Programming Research Group

APPROXIMATING SIMULATIONS

Ingrid M. Rewitzky

PRG-TR-1-98

Oxford University Computing Laboratory
Wolfson Building, Parks Road, Oxford OX1 3QD

Approximating simulations

Ingrid M. Rewitzky
Programming Research Group, Oxford University Computing Laboratory
Parks Road, Oxford OX1 3QD, United Kingdom

Abstract: Simulation rules provide a simple, sound and complete technique for proving that one data type refines another. This paper expresses the dependence of the soundness and completeness on the programming language by showing how sound and complete simulation rules for a given language are derived from those for its sublanguages. The dependence of the completeness on the semantic model is expressed using a Jónsson/Tarski duality translation between the relational and predicate transformer semantics of data types and simulation rules. Both these dependencies are considered in the context of a simple language for terminating programs.

Keywords: Data refinement, data structure, formal semantics, Jónsson/Tarski duality, simulation rule.

1 Introduction

Simulation rules have been introduced [10, 12, 8] as a technique for developing or verifying an implementation against its more abstract specification. Their importance for establishing *refinement* of abstract by concrete data structures results from their *soundness* and *completeness*. These theoretical properties depend on both the programming language and the semantic model in which the rules are studied. In this paper we investigate each of these dependencies.

Using relational methods, we show how simulations between data types invoked by a given language may be naturally approximated. These approximations are simulations for the simplest possible languages. However their value lies in providing an approach for establishing soundness and completeness of simulations from that of simpler simulations.

We use *Jónsson/Tarski duality* [9], based on *Stone's representation theorem* [14], to effect a translation (similar to that in [13]) between the relational [11, 8] and predicate transformer [3] semantics of data types and simulation rules. This automatically provides a reformulation of the explanation in [3] of why in the relational model two kinds of simulation rule are necessary to establish completeness while in the predicate transformer model only one is required.

The programming languages considered in the paper are all sublanguages (of terminating programs) of a language $\mathcal{L}$ which in turn is a sublanguage of the analogue of Dijkstra's [2] guarded command language considered in [7].

2 Data types and simulations

The language $\mathcal{L}$ is defined by

$$prog := skip \mid abort \mid z := e \mid proc \mid prog; prog \mid prog \sqcap prog \mid while\ c\ do\ prog.$$

Each *prog* updates a global data space, the cartesian product of the types of all the global program variables. The program '$z := e$' denotes an assignment of the global value of an arithmetic expression 'e' to a global variable 'z'. A *proc* denotes the invocation of a procedure which is defined as part of a *data type* $\mathcal{A}$. Such a type has its own local data space $l_{\mathcal{A}}$, an initialisation operation $in_{\mathcal{A}}$ from global to local data, an indexed set $\mathcal{O}$ of operations on local data (its procedures for invocation by programs), and a finalisation operation $fin_{\mathcal{A}}$ from local to global data. A data type $\mathcal{A}$ may be defined at any level of abstraction in a refinement calculus and independently of a semantic model. Its operations describe, at the appropriate level of abstraction, *$\mathcal{L}$-computations* which may include the natural operations on the data objects (such as *union*, *intersection* and *ismember* for the data type of *sets*), $\mathcal{L}$-program constructs, and the more specification oriented constructs (such as miracles, unbounded demonic nondeterministic choice, and angelic nondeterministic choice).

Two data types are *conformal* [8] if they have the same global data spaces and the same indexing set with respect to which corresponding operations have the same arity. Refinement of a data type $\mathcal{A}$ by a conformal data type $\mathcal{C}$ ensures that any program invoking the operations of $\mathcal{A}$ performs at least as well (in terms of program refinement '$\sqsubseteq$') as the corresponding program invoking instead the operations of $\mathcal{C}$. Such invocations may be compared in terms of refinement between *complete programs* [8]: for a data type $\mathcal{A}$, a complete program hides the effect of its operations on the local data and is of the form '$in_{\mathcal{A}}; prog_{\mathcal{A}}; fin_{\mathcal{A}}$' where $prog_{\mathcal{A}}$ is constructed from the operations of $\mathcal{A}$ using $\mathcal{L}$-program constructs.

Definition 2.1 *A data type $\mathcal{A}$ is $\mathcal{L}$-refined by a conformal data type $\mathcal{C}$, written $\mathcal{A} \sqsubseteq_{\mathcal{L}} \mathcal{C}$, if*

$$in_{\mathcal{A}}; prog_{\mathcal{A}}; fin_{\mathcal{A}} \sqsubseteq in_{\mathcal{C}}; prog_{\mathcal{C}}; fin_{\mathcal{C}}$$

for all $prog_{\mathcal{A}}$ invoking operations of $\mathcal{A}$ and the corresponding $prog_{\mathcal{C}}$ invoking operations of $\mathcal{C}$. There $\mathcal{A}$ and $\mathcal{C}$ are called respectively the abstract and concrete data types. Two data types that refine each other are said to be equivalent.

Simulations [8] are useful for establishing data refinement between conformal data types (since structural induction arguments over all their complete programs are not required) and they make program development possible. We will consider simulation rules for the hierarchy $\{\mathcal{L}_n\}_{n \leq \omega}$ of sublanguages of $\mathcal{L}$, where $\mathcal{L}_0$ is defined by

$$prog_0 := skip \mid z := e \mid proc \mid prog_0 \sqcap prog_0,$$

$\mathcal{L}_n$ (for $0 < n < \omega$) is the language in which each program is the composition of at most $(n+1)$ programs from $\mathcal{L}_0$, and $\mathcal{L}_\omega = \bigcup_{n<\omega} \mathcal{L}_n$. For each n, computations, *procs*, refinement '$\sqsubseteq_n$' and complete programs are defined in terms of the program constructs of $\mathcal{L}_n$ not $\mathcal{L}$. Observe that the languages of this hierarchy have increasingly expressive sequential composition constructs with $\mathcal{L}_\omega$ having the sequential composition construct of $\mathcal{L}$, but *abort* and programs with infinite behaviour are not allowed.

Let $\mathcal{A}$ and $\mathcal{C}$ be conformal data types. A $\mathcal{L}_\omega$-*downward* (*or forward*) simulation from $\mathcal{A}$ to $\mathcal{C}$, describes an (abstract) $\mathcal{L}_\omega$-computation which starts using global and local data of $\mathcal{A}$, and ends using global and local data of $\mathcal{C}$. Formally,

Definition 2.2 *A $\mathcal{L}_\omega$-computation dsim is a $\mathcal{L}_\omega$-downward simulation from $\mathcal{A}$ to $\mathcal{C}$, denoted by $dsim : \mathcal{A} \preccurlyeq_\omega \mathcal{C}$, iff*

$$in_\mathcal{A}; dsim \sqsubseteq in_\mathcal{C}$$
$$O_\mathcal{A}; dsim \sqsubseteq dsim; O_\mathcal{C}, \text{ for } O \in \mathcal{O}$$
$$fin_\mathcal{A} \sqsubseteq dsim; fin_\mathcal{C}.$$

The analogous notion of $\mathcal{L}_\omega$-*upward* (or *backward*) *simulation* is defined, by reversing the direction of the above computation. Formally,

Definition 2.3 *A $\mathcal{L}_\omega$-computation usim is a $\mathcal{L}_\omega$-upward simulation from $\mathcal{A}$ to $\mathcal{C}$, denoted by $usim : \mathcal{A} \curlyeqprec_\omega \mathcal{C}$, iff*

$$in_\mathcal{A} \sqsubseteq in_\mathcal{C}; usim$$
$$usim; O_\mathcal{A} \sqsubseteq O_\mathcal{C}; usim, \text{ for } O \in \mathcal{O}$$
$$usim; fin_\mathcal{A} \sqsubseteq fin_\mathcal{C}.$$

In Section 3, we consider simulations for $\mathcal{L}$ and $\mathcal{L}_n$ $(n < \omega)$. Simulation rules provide a useful technique for establishing refinement [8, 3], if we can prove *soundness* (i.e., refinement holds between two data types whenever there is a simulation between them) and *completeness* (i.e., if there is refinement between two data types then there is a simulation between them).

Theorem 2.4 *$\mathcal{L}_\omega$-downward simulations which are right-distributive over $\mathcal{L}_\omega$-constructs are sound, and $\mathcal{L}_\omega$-upward simulations which are left-distributive over $\mathcal{L}_\omega$-constructs are sound.*

Proof: By structural induction.

In a relational model, based on a global state space G, a data type $\mathcal{A}$ is a relational structure

$$\mathcal{A} = (L_\mathcal{A}; I_\mathcal{A}, \{O_\mathcal{A}\}_{O\in\mathcal{O}}, F_\mathcal{A}),$$

where $L_\mathcal{A}$ is the local state space, and $I_\mathcal{A}$, $O_\mathcal{A}$ $(O \in \mathcal{O})$, $F_\mathcal{A}$ are binary relations from initial states to final states representing respectively its initialisation operation, its operations indexed by $\mathcal{O}$ and its finalisation operation. Programs '$prog_\mathcal{A}$' are interpreted as relations '$P_{\mathcal{A}}$' over $L_{\mathcal{A}}$. Note that in this model there is no state of nontermination and nonterminating behaviour cannot be expressed. Non-miraculous operations are total relations, and boundedly nondeterministic operations are image-finite relations. A data type $\mathcal{A}$ is *total* and/or *image-finite* if $I_{\mathcal{A}}$ and $O_{\mathcal{A}}$ $(O \in \mathcal{O})$ are respectively total and/or image-finite. Refinement '$R \sqsubseteq S$' is interpreted as '$dom(R) \subseteq dom(S)$ and on $dom(R)$, $S \subseteq R$', which for total relations reduces to relational containment '$R \supseteq S$'[7]. A binary relation $D \subseteq L_{\mathcal{A}} \times L_{\mathcal{C}}$ is a $D : \mathcal{A} \preccurlyeq_\omega \mathcal{C}$ iff it satisfies the relational interpretation of Definition 2.2; similarly a $U : \mathcal{A} \curlyeqprec_\omega \mathcal{C}$ is a certain binary relation $U \subseteq L_{\mathcal{C}} \times L_{\mathcal{A}}$.

Theorem 2.5 *There is a $D : \mathcal{A} \preccurlyeq_\omega \mathcal{C}$ with $D^{\smile}$ a total function iff there is a total deterministic $U : \mathcal{A} \curlyeqprec_\omega \mathcal{C}$.*

Proof: Let D and U be converse relations – i.e. sDs' iff $s'Us$.

Deterministic simulations have been studied independently in [5], where a deterministic $D : \mathcal{A} \preccurlyeq_\omega \mathcal{C}$ is called a bounded map, and a deterministic $U : \mathcal{A} \curlyeqprec_\omega \mathcal{C}$ a relational morphism.

In a predicate transformer model, which does not express nonterminating behaviour and which is based on a global state space G, a data type $\mathcal{A}$ is a Boolean algebra with operators

$$\mathcal{PA} = (\mathcal{P}(L_{\mathcal{A}}); i_{\mathcal{A}}, \{O_{\mathcal{A}}\}_{O \in \mathcal{O}}, f_{\mathcal{A}}),$$

where $L_{\mathcal{A}}$ is the local state space and $i_{\mathcal{A}}$, $O_{\mathcal{A}}$ $(O \in \mathcal{O})$, $f_{\mathcal{A}}$ are universally conjunctive (i.e. $L_{\mathcal{A}}$-preserving, completely conjunctive) predicate transformers from post- to preconditions representing respectively its initialisation operation, its operations indexed by $\mathcal{O}$ and its finalisation operation. The predicate transformers for non-miraculous operations are also $\emptyset$-preserving, and those for boundedly nondeterministic operations are also continuous. Refinement '$\sqsubseteq$' is the pointwise extension of set inclusion, and '$f; g$' is functional composition '$g \circ f$'. A universally conjunctive d from $\mathcal{P}(L_{\mathcal{C}})$ to $\mathcal{P}(L_{\mathcal{A}})$ is a $d : \mathcal{PA} \preccurlyeq_\omega \mathcal{PC}$ iff it satisfies the predicate transformer interpretation of Definition 2.2; similarly a $u : \mathcal{PA} \curlyeqprec_\omega \mathcal{PC}$ is a certain universally conjunctive predicate transformer $u : \mathcal{P}(L_{\mathcal{A}}) \to \mathcal{P}(L_{\mathcal{C}})$.

Theorem 2.6 *There is a universally disjunctive $d : \mathcal{PA} \preccurlyeq_\omega \mathcal{PC}$ iff there is a universally conjunctive $u : \mathcal{PA} \curlyeqprec_\omega \mathcal{PC}$.*

Proof: Take a Galois pair (d, u) of maps, i.e. $u; d \sqsubseteq id_C$ and $id_A \sqsubseteq d; u$.

3 Soundness and Completeness

In the relational model, we show how soundness and completeness of $\mathcal{L}_\omega$-simulations may be established using simulations for $\mathcal{L}_n$ $(n < \omega)$.

Let $\mathcal{A}$ and $\mathcal{C}$ be conformal data types, and fix $n < \omega$. Define an indexed data type,

$$\mathcal{A}_n = (L_\mathcal{A} \times \mathbf{N}, (I_\mathcal{A})_n, \{(O_\mathcal{A})_n\}_{O \in \mathcal{O}}, (F_\mathcal{A})_n)$$

$$\begin{array}{ll} \text{where} & g(I_\mathcal{A})_n(s,k) \ \text{ iff } \ [gI_\mathcal{A}s \ \wedge \ k = 0]; \\ & (s,k)(O_\mathcal{A})_n(t,k+1) \ \text{ iff } \ [k \leq n \ \wedge \ sO_\mathcal{A}t]; \\ & (t,k+1)(F_\mathcal{A})_n g \ \text{ iff } \ [tF_\mathcal{A}g \ \wedge \ k \leq n]. \end{array}$$

Note that these relations do not contain pairs of states with index $m > n+1$ and hence the indices limit the number of relations that may be composed.

Theorem 3.1 *$\mathcal{A} \sqsubseteq_n \mathcal{C}$ iff $\mathcal{A}_n \sqsubseteq \mathcal{C}_n$.*

Proof: We prove one implication; the other is similar. Assume $\mathcal{A} \sqsubseteq_n \mathcal{C}$. If $P_\mathcal{A} = (P_k)_\mathcal{A}$ for some $k \leq n$,

$$\begin{array}{rcll} (I_\mathcal{A})_n; (P_\mathcal{A})_n; (F_\mathcal{A})_n & = & I_\mathcal{A}; (P_k)_\mathcal{A}; F_\mathcal{A} & \\ & \sqsubseteq & I_\mathcal{C}; (P_k)_\mathcal{C}; F_\mathcal{C} & \text{since } \mathcal{A} \sqsubseteq_k \mathcal{C} \\ & = & (I_\mathcal{C})_n; (P_\mathcal{C})_n; (F_\mathcal{C})_n. & \end{array}$$

If $P_\mathcal{A} = (P_m)_\mathcal{A}$ for some $m > n$ then $(P_\mathcal{A})_n; (F_\mathcal{A})_n = \emptyset$ and so the inclusion holds trivially.

Given a relation $D \subseteq L_\mathcal{A} \times L_\mathcal{C}$ define a relation $D_n \subseteq (L_\mathcal{A} \times \mathbf{N}) \times (L_\mathcal{C} \times \mathbf{N})$ by

$$(s,k)D_n(s',k) \ \text{ iff } \ (k \leq n \ \wedge \ sDs').$$

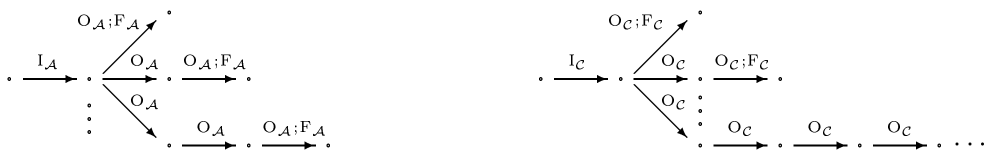
Definition 3.2 *A binary relation $D \subseteq L_\mathcal{A} \times L_\mathcal{C}$ is a $\mathcal{L}_n$-downward simulation from $\mathcal{A}$ to $\mathcal{C}$, denoted by $D : \mathcal{A} \preccurlyeq_n \mathcal{C}$, iff $D_n : \mathcal{A}_n \preccurlyeq_\omega \mathcal{C}_n$.*

Theorem 3.3 $(\forall n)[D_n \sqsubseteq D_{n+1}]$.

Proof: By definition, D_n and D_{n+1} agree on the domain of D_n.

Corollary 3.4 *If $D : \mathcal{A} \preccurlyeq_\omega \mathcal{C}$ then $(\forall n)[D : \mathcal{A} \preccurlyeq_n \mathcal{C}]$.*

The converse of Corollay 3.4 does in general not hold, as the following well-known example from the literature on bisimulations [6] shows. For the data types $\mathcal{A}$ and $\mathcal{C}$,



there is a $D : \mathcal{A} \preccurlyeq_n \mathcal{C}$ for all $n < \omega$, but there is no $D : \mathcal{A} \preccurlyeq_\omega \mathcal{C}$.

Theorem 3.5 *If $\mathcal{A}$ is image-finite and $(\forall n)[D : \mathcal{A} \preccurlyeq_n \mathcal{C}]$ then $D : \mathcal{A} \preccurlyeq_\omega \mathcal{C}$.*

Proof: For $s \in dom(O_\mathcal{A}; D)$,

$$\begin{array}{rcl} sD; O_\mathcal{C} t' & \Leftrightarrow & (\forall n)(\forall k \leq n)[(s,k) D_n ; (O_\mathcal{C})_n (t', k+1)] \\ & \Rightarrow^1 & (\forall n)(\forall k \leq n)[(s,k)(O_\mathcal{A})_n ; D_n (t', k+1)] \\ & \Rightarrow & (\forall n)(\forall k \leq n)(\exists t_n)[(s,k)(O_\mathcal{A})_n (t_n, k+1) \wedge (t_n, k+1) D_n (t', k+1)] \\ & \Rightarrow^2 & (\exists t)(\forall n)(\forall k \leq n)[(s,k)(O_\mathcal{A})_n (t, k+1) \wedge (t, k+1) D_n (t', k+1)] \\ & \Rightarrow & (\exists t)[s O_\mathcal{A} t \wedge t D t'] \end{array}$$

where $\Rightarrow^1$ holds since $D : \mathcal{A} \preccurlyeq_n \mathcal{C}$ and $\Rightarrow^2$ holds since $\mathcal{A}$ is image-finite and $\{D_n\}$ is $\sqsubseteq$-increasing.

Theorem 3.6 (Soundness) *If $D : \mathcal{A} \preccurlyeq_n \mathcal{C}$ then $\mathcal{A} \sqsubseteq_n \mathcal{C}$.*

Proof: By structural induction (c.f. [8, 3]) and Theorem 2.4.

Theorem 3.7 *$\mathcal{L}_n$-soundness for all $n < \omega$ implies $\mathcal{L}_\omega$-soundness. The converse holds for image-finite abstract data types.*

Proof: (of converse):

$$\begin{array}{rcll} (\forall n)[D : \mathcal{A} \preccurlyeq_n \mathcal{C}] & \Rightarrow & D : \mathcal{A} \preccurlyeq_\omega \mathcal{C} & \text{by Theorem 3.5} \\ & \Rightarrow & \mathcal{A} \sqsubseteq_\omega \mathcal{C} & \text{by soundness for } \mathcal{L}_\omega \\ & \Rightarrow & (\forall n)[\mathcal{A} \sqsubseteq_n \mathcal{C}] & \text{by definition of } \mathcal{L}_\omega. \end{array}$$

Completeness can be disproved by identifying *spade* and *fork shapes.*

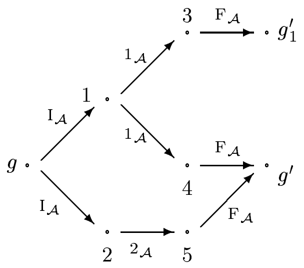
Definition 3.8 *Conformal $\mathcal{A}$ and $\mathcal{C}$ have a spade and a fork shape of depth n, if there are complete programs of $\mathcal{A}$ which exhibit a spade shape (as on the left) and the corresponding complete programs of $\mathcal{C}$ exhibit a fork shape (as on the right),*

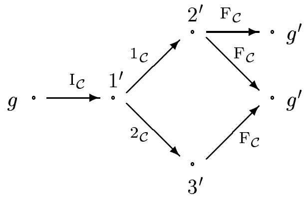
where $P1$ is a composition of $n < \omega$ operations and $P2, P3$ are each a composition of $m < \omega$ operations.

Theorem 3.9 *If conformal $\mathcal{A}$ and $\mathcal{C}$ have a spade and fork shape of depth at least n then there is no $D : \mathcal{A} \preccurlyeq_k \mathcal{C}$ for all $k \geq n$.*

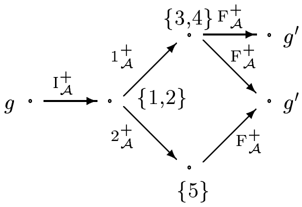
Proof: By definition, there exist $(s, n), (s', n), (t', n+1)$ such that for all $(t, n+1)$, $s' O_\mathcal{C} t' \wedge \neg s O_\mathcal{A} t$.

For example, let $\mathcal{A}$ and $\mathcal{C}$ be the data types,





where $1_\mathcal{A}, 2_\mathcal{A}$ denote operations of $\mathcal{A}$ and $1_\mathcal{C}, 2_\mathcal{C}$ the corresponding operations of $\mathcal{C}$. Then $\mathcal{A} \sqsubseteq \mathcal{C}$ but there is no $D : \mathcal{A} \preccurlyeq_n \mathcal{C}$ for all $n \leq \omega$ since $\mathcal{A}$ and $\mathcal{C}$ have a spade and fork shape of depth 0 and of depth 1. We can lift $\mathcal{A}$ to a deterministic data type, equivalent to $\mathcal{A}$ and from which there is a downward simulation to $\mathcal{C}$.



According to our approach, completeness can be established by a more intuitive construction (than that in [8]) of a downward simulation in terms of its approximations.

First, we define a subset of $\mathcal{P}(L_\mathcal{A})$ which includes only those sets of local states reachable from global states via the operations of $\mathcal{A}$. For each $g \in G$, let $\mathcal{P}_0(L_\mathcal{A}, g)$ consist of the sets

$$S = \{s \mid gI_\mathcal{A}s\},$$

and $\mathcal{P}_n(L_\mathcal{A}, g)$ $(0 < n < \omega)$ consist of the sets T such that for some $S \in \mathcal{P}_{n-1}(L_\mathcal{A}, g)$ and some $O \in \mathcal{O}$,

$$T = \{t \mid (\exists s \in S)[sO_\mathcal{A}t]\}.$$

Let $\mathcal{P}_\omega(L_\mathcal{A}, g) = \bigcup_{n<\omega} \mathcal{P}_n(L_\mathcal{A}, g)$ and $\mathcal{P}_\omega(L_\mathcal{A}) = \bigcup_{g \in G} \mathcal{P}_\omega(L_\mathcal{A}, g)$.

Second, we invoke a translation which defines from a binary relation R over sets A and B its *power relation* [1] $R^+ \subseteq \mathcal{P}(A) \times \mathcal{P}(B)$ where

$$XR^+Y \quad \text{iff} \quad (\forall y \in Y)(\exists x \in X)[xRy].$$

Applying this translation to each relation of $\mathcal{A}$, we obtain its *power relation data type*, denoted by $\mathcal{A}^+$,

$$\mathcal{A}^+ = (\mathcal{P}(L_\mathcal{A}); {I_\mathcal{A}}^+, \{{O_\mathcal{A}}^+\}_{O \in \mathcal{O}}, {F_\mathcal{A}}^+),$$

where ${I_\mathcal{A}}^+ \subseteq G \times \mathcal{P}(L_\mathcal{A})$ and ${F_\mathcal{A}}^+ \subseteq \mathcal{P}(L_\mathcal{A}) \times G$ are the 'partial' power relations given by

$$g{I_\mathcal{A}}^+S \quad \text{iff} \quad (\forall s \in S)[gI_\mathcal{A}s] \qquad \text{and} \qquad T{F_\mathcal{A}}^+g \quad \text{iff} \quad (\exists t \in T)[tF_\mathcal{A}g].$$

Restricting $\mathcal{A}^+$ to $\mathcal{P}_\omega(L_\mathcal{A})$, we obtain a total deterministic data type $\mathcal{A}^+_\omega$, which can be extended to the total deterministic data type $\mathcal{A}^+_d$ of [8] with local state space $\mathcal{P}(L_\mathcal{A})$, by up-closing each relation of $\mathcal{A}^+_\omega$ under non-empty subsets of $L_\mathcal{A}$.

Third, for each $n < \omega$, we define a relation D_n from $(\mathcal{P}_\omega(L_\mathcal{A}) \times \mathbf{N})$ to $(L_\mathcal{C} \times \mathbf{N})$ by

$$(S,k)D_n(s',k) \ \text{ iff } \ k \leq n \wedge (\forall (P_n)_\mathcal{A})(\forall g')[s'(P_n)_\mathcal{C}; F_\mathcal{C} g' \Rightarrow S(P_n)^+_\mathcal{A}; F_\mathcal{A}{}^+ g']$$

and define $D \subseteq \mathcal{P}_\omega(L_\mathcal{A}) \times L_\mathcal{C}$ by

$$SDs' \ \text{ iff } \ (\forall n)(\forall k \leq n)[(S,k)D_n(s',k)].$$

Note that by definition of $\mathcal{L}_\omega$,

$$SDs' \ \text{ iff } \ (\forall P_\mathcal{A})(\forall g')[s' P_\mathcal{C}; F_\mathcal{C} g' \Rightarrow P^+_\mathcal{A}; F_\mathcal{A}{}^+ g'].$$

Up-closing D under non-empty subsets of $L_\mathcal{A}$ yields the relation from $\mathcal{P}(L_\mathcal{A})$ to $L_\mathcal{C}$ constructed in [8].

Lemma 3.10 $(\forall n)[D : \mathcal{A}^+_\omega \preccurlyeq_n \mathcal{C}]$.

Proof: By a proof similar to that in [8], for all $n < \omega$ $D : \mathcal{A}^+_\omega \preccurlyeq_n \mathcal{C}$. By Theorem 3.5, $D : \mathcal{A}^+_\omega \preccurlyeq_\omega \mathcal{C}$.

Theorem 3.11 (Completeness) *If $\mathcal{A} \sqsubseteq_\omega \mathcal{C}$ then $U : \mathcal{A} \curlyeqprec_\omega \mathcal{A}^+_\omega$ and $D : \mathcal{A}^+_\omega \preccurlyeq_\omega \mathcal{C}$.*

Proof: Trivially, $\in^\smile : \mathcal{A} \curlyeqprec_\omega \mathcal{A}^+_\omega$ and, by Lemma 3.10, $D : \mathcal{A}^+_\omega \preccurlyeq_\omega \mathcal{C}$.

Theorem 3.12 *$\mathcal{L}_\omega$-completeness implies $\mathcal{L}_n$-completeness for all $n < \omega$. The converse holds for image-finite abstract data types.*

Proof: Similar to Lemma 3.10.

If our relational model includes a state of nontermination and hence *abort* and iteration, then [8] surjective $\mathcal{L}_\omega$-downward simulations are sound and complete for $\mathcal{L}$, and total finitary $\mathcal{L}_\omega$-upward simulations are sound and complete for $\mathcal{L}$ and total finitary abstract data types.

4 Jónsson/Tarski duality for data types

A data type may, as described in Section 2, be viewed both relationally and in terms of predicate transformers. Indeed Jónsson/Tarski duality [9] can be invoked to show that these views are intertranslatable.

First, we recall Stone's representation theorem [14]: any set can be embedded in the set of ultrafilters of its power set, and any Boolean algebra can be represented as a field of sets. Intuitively, states are identified with certain sets of properties, and properties with sets of states.

Consider a relational data type $\mathcal{A} = (L_\mathcal{A}; I_\mathcal{A}, \{O_\mathcal{A}\}_{O \in \mathcal{O}}, F_\mathcal{A})$. Then to turn $\mathcal{P}(L_\mathcal{A})$ into a predicate transformer data type we observe that any binary relation R over sets A and B gives rise to a Galois pair of *power operators* over their power sets [1], the universally conjunctive map $R^\uparrow : \mathcal{P}(B) \to \mathcal{P}(A)$ where

$$R^\uparrow(Y) = \{x \mid (\forall y)[xRy \Rightarrow y \in Y]\}$$

and the universally disjunctive map $(R^\smile)^{\uparrow d} : \mathcal{P}(A) \to \mathcal{P}(B)$ where $R^\smile$ is the converse of R and $R^{\uparrow d}$ is the dual of $R^\uparrow$ given by

$$R^{\uparrow d}(Y) = \{x \mid (\exists y \in Y)[xRy]\}.$$

If we think of sets as predicates then the mappings $R^\uparrow$ may be viewed as predicate transformers from post- to preconditions. Applying this translation to each relation of $\mathcal{A}$, we obtain its *power operator data type* $\mathcal{A}^\uparrow$.

On the other hand, consider a predicate transformer data type

$$\mathcal{PA} = (\mathcal{P}(L_\mathcal{A}); i_\mathcal{A}, \{O_\mathcal{A}\}_{O \in \mathcal{O}}, f_\mathcal{A}).$$

Let $\mathcal{U}(\mathcal{P}(L_\mathcal{A}))$ be the set of all principal ultrafilters in $\mathcal{P}(L_\mathcal{A})$. To turn $\mathcal{U}(\mathcal{P}(L_\mathcal{A}))$ into a relational data type, we invoke the Jónsson/Tarski translation which defines from a universally conjunctive map $g : \mathcal{P}(B) \to \mathcal{P}(A)$, a relation $g^\downarrow \subseteq \mathcal{U}(\mathcal{P}(A)) \times \mathcal{U}(\mathcal{P}(B))$ by

$$\mathsf{X} g^\downarrow \mathsf{Y} \ \text{ iff } \ x \in g(\{y\}),$$

where X and Y are the principal ultrafilters generated respectively by x and y. Applying this translation to each predicate transformer of $\mathcal{PA}$, we obtain its *ultrafilter data type* $\mathcal{PA}^\downarrow$.

Together with Stone's result, these translations yield a bijective correspondence between relational and predicate transformer data types. (The proof is similar to that in [13].)

Theorem 4.1 *Any relational data type is isomorphic to the principal ultrafilter data type of its power operator transformer data type. Dually, any predicate transformer data type is isomorphic to the power operator data type of its principal ultrafilter data type.*

Corollary 4.2 *For conformal $\mathcal{A}$ and $\mathcal{C}$, $\mathcal{PA}$ and $\mathcal{PC}$,*

(a) *if $\mathcal{A} \sqsubseteq \mathcal{C}$ then $\mathcal{A}^\uparrow \sqsubseteq \mathcal{C}^\uparrow$;*

(b) *if $D : \mathcal{A} \preccurlyeq_\omega \mathcal{C}$ then $D^\uparrow : \mathcal{A}^\uparrow \preccurlyeq_\omega \mathcal{C}^\uparrow$;*

(c) *if* $\mathcal{PA} \sqsubseteq \mathcal{PC}$ *then* $\mathcal{PA}^{\downarrow} \sqsubseteq \mathcal{PC}^{\downarrow}$;

(d) *if* $d : \mathcal{PA} \preccurlyeq_{\omega} \mathcal{PC}$ *then* $d^{\downarrow} : \mathcal{PA}^{\downarrow} \preccurlyeq_{\omega} \mathcal{PC}^{\downarrow}$.

Proof: By definition $(\cdot)^{\uparrow}$ and $(\cdot)^{\downarrow}$ distribute over ';' and preserve '$\sqsubseteq$' (for both total and partial relations).

These translations extend to properties of simulations and hence to sound $\mathcal{L}$-simulations. They also generalise, as below, to a translation for complete $\mathcal{L}_{\omega}$-downward simulations.

From any up-closed binary relation $R \subseteq C \times \mathcal{P}(A)$, we may define a monotone mapping $R^{\Uparrow} : \mathcal{P}(A) \to \mathcal{P}(C)$ by

$$R^{\Uparrow}(Q) = \{x \mid (\exists V)[xRV \wedge V \subseteq Q]\}.$$

From any monotone mapping $g : \mathcal{P}(A) \to \mathcal{P}(C)$ we may in turn define an up-closed binary relation $g^{\Downarrow} \subseteq C \times \mathcal{P}(A)$ by

$$xg^{\Downarrow}Q \quad \text{iff} \quad x \in g(Q).$$

These lowering and lifting actions cancel each other, and as a consequence we obtain a reformulation of the factoring theorem for monotone mappings, discovered in [4].

Theorem 4.3 *Any monotone mapping can be written as a composition of a universally conjunctive and a universally disjunctive mapping.*

Proof: For any monotone mapping $u : \mathcal{P}(A) \to \mathcal{P}(C)$,

$$\begin{array}{rcll} x \in u(Q) & \Leftrightarrow & x \in u^{\Downarrow\Uparrow}(Q) & \text{by definition} \\ & \Leftrightarrow & (\exists P)[xu^{\Downarrow}P \wedge P \subseteq Q] & \text{by definition of } u^{\Downarrow\Uparrow} \text{ from } u^{\Downarrow} \\ & \Leftrightarrow & x \in (u^{\Downarrow})^{\uparrow d}; (\in^{\smile})^{\uparrow}(Q) & \text{by definition of } (\cdot)^{\uparrow}, (\cdot)^{\uparrow d}, (\cdot)^{\smile}. \end{array}$$

Theorem 4.4 *For conformal* $\mathcal{A}$ *and* $\mathcal{C}$, $\mathcal{PA}$ *and* $\mathcal{PC}$,

(a) *if there is an up-closed* $D : \mathcal{A}_d^{+} \preccurlyeq_{\omega} \mathcal{C}$ *then there is a monotone* $d : \mathcal{A}^{\uparrow} \preccurlyeq_{\omega} \mathcal{C}^{\uparrow}$;

(b) *if there is a monotone* $u : \mathcal{PA} \preccurlyeq_{\omega} \mathcal{PC}$ *then there is an up-closed* $U : (\mathcal{PA}^{\downarrow})_d^{+} \preccurlyeq_{\omega} \mathcal{PC}^{\downarrow}$.

Proof: For (a), l et $D : \mathcal{A}_d^{+} \preccurlyeq_{\omega} \mathcal{C}$ be up-closed. By Corollary 4.2 and Theorem 2.6,

$$(D^{\smile})^{\uparrow d} : \mathcal{A}_d^{+\uparrow} \preccurlyeq_{\omega} \mathcal{C}^{\uparrow}.$$

Trivially, $(\in^{\smile})^{\uparrow} : \mathcal{A}^{\uparrow} \preccurlyeq_{\omega} \mathcal{A}_d^{+\uparrow}$, and hence there is a monotone $D^{\smile\Uparrow} : \mathcal{A}^{\uparrow} \preccurlyeq_{\omega} \mathcal{C}^{\uparrow}$.
For (b) suppose $u : \mathcal{PA} \preccurlyeq_{\omega} \mathcal{PC}$ is monotone. Invoking Corollary 4.2 and Theorem 2.6, we show that $u^{\Downarrow\uparrow d} : (\mathcal{PA}^{\downarrow})_d^{+\uparrow} \preccurlyeq_{\omega} \mathcal{PC}$ by proving the condition on operations; the other proofs are similar.

$$\begin{array}{rcll} u^{\Downarrow\uparrow d}; (O_{\mathcal{A}})_d^{+\uparrow}(\mathsf{Q}) & = & u^{\Downarrow\uparrow d}; (O_{\mathcal{A}})_d^{+\uparrow d}(\mathsf{Q}) & \text{since } (O_{\mathcal{A}})_d^{+} \text{ is deterministic} \\ & \subseteq & \cup_{Q \in \mathsf{Q}}(u; O_{\mathcal{A}}{}^{\uparrow})(Q) & \text{by definition and residuation} \\ & \subseteq & \cup_{Q \in \mathsf{Q}}(O_{\mathcal{C}}{}^{\uparrow}; u)(Q) & \text{since } u \text{ is an upward simulation} \\ & \subseteq & O_{\mathcal{C}}{}^{\uparrow}(\cup_{Q \in \mathsf{Q}} u(Q)) & \text{by monotonicity of } O_{\mathcal{C}}{}^{\uparrow} \\ & = & O_{\mathcal{C}}{}^{\uparrow}; u^{\Downarrow\uparrow d}(\mathsf{Q}) & \text{by definition of } u^{\Downarrow\uparrow d} \text{ from } u^{\Downarrow}. \end{array}$$

5 Conclusion

The essence of our approach in this paper is that for image-finite data types a $\mathcal{L}_\omega$-downward simulation is the limit of its finite approximations. These approximations are in turn simulations for sublanguages of $\mathcal{L}_\omega$, and provide a systematic way of proving soundness and completeness of $\mathcal{L}_\omega$-simulations and of $\mathcal{L}$-simulations.

In the relational model completeness is proved by invoking a power constuction and composing an upward simulation from the abstract data type to its power data type with a downward simulation from this data type to the concrete data type. In the predicate transformer model this power construction and composition is hidden by a factoring theorem, and consequently a single monotone upward simulation suffices to establish completeness. These simulations and their properties are indeed intertranslatable via Jónsson/Tarski duality, and hence constructed at the same level of generality.

Acknowledgements: I am grateful for funding from the EPSRC and I thank Jeff Sanders for his careful reading and helpful comments.

References

[1] C. Brink. Power structures. *Algebra Universalis* **30** (1993) p 177–216.

[2] E.W. Dijkstra. *A Discipline of Programming* (Prentice Hall, Englewood Cliffs, NJ, 1976).

[3] P.H.B. Gardiner and C.C. Morgan. A single complete rule for data refinement. *Formal Aspects of Computing* **5** (1993) p 367–382.

[4] P.H.B. Gardiner, C.E. Martin and O. de Moor. An algebraic construction of predicate transformers. *Science of Computer Programming* **22** (1-2) (1994) p 21–44.

[5] R. Goldblatt. Varieties of complex algebras. *Annals of Pure and Applied Logic* **44** (1989) p 173–242.

[6] M. Hennessy and R. Milner. Algebraic laws for nondeterminism and concurrency *Journal of the ACM* **32** (1985) p 137–161.

[7] C.A.R. Hoare *et al.* Laws of Programming. *Communications of the ACM* **30** (1987) p 672–686.

[8] C.A.R. Hoare, He Jifeng and J.W. Sanders. Prespecification in data refinement. *Information Processing Letters* **25** (1987) p 71–76.

[9] B. Jónsson and A. Tarski. Boolean algebras with operators I. *American Journal of Mathematics* **73** (1951) p 891–939.

[10] R. Milner. An algebraic definition of simulation between programs. In: *Proceedings of the 2nd Joint Conference on AI*, British Computer Society (1971) p 481–489.

[11] T. Nipkow. Non-deterministic data types: models and implementations. *Acta Informatica* **22** (1986) p 629–661.

[12] D. Park. Concurrency and automata on infinite sequences. *Lecture Notes in Computer Science* **154** (1981) p 561–572.

[13] I. Rewitzky and C. Brink. Predicate transformers as power operations. *Formal Aspects of Computing* **7** (2) (1995) p 169–182.

[14] M.H. Stone. The theory of representations for Boolean algebras. *Transactions of the AMS* **40** (1956) p 37–111.