta'[\![c]\!]) \mid (c : X) \in \Gamma_C\}$$

where $(\delta', \eta') = f(i)$. Then (δ, η) and (δ', η') are related by (ρ, δ, δ'). We define

$$M^g_{(\delta,\eta)} = \begin{cases} \{\}, & \textit{if } g(i) = 0 \\ \{(R_{i,g(i)}, (u^{i,g(i)}_1, \ldots, u^{i,g(i)}_{n'_{R_{i,g(i)}}}))\}, & \textit{if } g(i) \neq 0 \end{cases}$$

where the $u^{i,g(i)}_k$ are uniquely determined by $u^{i,g(i)}_k \rho[\![X_{R_{i,g(i)},k}]\!] w^{i,g(i)}_k$.

It is straightforward to check that each $\mathcal{M}_g = (\Upsilon, \Gamma, T, \mathcal{I}, \underline{M^g})$ is a parametric family of sets, and that (i)–(iv) in the statement of the lemma hold. $\square$

Proof (Theorem 1). Let us first fix some notation:

$$T = \sum_{i=1}^{n} \prod_{j=1}^{n'_i} X_{i,j}$$

$$U = \sum_{i=1}^{m} \prod_{j=1}^{m'_i} Y_{i,j}$$

$$S_{(\delta,\eta)} = ([\![T]\!]_{(\delta,\eta)}, [\![U]\!]_{(\delta,\eta)}, I_{(\delta,\eta)}, \longrightarrow_{(\delta,\eta)})$$

We define an SLTS

$$\mathbf{S} = (\Upsilon, \Gamma, \mathbf{A}, \Phi, \mathbf{B}, \Psi, \mathbf{I}, \mathbf{R})$$

as follows.

$$\begin{aligned} \mathbf{A} &= \{1, \ldots, n\} \\ \Phi(i) &= \langle x_{i,j} : X_{i,j} \mid j = 1, \ldots, n'_i \rangle \\ \mathbf{B} &= \{1, \ldots, m\} \\ \Psi(i) &= \langle y_{i,j} : Y_{i,j} \mid j = 1, \ldots, m'_i \rangle \end{aligned}$$

where the $x_{i,j}$ and $y_{i,j}$ do not appear in Γ.

Suppose $i \in \mathbf{A}$. Let

$$\mathcal{V} = (\Upsilon, \Gamma\Phi(i), Bool, \mathcal{J}, \underline{v})$$

be the family of values such that $\mathcal{J}$ is full and

$$v_{(\delta,\theta)} = \begin{cases} \overline{true}, & \textit{if } (i, \theta \restriction \Phi(i)) \in I_{(\delta, \theta \restriction \Gamma)} \\ \overline{false}, & \textit{otherwise} \end{cases}$$

Then $\mathcal{V}$ is parametric, so Proposition 3 gives us a term $\Upsilon, \Gamma\Phi(i) \vdash t_i : Bool$ such that $[\![t_i]\!]_{(\delta,\theta)} = v_{(\delta,\theta)}$ for all $(\delta, \theta) \in \mathcal{J}$.

We define $\mathbf{I} = \{(i, t_i) \mid i \in \mathbf{A}\}$.

Suppose $i \in \mathbf{A}$ and $j \in \mathbf{B}$. Let

$$\mathcal{N} = (\Upsilon, \Gamma\Phi(i)\Psi(j), T, \mathcal{K}, \underline{N})$$

be the family of sets such that $\mathcal{K}$ is full and

$$N_{(\delta,\zeta)} = \{a \mid (i, \zeta \upharpoonright \Phi(i)) \overset{(j,\zeta\upharpoonright\Psi(j))}{\longrightarrow}_{(\delta,\zeta\upharpoonright\Gamma)} a\}$$

Then $\mathcal{N}$ is parametric, so we can apply Lemma 1 to obtain $\mathcal{M}_1, \ldots, \mathcal{M}_G$.

Consider $k \in \{1, \ldots, G\}$. Let

$$\mathcal{W} = (\Upsilon, \Gamma\Phi(i)\Psi(j), T + Unit, \mathcal{K}, \underline{w})$$

be the family of values such that

$$w_{(\delta,\zeta)} = \begin{cases} a, \ \textit{if } M^k_{(\delta,\zeta)} = \{a\} \\ (n+1, ()), \ \textit{if } M^k_{(\delta,\zeta)} = \{\} \end{cases}$$

$\mathcal{W}$ is parametric by parametricity of $\mathcal{M}_k$, so Proposition 3 gives us a term

$$\Upsilon, \Gamma\Phi(i)\Psi(j) \vdash s : T + Unit$$

which defines $\mathcal{W}$ and which is of the form

$$match\ h\ with\ []_{l=1}^{H}\ \ in_l(x)\ \Rightarrow\ in_{R_l}^{T+Unit}(r_l)$$

For any $l \in \{1, \ldots, H\}$ such that $R_l \in \{1, \ldots, n\}$, let

$$\begin{aligned} g_{i,j,k,l} &= if\ h = l\ then\ true\ else\ false \\ E_{i,j,k,l}(x_{R_l,j'}) &= \pi_{j'}(r_l) \\ i'_{i,j,k,l} &= R_l \end{aligned}$$

$\mathbf{R}$ is defined to be the set of all $(i, j, g_{i,j,k,l}, E_{i,j,k,l}, i'_{i,j,k,l})$ as above.

It is routine to check that, for any $(\delta, \eta) \in \mathcal{I}$, $[\![\mathbf{S}]\!]_{(\delta,\eta)} = S_{(\delta,\eta)}$. □