Part I

Basic dualities

Chapter 3

The weakest precondition calculus

The role of a *sequential program* is to produce a final result at the end of a terminating computation. Computations may possibly be non-deterministic and also fail to terminate. The main characteristic of sequential programs is that no interaction with its environment is possible. Programs written in classical programming languages like Pascal are examples of sequential programs. Different semantics for this type of programs (and their relationships) are our main interest in this first part.

The *semantics* of a programming language $\mathcal{L}$ is a function which assigns to each program in $\mathcal{L}$ its meaning, that is, an element of a domain of meanings chosen for modeling the computations specified by the program. There are different approaches to the definitions of the semantic function and of the semantic domain.

The *operational* approach is intended to specify the meaning of a program in terms of the steps performed by an abstract machine when executing it. Formally, a transition relation on the configurations of an abstract machine is specified [94,161]: a transition from a configuration to another one represents one atomic step of a computation. Then the semantic function is defined in terms of the transition relation. A computation of a program may fail to terminate if it contains an infinite transition sequence. A computation deadlocks if there is a configuration reached by the computation from which no transition is possible. The operational view of a program on the one hand corresponds often to its intuitive meaning, but, on the other hand, it is not always abstract enough to be computationally useful since it might require a rather detailed

and intricate analysis.

Another approach to semantics is the *denotational* one [177,142,186,81]: first provide an appropriate semantic domain according to the principle that program constructs denote values, and then define the semantic function in such way that the meaning of each syntactic construction of a program is given in terms of the meanings of its constituent parts. In particular fixed point techniques are needed to deal with recursion. For sequential programs this results in the relation between input and output values. Thus the most simple abstract denotational domain for sequential programs is that of all functions from a starting state space (the set of all admissible inputs values) to a final state space (the set of all possible output values). The semantics of a program is a function, which we call *state transformer*. In order to take into account non-termination of programs it is a natural step to consider state transformers employing complete partial orders with a bottom element—a fictitious state representing non-termination. Within this framework, non-determinism can be handled using powerdomains. The state transformer model reflects closely the operational view of a program, but abstracts from the intermediate configurations.

The *axiomatic* approach has different aims from the operational and the denotational ones: proving program correctness, analyzing program properties, and synthesizing correct programs from formal specifications [56,12,58,17]. Informally, a sequential program is correct if it satisfies the intended relation between input values and output value. Program correctness is expressed by statements of the form $\{P\}S\{Q\}$, where S is a sequential program, P is a predicate on the set of input values (precondition) and Q is a predicate on the set of output values (postcondition) [101]. The precondition P describes the initial input values in which the program S is started, and the postcondition Q describes the set of the desirable output values. More abstractly, correctness statements can be defined with the weakest precondition and the weakest liberal precondition: programs can be identified with functions, called *predicate transformers*, from predicates on the set of all possible output values to predicates on the set of all admissible input values. The weakest (liberal) precondition calculus was introduced by Dijkstra [56] as a mathematical tool for reasoning about the partial and total correctness of programs, and it has been further developed in [82,58,98]. This predicate transformer model is called axiomatic because it relies only on algebraic properties of predicates (described for example in [86]).

In this chapter we start by introducing the syntax of a sequential language. Then we define three different state transformer semantic domains. Accord-

ingly, three state transformer semantics for our language are introduced and related. We define two predicate transformer semantics, one by taking into account the possibility of non-termination, and another one by not doing so. State transformer semantics and predicate transformer semantics will be proved to be equivalent. We conclude the chapter with a formal treatment of a backtrack operator in the weakest precondition calculus.

3.1 The sequential language $\mathcal{L}_0$

We begin by introducing a simple sequential language $\mathcal{L}_0$ which is inspired by Dijkstra's language of guarded commands [56]. The language constructors are assignment, conditional, non-deterministic choice and sequential composition. The language allows for recursion by means of procedure variables. Dijkstra's guarded commands, conditionals and recursive combinators can be expressed in terms of the basic constructors of $\mathcal{L}_0$.

All the constructors of the language are well-known. The free occurrence of guards as a conditional is already present in Hoare [103]. The non-deterministic choice is studied, for example, by De Bakker in [20]. More generally, the language $\mathcal{L}_0$ is a slight variation of Hesselink's calculus of commands [95].

To define the language, we need as basic blocks the sets $(v \in)\, \mathit{IVar}$ of (individual) variables, $(e \in)\, \mathit{Exp}$ of expressions, $(b \in)\, \mathit{BExp}$ of Boolean expressions, and $(x \in)\, \mathit{PVar}$ of procedure variables, respectively. For a fixed set of values Val, the set of states $(s, t \in)\, \mathtt{St}$ is given by $\mathtt{St} = \mathit{IVar} \to \mathit{Val}$. As usual, for every state $s \in \mathtt{St}$, individual variable $v \in \mathit{IVar}$ and value $z \in \mathit{Val}$, $s[z/v]$ denotes the state which evaluates to $s(v')$ for every $v' \neq v$ and evaluates to z otherwise. Also, we postulate valuations

$$\mathcal{E}_{\mathcal{V}} : \mathit{Exp} \to (\mathtt{St} \to \mathit{Val}) \text{ and } \mathcal{B}_{\mathcal{V}} : \mathit{BExp} \to \mathcal{P}(\mathtt{St}).$$

These functions provide, in a rather abstract way, the semantics of expressions and Boolean expressions. Clearly $\mathcal{E}_{\mathcal{V}}(e)(s) = z$ means that the expression e in a state s has value z, and, similarly, $s \in \mathcal{B}_{\mathcal{V}}(b)$ means that the Boolean expression b is true in a state s. Notice that for simplicity we assume that the evaluation of an expression and of a Boolean expression is deterministic and always terminates.

The language below has assignment ':=', conditional '$b\to$', sequential composition ';', choice '$\Box$', and recursion through procedure variables. Its syntax is

defined as follows.

Definition 3.1.1 (i) *The set* $(S \in)\, Stat_0$ *of* statements *is given by*

$$S ::= v \;:=\; e \mid b\!\rightarrow \mid x \mid S \;;\; S \mid S \;\Box\; S\,.$$

(ii) *The set* $(d \in)\, Decl_0$ *of* declarations *is defined by* $Decl_0 = PVar \rightarrow Stat_0$.

(iii) *The language* $\mathcal{L}_0$ *is given by* $Decl_0 \times Stat_0$.

The computational intuition behind assignments is as usual. The conditional '$b\!\rightarrow$' deadlocks in a state in which the Boolean expression 'b' does not evaluate to true and acts as a skip otherwise. We assume deadlock is not signaled. The sequential composition executes the first component and then it executes the second component. The choice executes one of its components (the choice as to which component is taken may be made by an implementation or, for non-sequential languages, may be forced by some external factor). The intended meaning of a procedure variable is body replacement.

We do not give an operational semantics for the language $\mathcal{L}_0$, since we will not deal with the connection between the operational and denotational semantics (which, of course, is an important topic [160,22,23]). We concentrate on state transformer and predicate transformer models, and we shall rely on our computational intuition when formulating the semantic function.

3.2 State transformer models

In the state transformer approach programs are denoted by functions that relate an input state s to the outcomes of all the computations of the program when started in s. There are two important aspects to be considered. There may be input states s for which the program deadlocks or fails to terminate. In the first case, since no outcome is present, the input s is related to the empty set. This is in accordance with the fact that if a program at input s can either deadlock or produce some outputs then there is no reason to signal deadlock as a result of a computation. In the second case we need to introduce a special value—usually $\perp$—to which a non-terminating computation is mapped.

Some difficulties arise when we consider non-deterministic programs. Suppose we have a procedure variable $x \in PVar$ declared as $d(x) = v := 0 \;;\; x$, and let us consider the programs

– $P_1 = \langle d, v \;:=\; 1 \rangle$

- $P_2 = \langle d, x \rangle$
- $P_3 = \langle d, v := 1 \,\square\, x \rangle$.

While program P_1 always terminates when activated, an execution of the program P_2 gets stuck in a loop. An execution of the program P_3 consists of either executing the program P_1 or the program P_2. Which of these three programs should be considered equivalent by a state transformer semantics?

One view is to consider equivalent those programs which have computations that may fail to terminate since nothing can be guaranteed for them. Hence the program P_3 should be identified with the program P_2 and it should differ from the program P_1.

Another view is to identify those programs that have the same sets of outcomes, if any. Then the program P_1 should be identified with the program P_3, and both should be different from the program P_2.

Finally, another view is to consider what actually happens: all three programs are different. Below we give three state transformer domains corresponding to these three views.

Smyth state transformers

Let X be the set of inputs and Y be the set of all possible outcomes of a class of programs we consider. Computations that are possibly non-terminating are identified (since nothing can be guaranteed of any of them) and mapped to $Y_\perp = Y \cup \{\perp\}$. Computations that deadlock are mapped to the empty set.

Definition 3.2.1 *The set of* Smyth state transformers *from a set X to a set Y is defined by*

$$ST_S(X, Y) = X \to (\mathcal{P}(Y) \cup \{Y_\perp\}).$$

In general, Smyth state transformers are ordered by the pointwise extension of the superset order, that is, for $\sigma, \tau \in ST_S(X, Y)$

$$\sigma \leq \tau \text{ if and only if } \forall x \in X\colon \sigma(x) \supseteq \tau(x).$$

The above order can be justified as follows: the smaller the set of outcomes of a program the more can be guaranteed of it. Smyth state transformers form a

poset with a least element given by the function mapping every $x \in X$ to $Y_\perp$ (corresponding to the program which always fails to terminate, and for which nothing at all can be guaranteed).

Not all Smyth state transformers are 'reasonable' denotations of programs. In particular, we may wish to consider only programs which are finitely non-deterministic:

$$ST_S^{fin}(X, Y) = X \to (\mathcal{P}_{fin}(Y) \cup \{Y_\perp\}),$$

where $\mathcal{P}_{fin}(Y)$ consists of the finite subsets of Y.

Lemma 3.2.2 *For every set X and Y, both $ST_S(X, Y)$ and $ST_S^{fin}(X, Y)$ are complete partial orders.*

Proof. Since the function $\lambda x \in X.Y_\perp$ is in both $ST_S(X, Y)$ and $ST_S^{fin}(X, Y)$, it is their least element. Assume now $\mathcal{V}$ is a directed set of functions in $ST_S(X, Y)$. It is easy to see that

$$(3.1)\lambda x \in X. \bigcap\{\sigma(x) \mid \sigma \in \mathcal{V}\}$$

is the least upper bound of $\mathcal{V}$ in $ST_S(X, Y)$. If every $\sigma \in \mathcal{V}$ is in $ST_S^{fin}(X, Y)$ then $\sigma(x)$ is either a finite set or $\{Y_\perp\}$. Thus also

$$\bigcap\{\sigma(x) \mid \sigma \in \mathcal{V}\}$$

is a finite set or $\{Y_\perp\}$ for every $x \in X$. It follows that (3.1) is the least upper bound of $\mathcal{V}$ also in $ST_S^{fin}(X, Y)$. □

An alternative way to prove that $ST_S^{fin}(X, Y)$ is a complete partial order is to define it as the set of all functions from X to $\mathcal{S}(Y_\perp)^\top$, the Smyth powerdomain with emptyset (added as a top element) of the flat cpo $Y_\perp$.

There are two basic operators for Smyth state transformers which can be used as the semantical counterpart of the syntactical operators of $\mathcal{L}_0$.

Definition 3.2.3 *Let X, Y and Z be three sets. Define, for every $x \in X$, the* union function $\Box : ST_S(X, Y) \times ST_S(X, Y) \to ST_S(X, Y)$ *by*

$$(\sigma_1 \Box \sigma_2)(x) = \sigma_1(x) \cup \sigma_2(x),$$

and the composition function $;: ST_S(X, Y) \times ST_S(Y, Z) \to ST_S(X, Z)$ *by*

$$(\sigma_1 ; \sigma_2)(x) = \begin{cases} Y_\perp & \text{if } \perp \in \sigma_1(x) \text{ or} \\ & \exists y \in \sigma_1(x) \colon \perp \in \sigma_2(y) \\ \bigcup\{\sigma_2(y) \mid y \in \sigma_1(x)\} & \text{otherwise.} \end{cases}$$

These functions are monotone in both their arguments. Moreover, if σ_1 and σ_2 are in $ST_S^{fin}(X, Y)$, then also $\sigma_1 \Box \sigma_2$ is in $ST_S^{fin}(X, Y)$. Similarly, because the finite union of finite sets is a finite set, if σ_1 is an element of $ST_S^{fin}(X, Y)$ and σ_2 is an element of $ST_S^{fin}(Y, Z)$ then their composition $\sigma_1 ; \sigma_2$ is an element of $ST_S^{fin}(X, Z)$.

Once we have defined the semantical operators which will denote the syntactic operators ';' and '$\Box$' of the language $\mathcal{L}_0$, we have almost all ingredients to define a state transformer semantics for $\mathcal{L}_0$ using $ST_S(\mathtt{St}, \mathtt{St})$ as semantic domain: we have only to define the semantics for the atomic commands '$v := e$' and '$b\to$', and for the procedure variables 'x'.

Definition 3.2.4 *The semantic function $St_S[\![\cdot]\!]$ is defined as the least function in $\mathcal{L}_0 \to ST_S(\mathtt{St}, \mathtt{St})$ such that, for all $s \in S$,*

$$\begin{aligned} St_S[\![\langle d, v := e\rangle]\!](s) &= \{s[\mathcal{E}\nu(e)(s)/v]\}, \\ St_S[\![\langle d, b\to\rangle]\!](s) &= \begin{cases} \{s\} & \text{if } s \in \mathcal{B}\nu(b) \\ \emptyset & \text{otherwise,} \end{cases} \\ St_S[\![\langle d, x\rangle]\!](s) &= St_S[\![\langle d, d(x)\rangle]\!](s), \\ St_S[\![\langle d, S_1 ; S_2\rangle]\!](s) &= (St_S[\![\langle d, S_1\rangle]\!] ; St_S[\![\langle d, S_2\rangle]\!])(s), \\ St_S[\![\langle d, S_1 \Box S_2\rangle]\!](s) &= (St_S[\![\langle d, S_1\rangle]\!] \Box St_S[\![\langle d, S_2\rangle]\!])(s). \end{aligned}$$

The well-definedness of the above semantics can be justified as follows. The semantics $St_S[\![\cdot]\!]$ can be obtained as the least fixed point of a higher order transformation.

Lemma 3.2.5 *Let $F \in Sem_S = \mathcal{L}_0 \to ST_S(\mathtt{St}, \mathtt{St})$ and define the function*

$\Psi_S : Sem_S \to Sem_S$ *inductively, for all* $s \in \mathtt{St}$*, by*

$$\begin{aligned}
\Psi_S(F)(\langle d, v := e\rangle)(s) &= \{s[\mathcal{E}\nu(e)(s)/v]\},\\
\Psi_S(F)(\langle d, b\rightarrow\rangle)(s) &= \begin{cases} \{s\} \text{ if } s \in \mathcal{B}\nu(b) \\ \emptyset \quad \text{otherwise} \end{cases}\\
\Psi_S(F)(\langle d, x\rangle)(s) &= F(\langle d, d(x)\rangle)(s),\\
\Psi_S(F)(\langle d, S_1 \,;\, S_2\rangle)(s) &= (\Psi_S(F)(\langle d, S_1\rangle) \,;\, \Psi_S(F)(\langle d, S_2\rangle))(s),\\
\Psi_S(F)(\langle d, S_1 \,\square\, S_2\rangle)(s) &= (\Psi_S(F)(\langle d, S_1\rangle) \square \Psi_S(F)(\langle d, S_2\rangle))(s),
\end{aligned}$$

Then Ψ_S *is well-defined, monotone, and the function* $St_S[\![\cdot]\!]$ *defined in Definition 3.2.4 is the least fixed point of* Ψ_S*.*

Proof. Well-definedness of Ψ_S is readily checked. To prove monotonicity of Ψ_S assume $F_1 \leq F_2$ in Sem_S. We show that $\Psi_S(F_1)(\langle d, S\rangle) \leq \Psi_S(F_2)(\langle d, S\rangle)$ for any program $\langle d, S\rangle$ by induction on the structure of S. The base cases are immediate, and for the cases when $S \equiv S_1 \,\square\, S_2$ or $S \equiv S_1 \,;\, S_2$ we use induction and the fact that both the union function '$\square$' and the composition function ';' are monotone in each argument.

Finally, since $ST_S(\mathtt{St}, \mathtt{St})$ is a cpo, Sem_S is also a cpo. Thus, by Proposition 2.2.3 the function Ψ_S has a least fixed point, which, from Definition 3.2.4, is $St_S[\![\cdot]\!]$. $\square$

By structural induction on the statement S, and because $ST_S^{fin}(\mathtt{St}, \mathtt{St})$ is closed under the union function '$\square$' and the composition function ';', it follows that $St_S[\![\langle d, S\rangle]\!] \in ST_S^{fin}(\mathtt{St}, \mathtt{St})$ for every program $\langle d, S\rangle$ in $\mathcal{L}_0$.

Hoare state transformers

Next we consider a domain of state transformers which can be used for identifying programs only on the basis of their sets of outcomes, if any. The main difference with the Smyth state transformers is that now we do not wish to record non-termination. Deadlocking computations are mapped to the empty set, as before.

Definition 3.2.6 *The set of* Hoare state transformers *from a set* X *to a set* Y *is defined by*

$$ST_H(X, Y) = X \to \mathcal{P}(Y).$$

Alternatively, Hoare state transformers can be defined as the cpo of all functions from X to $(\mathcal{H}(Y_\perp))_\perp$, the Hoare powerdomain with emptyset (added as a bottom element) of the flat cpo $Y_\perp$. We prefer our definition above since its conceptually simpler (no extra bottom elements $\perp$ have to be added to Y).

Since Hoare state transformers do not record non-termination, infinite sets of outcomes are possible also for programs with a finite non-deterministic behaviour [20]. Consider for example the program $\langle d, x\rangle$ in $\mathcal{L}_0$ where the program variable x is declared as

$$d(x) = (v := v + 1\ ;\ x) \sqcap v := v.$$

According to the intended meaning, if we start the above program in a state where $v = 0$ then we expect that the program either fails to terminate or delivers a state in which the variable v has an arbitrary natural number as resulting outcome.

The set $ST_H(X, Y)$ is ordered by the pointwise extension of the subset inclusion, the natural order in $\mathcal{P}(Y)$. Thus, for σ and τ in $ST_H(X, Y)$,

$$\sigma \leq \tau \text{ if and only if } \forall x \in X\colon \sigma(x) \subseteq \tau(x).$$

The set $ST_H(X, Y)$ ordered as above forms a complete partial order with least element given by the function $\lambda x \in X.\emptyset$. The least upper bound of a directed set $\{\sigma_i \mid i \in I\}$ of state transformers in $ST_H(X, Y)$ is calculated pointwise, that is,

$$(\bigvee\{\sigma_i \mid i \in I\})(x) = \bigcup\{\sigma_i(x) \mid i \in I\},$$

for all $x \in X$.

It is important to note that $ST_H(X, Y)$ is isomorphic to $\mathcal{P}(X \times Y)$, the set of all relations on X and Y. This explains why the Hoare state transformer semantics is often called *relational semantics* [160].

Every state transformer in $ST_H(X, Y)$ is a state transformer in $ST_S(X, Y)$. Hence we can define a union function and a composition function exactly in the same way as for the Smyth state transformers.

Definition 3.2.7 *Let X, Y and Z be three sets. Define, for every $x \in X$ the* union function $\Box : ST_H(X, Y) \times ST_H(X, Y) \to ST_H(X, Y)$ *by*

$$(\sigma_1 \Box \sigma_2)(x) = \sigma_1(x) \cup \sigma_2(x),$$

and the composition function $; : ST_H(X, Y) \times ST_H(Y, Z) \to ST_H(X, Z)$ *by*

$$(\sigma_1 \; ; \sigma_2)(x) = \bigcup \{\sigma_2(y) \mid y \in \sigma_1(x)\}$$

for every $x \in X$.

The above '$\Box$' and ';' are well-defined and continuous in each argument. We are now in a position to define the Hoare state transformer semantics for $\mathcal{L}_0$.

Definition 3.2.8 *The semantic function $St_H[\![\cdot]\!]$ is defined as the least function in $\mathcal{L}_0 \to ST_H(\mathtt{St}, \mathtt{St})$ such that,*

$$\begin{aligned}
St_H[\![\langle d, v := e\rangle]\!] &= St_S[\![\langle d, v := e\rangle]\!],\\
St_H[\![\langle d, b\to\rangle]\!] &= St_S[\![\langle d, b\to\rangle]\!],\\
St_H[\![\langle d, x\rangle]\!] &= St_H[\![\langle d, d(x)\rangle]\!],\\
St_H[\![\langle d, S_1 \; ; S_2\rangle]\!] &= St_H[\![\langle d, S_1\rangle]\!] \; ; St_H[\![\langle d, S_2\rangle]\!],\\
St_H[\![\langle d, S_1 \Box S_2\rangle]\!] &= St_H[\![\langle d, S_1\rangle]\!] \Box St_H[\![\langle d, S_2\rangle]\!].
\end{aligned}$$

The well-definedness of the above semantics can be proved in a similar way as for the semantics $St_S[\![\cdot]\!]$.

Egli-Milner state transformers

Finally we turn to the possibility of identifying programs on the basis of what actually happens. Computations are mapped to the subset of all their possible outcomes, including $\perp$ to denote the possibility of non-termination. Note that we differ from the Smyth state transformers because we do not neces-

sarily identify computations which fail to terminate. As always, deadlocking computations are mapped to the empty set.

Definition 3.2.9 *The set of* Egli-Milner state transformers *from a set X to a set Y is defined by*

$$ST_E(X, Y) = X \to \mathcal{P}(Y \cup \{\bot\}).$$

The set $ST_E(X, Y)$ can be turned into a cpo by the following order. For $\sigma, \tau \in ST_E(X, Y)$,

$$\begin{aligned}\sigma \leq \tau \quad \text{if and only if} \quad \forall x \in X\colon\ & (\bot \notin \sigma(x)\ \&\ \sigma(x) = \tau(x)) \text{ or} \\ & (\bot \in \sigma(x)\ \&\ \sigma(x) \setminus \{\bot\} \subseteq \tau(x)).\end{aligned}$$

This ordering has been introduced for the semantics of non-deterministic programs by Egli [60], and it has been studied in detail by De Bakker [20]. It is often referred to as the Egli-Milner ordering because Milner has defined it in an essentially equivalent formulation (as reported by Plotkin [158]). The Egli-Milner ordering is an approximation ordering: the computation represented by τ is 'better' than the one represented by σ if, for any input x, $\tau(x)$ can be obtained form $\sigma(x)$ by replacing the partialness in $\sigma(x)$ (represented by the presence of $\bot$ in $\sigma(x)$) by some set of outcomes.

Not all Egli-Milner state transformers correspond to denotations of programs that are finitely non-deterministic. We could restrict them by considering only a finite set of outcomes. However, if a computation fails to terminate then an infinite set of outcomes is also possible (essentially for the same reason as for the Hoare state transformers). Therefore, we take $ST_E^{fin}(X, Y)$ to be the set of all functions from the set X to all subsets of $Y \cup \{\bot\}$ which are either finite or contain $\bot$.

Lemma 3.2.10 *For every set X and Y, both $ST_E(X, Y)$ and $ST_E^{fin}(X, Y)$ are complete partial orders.*

Proof. If $\mathcal{V}$ is a directed set in $ST_E(X, Y)$ then

$$(3.2)\bigvee \mathcal{V} = \lambda x \in X. \begin{cases} \bigcup\{\sigma(x) \mid \sigma \in \mathcal{V}\} & \text{if } \forall \sigma \in \mathcal{V}: \bot \in \sigma(x) \\ \bigcup\{\sigma(x) \setminus \{\bot\} \mid \sigma \in \mathcal{V}\} & \text{otherwise.} \end{cases}$$

Assume now that $\sigma \in ST_E^{fin}(X, Y)$ for every $\sigma \in \mathcal{V}$, and let $x \in X$. In order to show that $\bigvee \mathcal{V}$ is the least upper bound of $\mathcal{V}$ in $ST_E^{fin}(X, Y)$ we need to prove that the set $(\bigvee \mathcal{V})(x)$ is finite whenever $\bot \notin (\bigvee \mathcal{V})(x)$.

Assume $\bot \notin (\bigvee \mathcal{V})(x)$. Then by (3.2), there exists $\sigma_0 \in \mathcal{V}$ with $\bot \notin \sigma_0(x)$. Since $\mathcal{V}$ is a directed set, for every $\sigma_1 \in \mathcal{V}$, there exists $\sigma_2 \in \mathcal{V}$ which is an upper bound of both σ_0 and σ_1. By definition of the Egli-Milner order and because $\bot \notin \sigma_0(x)$ it must be the case that $\sigma_2(x) = \sigma_0(x)$. Hence

$$\bigcup\{\sigma(x) \mid \sigma \in \mathcal{V}\} = \sigma_0(x).$$

By (3.2) and because $\sigma_0(x)$ is a finite subset of Y, $(\bigvee \mathcal{V})(x)$ is also a finite subset of Y.

Finally, the function $\lambda x \in X.\{\bot\}$ is the least element for both $ST_E(X, Y)$ and $ST_E^{fin}(X, Y)$. Hence they both are cpo's. $\Box$

As for the finitary Smyth state transformers, an alternative way to prove that $ST_E^{fin}(X, Y)$ is a complete partial order is to define it as the set of all functions from X to $\mathcal{E}(Y_\bot) \oplus (\mathbf{1})_\bot$, the Plotkin powerdomain with emptyset (added by means of a coalesced sum) of the flat cpo $Y_\bot$.

Next we give the semantical counterparts of the syntactic operators in $\mathcal{L}_0$.

Definition 3.2.11 *Let X, Y and Z be three sets. Define, for every $x \in X$, the* union function $\Box : ST_E(X, Y) \times ST_E(X, Y) \to ST_E(X, Y)$ *by*

$$(\sigma_1 \Box \sigma_2)(x) = \sigma_1(x) \cup \sigma_2(x),$$

and the composition function $; : ST_E(X, Y) \times ST_E(Y, Z) \to ST_E(X, Z)$ *by*

$$(\sigma_1 ; \sigma_2)(x) = \bigcup\{\sigma_2(y) \mid y \in \sigma_1(x) \setminus \{\bot\}\} \cup \{\bot \mid \bot \in \sigma_1(x)\}.$$

Both these functions are monotone in their arguments. Moreover, the set $ST_E^{fin}(X, Y)$ is closed under the union operation, and, if $\sigma_1 \in ST_E^{fin}(X, Y)$ and $\sigma_2 \in ST_E^{fin}(Y, Z)$ then $\sigma_1 ; \sigma_2 \in ST_E^{fin}(X, Z)$. We are now ready for the definition of the Egli-Milner state transformer semantics of $\mathcal{L}_0$.

Definition 3.2.12 *The semantic function $St_E[\![\cdot]\!]$ is defined as the least function in $\mathcal{L}_0 \to ST_E(\mathtt{St}, \mathtt{St})$ such that,*

$$\begin{aligned}
St_E[\![\langle d, v := e\rangle]\!] &= St_S[\![\langle d, v := e\rangle]\!],\\
St_E[\![\langle d, b\rightarrow\rangle]\!] &= St_S[\![\langle d, b\rightarrow\rangle]\!],\\
St_E[\![\langle d, x\rangle]\!] &= St_E[\![\langle d, d(x)\rangle]\!],\\
St_E[\![\langle d, S_1 \,;\, S_2\rangle]\!] &= St_E[\![\langle d, S_1\rangle]\!] \,;\, St_E[\![\langle d, S_2\rangle]\!],\\
St_E[\![\langle d, S_1 \,\Box\, S_2\rangle]\!] &= St_E[\![\langle d, S_1\rangle]\!] \Box St_E[\![\langle d, S_2\rangle]\!].
\end{aligned}$$

We omit the proof of the well-definedness of the above semantics since it can be obtained in a similar way as for the semantics $St_S[\![\cdot]\!]$.

Relating the three state transformer models

So far we introduced three state transformer semantics for $\mathcal{L}_0$. Next we discuss how these semantics are related.

For fixed sets X and Y, define the functions $E_H : ST_E(X, Y) \to ST_H(X, Y)$ and $E_S : ST_E(X, Y) \to ST_S(X, Y)$ respectively by

$$E_H(\sigma)(x) = \sigma(x) \setminus \{\bot\} \text{ and } E_S(\sigma)(x) = \begin{cases} Y_\bot & \text{if } \bot \in \sigma(x)\\ \sigma(x) & \text{otherwise}\end{cases}$$

for every $\sigma \in ST_E(X, Y)$ and $x \in X$. Then both E_H and E_S are strict, continuous, and onto, as can be easily verified. Moreover, if $\sigma \in ST_E^{fin}(X, Y)$ then $E_S(\sigma) \in ST_S^{fin}(X, Y)$.

Lemma 3.2.13 *For $\sigma_0, \sigma_1 \in ST_E(X, Y)$ and $\sigma_2 \in ST_E(Y, Z)$*

$$\begin{aligned}
&E_S(\sigma_0 \Box \sigma_1) = E_S(\sigma_0) \Box E_S(\sigma_1) \text{ and } E_H(\sigma_0 \Box \sigma_1) = E_H(\sigma_0) \Box E_H(\sigma_1),\\
&E_S(\sigma_0 \,;\, \sigma_1) = E_S(\sigma_0) \,;\, E_S(\sigma_1) \text{ and } E_H(\sigma_0 \,;\, \sigma_1) = E_H(\sigma_0) \,;\, E_H(\sigma_1).
\end{aligned}$$

Proof. Immediate from the definitions of E_S and E_H, and of the union and composition functions on the Egli-Milner, the Smyth and the Hoare state transformers. □

Both the semantics based on the Smyth and Hoare state transformers are projections, under E_S and E_H respectively, of the semantics based on the Egli-Milner state transformers.

Theorem 3.2.14 *For all* $\langle d, S\rangle \in \mathcal{L}_0$, $E_S(St_E[\![\langle d, S\rangle]\!]) = St_S[\![\langle d, S\rangle]\!]$ *and* $E_H(St_E[\![\langle d, S\rangle]\!]) = St_H[\![\langle d, S\rangle]\!]$.

Proof. We prove that $E_S(St_E[\![\langle d, S\rangle]\!]) = St_S[\![\langle d, S\rangle]\!]$. The other equality $E_H(St_E[\![\langle d, S\rangle]\!]) = St_H[\![\langle d, S\rangle]\!]$ can be proved in a similar way.

Let Sem_E denote the set $\mathcal{L}_0 \to ST_E(\mathtt{St}, \mathtt{St})$, and define a monotone function $\Psi_E : Sem_E \to Sem_E$ such that $St_E[\![\cdot]\!]$ is the least fixed point of Ψ_E (the definition of Ψ_E can be obtained adapting the definition of Ψ_S given in Lemma 3.2.5).

By structural induction on S, following the definition of Ψ_E, and using also Lemma 3.2.13 it is straightforward to prove that the following diagram commutes:

$$\begin{array}{ccc} Sem_E & \xrightarrow{\Psi_E} & Sem_E \\ {\scriptstyle \lambda F.E_S \circ F}\downarrow & * & \downarrow{\scriptstyle \lambda F.E_S \circ F} \\ Sem_S & \xrightarrow[\Psi_S]{} & Sem_S. \end{array}$$

Since E_S is strict and continuous and Sem_E is a cpo, we can use Proposition 2.2.5: the least fixed point of Ψ_S coincides with the projection under $\lambda F \in Sem_E.E_S \circ F$ of the least fixed point of Ψ_E, showing that

$$E_S(St_E[\![\langle d, S\rangle]\!]) = St_S[\![\langle d, S\rangle]\!],$$

for all $\langle d, S\rangle \in \mathcal{L}_0$. □

3.3 Predicate transformer models

In this section we introduce predicate transformer models for sequential programs. We will proceed as follows. First we introduce informally predicate transformers for partial and total correctness. Then we give a partial correctness semantics and a total correctness semantics to $\mathcal{L}_0$. Subsequently, we show

that for every state transformer there is an associated predicate transformer, and conversely, every predicate transformer corresponds uniquely to a state transformer. These relationships form the basic dualities we will study. The duality between the predicate transformers for total correctness and the finitary Smyth state transformers is well-known: it appears already in [194,14], and it is formally studied by Plotkin [159]. Various generalizations of this duality have been studied in [29,10,40]. The connection between predicate transformers for partial correctness and the Hoare state transformers is presented in [160].

Predicate transformers for partial and total correctness

Let X be a set. Intensionally, a *predicate* on X is a function which maps each element of X to either *true* or *false*. We will use the extensional characterization of a predicate as the set of all points of X for which, intensionally, the predicate is true. This extensional view leads us to define the set of predicates on X as $\mathcal{P}(X)$, the collection of all subsets of X. We will usually denote predicates by P and Q. Predicates are ordered by subset inclusion when not stated otherwise.

Definition 3.3.1 *A* predicate transformer *is a total function—typically denoted by π, ρ—from predicates on Y to predicates on X, that is*

$$PT(Y, X) = \mathcal{P}(Y) \to \mathcal{P}(X).$$

Predicate transformers are ordered by pointwise extension of the subset order on X, that is, for $\pi, \rho \in PT(Y, X)$,

$$\pi \leq \rho \text{ if and only if } \forall P \subseteq Y : \pi(P) \subseteq \rho(P).$$

The poset of predicate transformers $PT(Y, X)$ inherits much of the structure of $\mathcal{P}(X)$: as $PT(Y, X)$ is the pointwise extension of the complete Boolean algebra $\mathcal{P}(X)$, it will also be a complete Boolean algebra. Meets and joins are defined pointwise by

$$(\bigwedge_I \pi_i)(P) = \bigcap_I \pi_i(P) \text{ and } (\bigvee_I \pi_i)(P) = \bigcup_I \pi_i(P),$$

for every set I, predicate transformers $\pi_i \in PT(Y, X)$ $(i \in I)$, and $P \subseteq Y$. Also the complement $\neg\pi$ of a predicate transformer $\pi \in PT(Y, X)$ is defined

pointwise by

$$(\neg\pi)(P) = X \setminus \pi(P),$$

for every $P \subseteq Y$.

Predicate transformers in $PT(Y, X)$ can be used for the interpretation of a program which starts from a state in X and eventually terminates in some states that are elements of Y. We consider two different semantic models:

- *The total correctness model*: for a predicate P on Y and $\pi \in PT(Y, X)$, the predicate $\pi(P)$ holds precisely for those inputs $x \in X$ for which each computation of the program represented by π terminates in a final state $y \in Y$ satisfying the predicate P;
- *The partial correctness model*: for a predicate P on Y and $\pi \in PT(Y, X)$, the predicate $\pi(P)$ holds precisely for those inputs $x \in X$ for which each computation of the program represented by π either fails to terminate or terminates in a final state $y \in Y$ satisfying the predicate P.

In the total correctness model $\pi(Y)$ holds precisely for those inputs $x \in X$ for which each computation of the program represented by π terminates, whereas, according to the partial correctness model $\pi(Y) = X$.

Not every predicate transformer represents a 'reasonable' program. For example, a predicate transformer representing a program is required to preserve non-empty intersections: every computation of a program S at input x terminates in a final state $y \in Y$ satisfying the predicate $\bigcap_I P_i$ if and only if every computation of a program S at input x terminates in a final state $y \in Y$ satisfying P_i for all $i \in I$.

Definition 3.3.2 *Let X and Y be two sets. We define*

(i) *the domain of* total correctness predicate transformers $PT_T(Y, X)$ *to be the set of all predicate transformers in $\mathcal{P}(Y) \to \mathcal{P}(X)$ that preserve non-empty intersections;*

(ii) *the domain of* partial correctness predicate transformers $PT_P(Y, X)$ *to be the set of all total correctness predicate transformers $\pi \in PT_T(Y, X)$ such that $\pi(Y) = X$.*

Both the total and partial correctness predicate transformers are closed under arbitrary meets (defined pointwise) and functional composition. The closure under arbitrary meets turns $PT_T(Y, X)$ into a complete lattice.

We are now ready for the definition of two predicate transformer semantics for $\mathcal{L}_0$. We define them as the greatest and the least fixed point of a monotone function on the domain of all possible predicate transformer semantics for $\mathcal{L}_0$.

Lemma 3.3.3 *Let $F \in Sem_T = \mathcal{L}_0 \to PT_T(\mathtt{St}, \mathtt{St})$ and define the function $\Psi_T : Sem_T \to Sem_T$ inductively, for all $P \subseteq \mathtt{St}$, by*

$$\begin{aligned}
\Psi_T(F)(\langle d, v := e\rangle)(P) &= \{s \mid s[\mathcal{E}_\nu(e)(s)/v] \in P\},\\
\Psi_T(F)(\langle d, b\to\rangle)(P) &= \{s \mid s \in \mathcal{B}_\nu(b) \;\Rightarrow\; s \in P\},\\
\Psi_T(F)(\langle d, x\rangle)(P) &= F(\langle d, d(x)\rangle)(P),\\
\Psi_T(F)(\langle d, S_1 \,;\, S_2\rangle)(P) &= \Psi_T(F)(\langle d, S_1\rangle)(\Psi_T(F)(\langle d, S_2\rangle)(P)),\\
\Psi_T(F)(\langle d, S_1 \,\Box\, S_2\rangle)(P) &= \Psi_T(F)(\langle d, S_1\rangle)(P) \cap \Psi_T(F)(\langle d, S_2\rangle)(P).
\end{aligned}$$

Then Ψ_T is well-defined and monotone.

Proof. Both well-definedness and monotonicity are immediately proved using induction on the structure of $S \in \mathcal{L}_0$. $\Box$

As a consequence of Proposition 2.2.1, Ψ_T has both least and greatest fixed points. We denote them by $Wp_0[\![\cdot]\!]$ and $Wlp_0[\![\cdot]\!]$, respectively. The names Wp_0 and Wlp_0 stands for '*weakest precondition*' and '*weakest liberal precondition*', respectively (the subscripts indicate the language to which they are referred to).

Dijkstra's weakest precondition calculus [56] can be expressed by the semantics $Wp_0[\![\cdot]\!]$ if we allow 'enough' Boolean expressions in $BExp$. For example, the meaning of Dijkstra's guarded command $b\to S$ is the predicate transformer $Wp_0[\![\langle d, b\to \,;\, S\rangle]\!]$; the meaning of Dijkstra's conditional command

if $b_1\to S_1 \,\Box\, b_2\to S_2$ **fi**

is equivalent to $Wp_0[\![\langle d, x\rangle]\!]$ where the procedure variable x is declared by

$$d(x) = ((b_1\to \,;\, S_1) \,\Box\, (b_2\to \,;\, S_2)) \,\Box\, (b_3\to \,;\, x)$$

and $\mathcal{B}_\nu(b_3) = \mathtt{St} \setminus (\mathcal{B}_\nu(b_1) \cup \mathcal{B}_\nu(b_2))$. Finally, Dijkstra's iteration command

do $b_1\to S_1 \,\Box\, b_2\to S_2$ **od**

corresponds to $Wp_0[\![\langle d, x\rangle]\!]$ where the procedure variable x is declared by

$$d(x) = (((b_1\rightarrow\ ;\ S_1)\ ;\ x)\ \Box\ ((b_2\rightarrow\ ;\ S_2)\ ;\ x))\ \Box\ b_3\rightarrow,$$

and $\mathcal{B}\nu(b_3) = \mathtt{St} \setminus (\mathcal{B}\nu(b_1) \cap \mathcal{B}\nu(b_2))$.

Another form of conditional command '$\{b\}$' for $b \in BExp$, is often considered [95]. The computational intuition behind the command '$\{b\}$' is that it is undefined in a state in which the Boolean expression 'b' does not evaluate to true and acts as a skip otherwise. Identifying undefined with failure of termination (nothing can be guaranteed for an undefined statement), we obtain that the meaning of '$\{b\}$' is equivalent to the predicate transformer $Wp_0[\![\langle d, x\rangle]\!]$ where x is a procedure variable declared as $d(x) = b\rightarrow \Box\,(b'\rightarrow\,;x)$ and $\mathcal{B}\nu(b') = \mathtt{St} \setminus \mathcal{B}\nu(b)$.

By definition, the $Wp_0[\![\cdot]\!]$ semantics is about the total correctness of $\mathcal{L}_0$. Next we show that $Wlp_0[\![\cdot]\!]$ is concerned with the partial correctness of $\mathcal{L}_0$.

Lemma 3.3.4 *For every $\langle d, S\rangle \in \mathcal{L}_0$, $Wlp_0[\![\langle d, S\rangle]\!](\mathtt{St}) = \mathtt{St}$.*

Proof. We prove, by induction on α, that $\Psi_T^{[\alpha]}(\langle d, S\rangle)(\mathtt{St}) = \mathtt{St}$ for all ordinals α.

For $\alpha = 0$, it is straightforward to see (by structural induction on S) that $\Psi_T^{[0]}(\langle d, S\rangle)(\mathtt{St}) = \mathtt{St}$. Note that if $S \equiv x$, for $x \in PVar$, then

$$\Psi_T^{[0]}(\langle d, x\rangle)(\mathtt{St}) = F^{\top}(\langle d, d(x)\rangle)(\mathtt{St})$$

where $F^{\top}$ is the top element of Sem_T, that is, the function mapping every program $\langle d, S\rangle \in \mathcal{L}_0$ and every $P \subseteq \mathtt{St}$ to $\mathtt{St}$. Hence $F^{\top}(\langle d, d(x)\rangle)(\mathtt{St}) = \mathtt{St}$.

Next we assume for an ordinal α that for all ordinals $\beta < \alpha$,

$$\Psi_T^{[\beta]}(\langle d, S\rangle)(\mathtt{St}) = \mathtt{St},$$

and we prove that also $\Psi_T^{[\alpha]}(\langle d, S\rangle)(\mathtt{St}) = \mathtt{St}$. Recall that

$$\Psi_T^{[\alpha]}(\langle d, S\rangle)(\mathtt{St}) = \Psi_T(\bigwedge\{\Psi_T^{[\beta]} \mid \beta < \alpha\})(\langle d, S\rangle)(\mathtt{St}).$$

By structural induction on S we verify that the above right-hand side equals $\mathtt{St}$. The only interesting case is when $S \equiv x$ for $x \in PVar$:

$$\begin{aligned}
&\Psi_T(\bigwedge\{\Psi_T^{[\beta]} \mid \beta < \alpha\})(\langle d, x\rangle)(\mathtt{St}) \\
&= (\bigwedge\{\Psi_T^{[\beta]} \mid \beta < \alpha\})(\langle d, d(x)\rangle)(\mathtt{St}) \\
&= \bigcap\{\Psi_T^{[\beta]}(\langle d, d(x)\rangle)(\mathtt{St}) \mid \beta < \alpha\} \quad \text{[meets are pointwise]} \\
&= \bigcap\{\mathtt{St} \mid \beta < \alpha\} \quad \text{[induction hypothesis]} \\
&= \mathtt{St}.
\end{aligned}$$

We can conclude that $\Psi_T^{[\alpha]}(\langle d, S\rangle)(\mathtt{St}) = \mathtt{St}$ for every ordinal α. Since $Wlp_0[\![\cdot]\!]$ is defined as the greatest fixed point of Ψ_T, by Proposition 2.2.4 there exists an ordinal λ such that $Wlp_0[\![\cdot]\!] = \Psi_T^{[\lambda]}$. Therefore $Wlp_0[\![\langle d, S\rangle]\!](\mathtt{St}) = \mathtt{St}$ for every $\langle d, S\rangle \in \mathcal{L}_0$. $\square$

Intuitively, the $Wp_0[\![\cdot]\!]$ and the $Wlp_0[\![\cdot]\!]$ semantics of $\mathcal{L}_0$ agree with the informal characterization of the total and partial correctness models. To make these correspondences precise we will give duality theorems which relate the state transformer models with these predicate transformer models.

The total correctness model

Smyth state transformers capture the operational meaning of programs for the total correctness semantic model. To determine their associated predicate transformers we define the function $\omega : ST_S(X, Y) \to PT_T(Y, X)$ by

$$(3.3)\quad \omega(\sigma)(P) = \{x \in X \mid \sigma(x) \subseteq P\},$$

for $\sigma \in ST_S(X, Y)$ and $P \subseteq Y$. Well-definedness of ω is easily verified. If $\sigma(x) = Y_\perp$ then $x \notin \omega(\sigma)(P)$ for all predicates P of Y. Accordingly, if σ is the denotation of a program then $x \in \omega(\sigma)(P)$ precisely for those inputs $x \in X$ for which each computation of the program represented by σ terminates in a final state $y \in Y$ satisfying the predicate P.

We are now in a position to show that $ST_S(X, Y)$ and $PT_T(Y, X)$ are order-isomorphic, and that the two semantics $St_S[\![\cdot]\!]$ (based on the Smyth state transformers) and $Wp_0[\![\cdot]\!]$ (based on the total correctness predicate transformers) are isomorphic. To define an inverse for the function ω above we need the following lemma. It is a variation of the stability lemma in [159,10].

Lemma 3.3.5 *Let π be a predicate transformer in $PT_T(Y, X)$ and $x \in X$ with $x \in \pi(Y)$. Then there is a set $q(x, \pi)$ such that*

$$x \in \pi(P) \text{ if and only if } q(x, \pi) \subseteq P,$$

for every $P \subseteq Y$.

Proof. Define $q(x, \pi) = \bigcap\{ Q \in \mathcal{P}(Y) \mid x \in \pi(Q)\}$. If $x \in \pi(P)$ then clearly $q(x, \pi) \subseteq P$. For the converse we use the fact that total correctness predicate transformers preserve non-empty intersections. Since $x \in \pi(Y)$, the set $\{ Q \in \mathcal{P}(Y) \mid x \in \pi(Q)\}$ is non-empty. Hence

$$\pi(q(x, \pi)) = \bigcap\{\pi(Q) \mid x \in \pi(Q)\},$$

from which it follows that $x \in \pi(q(x, \pi))$. Because $q(x, \pi) \subseteq P$ and π is monotone (preserving non-empty intersections),

$$\pi(q(x, \pi)) \subseteq \pi(P).$$

Thus $x \in \pi(P)$. $\Box$

For any partial correctness predicate transformer π the above lemma shows that $q(x, \pi)$ exists and that it is uniquely determined. This set can be used to obtain a state transformer from a predicate transformer. Indeed, we can now define $\omega^{-1} : PT_T(Y, X) \to ST_S(X, Y)$ by

$$(3.4)\omega^{-1}(\pi)(x) = \begin{cases} q(x, \pi) \text{ if } x \in \pi(Y) \\ Y_\perp \qquad \text{otherwise,} \end{cases}$$

for every $\pi \in PT_T(Y, X)$ and $x \in X$.

Theorem 3.3.6 *The function $\omega : ST_S(X, Y) \to PT_T(Y, X)$ is an order isomorphism with inverse ω^{-1}.*

Proof. We first prove that both ω and ω^{-1} are monotone. Let $\sigma_1 \leq \sigma_2$ in $ST_S(X, Y)$ and let $P \subseteq Y$. If $x \in \omega(\sigma_1)(P)$ then $\sigma_1(x) \subseteq P$. But $\sigma_2(x) \subseteq$

$\sigma_1(x)$, hence also $\sigma_2(x) \subseteq P$. It follows that $x \in \omega(\sigma_2)(P)$. Hence $\omega(\sigma_1) \leq \omega(\sigma_2)$ in $PT_T(Y, X)$.

Assume now that $\pi_1 \leq \pi_2$ in $PT_T(Y, X)$ and take $x \in X$. The only interesting case is when $\omega^{-1}(\pi_1)(x) \neq Y_\perp$. In this case $x \in \pi_1(Y)$. Since $\pi_1(Y) \subseteq \pi_2(Y)$, $x \in \pi_2(Y)$. Hence $\omega^{-1}(\pi_2)(x) = q(x, \pi_2)$. But $q(x, \pi_2) \subseteq q(x, \pi_1)$ because $\pi_1 \leq \pi_2$. Thus $\omega^{-1}(\pi_2)(x) \subseteq \omega^{-1}(\pi_1)(x)$.

Next we prove that both ω and ω^{-1} are isomorphisms. For π in $PT_T(Y, X)$ and $P \subseteq Y$ we have

$$\begin{aligned}
\omega((\omega^{-1}(\pi))(P) &= \{x \in X \mid \omega^{-1}(\pi)(x) \subseteq P\} \\
&= \{x \in X \mid x \in \pi(Y) \,\&\, q(x, \pi) \subseteq P\} \\
&= \{x \in X \mid x \in \pi(Y) \,\&\, x \in \pi(P)\} \quad [\text{Lemma 3.3.5}] \\
&= \pi(P). \quad [\pi \text{ is monotone}]
\end{aligned}$$

Conversely, let σ in $ST_S(X, Y)$ and x in X. If $\sigma(x) = Y_\perp$ then $x \notin \omega(\sigma)(Y)$. Hence $\omega^{-1}(\omega(\sigma))(x) = Y_\perp = \sigma(x)$. Otherwise $\omega^{-1}(\omega(\sigma))(x) = q(x, \omega(\sigma))$. By definition of ω, $x \in \omega(\sigma)(P)$ if and only if $\sigma(x) \subseteq P$ for all $P \subseteq Y$. Hence, by Lemma 3.3.5, $q(x, \omega(\sigma)) = \sigma(x)$, from which we conclude $\omega^{-1}(\omega(\sigma))(x) = \sigma(x)$. $\Box$

Assume $\sigma \in ST_S^{fin}(X, Y)$, and let $\mathcal{V}$ be a directed set of subsets of Y. Then

$$(3.5)\sigma(x) \subseteq \bigcup \mathcal{V} \;\Rightarrow\; \exists P \in \mathcal{V} : \sigma(x) \subseteq P$$

because $\mathcal{V}$ is directed and $\sigma(x)$ is either a finite set or $Y_\perp$. Hence

$$\omega(\sigma)(\bigcup \mathcal{V}) = \bigcup \{\omega(\sigma)(P) \mid P \in \mathcal{V}\},$$

that is, $\omega(\sigma)$ is continuous. Conversely, if π is a continuous predicate transformer in $PT_T(Y, X)$ then $\omega^{-1}(\pi) \in ST_S^{fin}(X, Y)$ because the set $q(x, \pi)$ is finite. This can be proved using the property that every set is the directed union of all its finite subsets. Hence

$$\begin{aligned}
q(x, \pi) \;&=\; \bigcup \{P \subseteq q(x, \pi) \mid P \text{ finite}\} \\
&\Leftrightarrow\; x \in \pi(\bigcup \{P \subseteq q(x, \pi) \mid P \text{ finite}\}) \quad [\text{Lemma 3.3.5}] \\
&\Leftrightarrow\; x \in \bigcup \{\pi(P) \mid P \subseteq_{fin} q(x, \pi)\} \quad [\pi \text{ is continuous}]
\end{aligned}$$

$$\Leftrightarrow \exists P \subseteq_{fin} q(x,\pi)\colon q(x,\pi) \subseteq P. \quad \text{[Lemma 3.3.5]}$$

Therefore the isomorphism of Theorem 3.3.6 restricts to an isomorphism between $ST_S^{fin}(X, Y)$ and the continuous predicate transformers in $PT_T(Y, X)$.

Lemma 3.3.7 *Let $\sigma_0 \in ST_S(X, Y)$ and $\sigma_1, \sigma_2 \in ST_S(Y, Z)$. Then*

$$\begin{aligned}\omega(\sigma_1 \,\square\, \sigma_2)(P) &= \omega(\sigma_1)(P) \cap \omega(\sigma_2)(P), \text{ and}\\ \omega(\sigma_0 \,;\, \sigma_1)(P) &= \omega(\sigma_0)(\omega(\sigma_1)(P)),\end{aligned}$$

for all $P \subseteq Z$.

Proof. For $P \subseteq Z$ we have

$$\begin{aligned}\omega(\sigma_1 \,\square\, \sigma_2)(P) &= \{x \in X \mid (\sigma_1 \,\square\, \sigma_2)(x) \subseteq P\}\\ &= \{x \in X \mid \sigma_1(x) \cup \sigma_2(x) \subseteq P\}\\ &= \{x \in X \mid \sigma_1(x) \subseteq P \;\&\; \sigma_2(x) \subseteq P\}\\ &= \{x \in X \mid \sigma_1(x) \subseteq P\} \cap \{x \in X \mid \sigma_2(x) \subseteq P\}\\ &= \omega(\sigma_1)(P) \cap \omega(\sigma_2)(P);\end{aligned}$$

and also

$$\begin{aligned}\omega(\sigma_0 \,;\, \sigma_1)(P) &= \{x \in X \mid (\sigma_0 \,;\, \sigma_1)(x) \subseteq P\}\\ &= \{x \in X \mid \bigcup\{\sigma_1(y) \mid y \in \sigma_0(x)\} \subseteq P\}\\ &= \{x \in X \mid \bot \notin \sigma_0(x) \;\&\; \forall y \in \sigma_0(x)\colon \sigma_1(y) \subseteq P\}\\ &= \{x \in X \mid \sigma_0(x) \subseteq \{y \mid \sigma_1(y) \subseteq P\}\}\\ &= \{x \in X \mid \sigma_0(x) \subseteq \omega(\sigma_1)(P)\}\\ &= \omega(\sigma_0)(\omega(\sigma_1)(P)).\end{aligned}$$

□

By Theorem 3.3.6 and the above lemma it follows that if $\pi_0 \in PT_T(Y, X)$ and $\pi_1, \pi_2 \in PT_T(Z, Y)$ then

$$\begin{aligned}\omega^{-1}(\pi_1 \wedge \pi_2) &= \omega^{-1}(\pi_1) \,\square\, \omega^{-1}(\pi_2)\\ \omega^{-1}(\pi_0 \circ \pi_1) &= \omega^{-1}(\pi_0) \,;\, \omega^{-1}(\pi_1).\end{aligned}$$

Below we demonstrate the equivalence between the $Wp_0[\![\cdot]\!]$ semantics and the $St_S[\![\cdot]\!]$ semantics of $\mathcal{L}_0$.

Theorem 3.3.8 *For all $\langle d, S\rangle \in \mathcal{L}_0$ we have*

$$\omega(St_S[\![\langle d, S\rangle]\!]) = Wp_0[\![\langle d, S\rangle]\!] \text{ and } \omega^{-1}(Wp_0[\![\langle d, S\rangle]\!]) = St_S[\![\langle d, S\rangle]\!].$$

Proof. We begin by proving that $\omega(St_S[\![\cdot]\!])$ is a fixed point of Ψ_T. We proceed by structural induction on the statement S. If $S \equiv v := e$ then, for $P \subseteq \mathtt{St}$,

$$\begin{aligned}\omega(St_S[\![\langle d, v := e\rangle]\!])(P) &= \{s \in \mathtt{St} \mid St_S[\![\langle d, v := e\rangle]\!](s) \subseteq P\}\\ &= \{s \in \mathtt{St} \mid s[\mathcal{E}_\nu(e)(s)/v] \in P\}\\ &= \Psi_T(\omega(St_S[\![\cdot]\!]))(\langle d, v := e\rangle)(P).\end{aligned}$$

If $S \equiv b\rightarrow$ then, for $P \subseteq \mathtt{St}$,

$$\begin{aligned}\omega(St_S[\![\langle d, b\rightarrow\rangle]\!])(P) &= \{s \in \mathtt{St} \mid St_S[\![\langle d, b\rightarrow\rangle]\!](s) \subseteq P\}\\ &= \{s \in \mathtt{St} \mid s \in \mathcal{B}_\nu(b) \;\Rightarrow\; s \in P\}\\ &= \Psi_T(\omega(St_S[\![\cdot]\!]))(\langle d, b\rightarrow\rangle)(P).\end{aligned}$$

If $S \equiv x$ then

$$\omega(St_S[\![\langle d, x\rangle]\!]) = \omega(St_S[\![\langle d, d(x)\rangle]\!]) = \Psi_T(\omega(St_S[\![\cdot]\!]))(\langle d, x\rangle).$$

Assume now $S \equiv S_1 \,;\, S_2$. Then, for $P \subseteq \mathtt{St}$,

$$\begin{aligned}&\Psi_T(\omega(St_S[\![\cdot]\!]))(\langle d, S_1 \,;\, S_2\rangle)(P)\\ =\;& \Psi_T(\omega(St_S[\![\cdot]\!]))(\langle d, S_1\rangle)(\Psi_T(\omega(St_S[\![\cdot]\!]))(\langle d, S_2\rangle)(P))\\ =\;& \omega(St_S[\![\langle d, S_1\rangle]\!])(\omega(St_S[\![\langle d, S_2\rangle]\!])(P)) \quad \text{[induction hypothesis]}\\ =\;& \omega(St_S[\![\langle d, S_1\rangle]\!] \,;\, St_S[\![\langle d, S_2\rangle]\!])(P) \quad \text{[Lemma 3.3.7]}\\ =\;& \omega(St_S[\![\langle d, S_1 \,;\, S_2\rangle]\!])(P).\end{aligned}$$

In case $S \equiv S_1 \,\square\, S_2$ we proceed similarly. Therefore $St_S[\![\cdot]\!]$ is a fixed point of Ψ_T. Since $Wp_0[\![\cdot]\!]$ is the least fixed point of Ψ_T,

$$(3.6)\; Wp_0[\![\langle d, S\rangle]\!] \leq \omega(St[\![\langle d, S\rangle]\!]),$$

for all $\langle d, S\rangle \in \mathcal{L}_0$. Following essentially the same pattern, we can prove that $\omega^{-1}(Wp_0[\![\cdot]\!])$ is a fixed point of the semantic transformation Ψ_S defined in Lemma 3.2.5. Hence

$$(3.7) St[\![\langle d, S\rangle]\!] \leq \omega^{-1}(Wp_0[\![\langle d, S\rangle]\!]).$$

Because ω and ω^{-1} form an order isomorphism, we can conclude that the inequalities in (3.6) and (3.7) are in fact equalities. □

Since for all $\langle d, S\rangle \in \mathcal{L}_0$, $St_S[\![\langle d, S\rangle]\!]$ is in $ST_S^{fin}(\mathtt{St}, \mathtt{St})$, and the latter domain is isomorphic to the set of continuous predicate transformers in $PT_T(\mathtt{St}, \mathtt{St})$, the following corollary is immediate from Theorem 3.3.8.

Corollary 3.3.9 *For $\langle d, S\rangle \in \mathcal{L}_0$, the predicate transformer $Wp_0[\![\langle d, S\rangle]\!]$ is continuous.* □

The partial correctness model

We relate the set of Hoare state transformers to the set of partial correctness predicate transformers by restricting and co-restricting the isomorphism of Theorem 3.3.6.

The set of Hoare state transformers $ST_H(X, Y)$ is a subset of $ST_S(X, Y)$. If we apply the function ω to a Hoare state transformer $\sigma \in ST_H(X, Y)$ then

$$\omega(\sigma)(Y) = \{x \in X \mid \sigma(x) \subseteq Y\} = X.$$

Thus $\omega(\sigma)$ is a partial correctness predicate transformer in $PT_P(Y, X)$. Conversely, if π is a partial correctness predicate transformer in $PT_P(Y, X)$ then, by applying ω^{-1} to π we obtain a Hoare state transformer because $x \in \pi(Y)$ for all $x \in X$. Therefore, by Theorem 3.3.6 we have the following isomorphism.

Theorem 3.3.10 *The function $\omega : ST_H(X, Y) \to PT_P(Y, X)$ is an isomorphism with inverse ω^{-1}.* □

Note that the above isomorphism is not an order isomorphism. If $\sigma_1 \leq \sigma_2$ in $ST_H(X, Y)$ then, for all $P \subseteq Y$,

$$\omega(\sigma_1)(P) \supseteq \omega(\sigma_2)(P)$$

because $\sigma_1(x) \subseteq \sigma_2(x)$ for all $x \in X$. Similarly, for $\pi_1, \pi_2 \in PT_P(Y, X)$, if $\pi_1(P) \subseteq \pi_2(P)$ for all $P \subseteq Y$ then $\omega^{-1}(\pi_1) \geq \omega^{-1}(\pi_2)$ in $ST_H(X, Y)$.

Theorem 3.3.11 *For all $\langle d, S\rangle \in \mathcal{L}_0$ we have*

$$\omega(St_H[\![\langle d, S\rangle]\!]) = Wlp_0[\![\langle d, S\rangle]\!] \text{ and } \omega^{-1}(Wlp_0[\![\langle d, S\rangle]\!]) = St_H[\![\langle d, S\rangle]\!].$$

Proof. In a way similar to the proof of Theorem 3.3.8, we first note that $\omega(St_H[\![\langle d, S\rangle]\!])$ is a fixed point of Ψ_T. Hence

$$(3.8)\omega(St_H[\![\langle d, S\rangle]\!])(P) \subseteq Wlp_0[\![\langle d, S\rangle]\!](P),$$

for all $\langle d, S\rangle \in \mathcal{L}_0$, $P \subseteq \mathtt{St}$. Similarly, $St_H[\![\langle d, S\rangle]\!](x) \subseteq \omega^{-1}(Wlp_0[\![\langle d, S\rangle]\!])(x)$ for all $x \in X$. Since ω and ω^{-1} are monotone with respect to the opposite of the Hoare order, it follows that the above inclusions are, in fact, equalities. □

Total and partial correctness, together

Egli-Milner state transformers denote programs on the basis of what 'actually' happens. In the predicate transformer model this is done by describing both the total and the partial correctness of a program [58]. The relationship between the two domains is described informally by Nelson [154], it is briefly mentioned by De Roever [167] and De Bakker [20], and it has been proved in its full generality in [37,40].

First we need to characterize those pairs of predicate transformers in the total and partial correctness models which denote the semantics of the same computation. To this end, assume π_1 and π_2 denote the semantics of the same program in the total and partial correctness model, respectively. Intuitively it holds that, for every predicate P on the output state space Y,

$$(3.9)\pi_1(P) = \pi_1(Y) \cap \pi_2(P)$$

because, $\pi_1(P)$ holds for an input state x if and only if every computation of the program denoted by π_1 at input x terminates (and hence $x \in \pi_1(Y)$) in a final state satisfying the predicate P (and hence $x \in \pi_2(P)$).

Definition 3.3.12 *Let X and Y be two sets. The domain of* Nelson predi-

cate transformers $PT_N(Y, X)$ *consists of pairs* (π_1, π_2) *such that*

(i) $\pi_1 \in PT_T(Y, X)$,
(ii) $\pi_2 \in PT_P(Y, X)$, *and*
(iii) $\pi_1(P) = \pi_1(Y) \cap \pi_2(P)$ *for all* $P \subseteq Y$.

We show that the Nelson predicate transformers are in a bijective correspondence with the Egli-Milner state transformers. Define the trasformation $\eta : ST_E(X, Y) \to PT_N(Y, X)$ by

$$(3.10)\eta(\sigma) = \langle \omega(E_S(\sigma)), \omega(E_H(\sigma)) \rangle,$$

for all $\sigma \in ST_E(X, Y)$. Well-definedness of η is proved in the following lemma.

Lemma 3.3.13 *For every* $\sigma \in ST_E(X, Y)$, $\eta(\sigma) \in PT_N(Y, X)$.

Proof. Since $E_S(\sigma) \in ST_S(X, Y)$, by Theorem 3.3.6, $\omega(E_S(\sigma))$ is a total correctness predicate transformer in $PT_T(Y, X)$. Similarly, $\omega(E_H(\sigma))$ is a partial correctness predicate transformer in $PT_P(Y, X)$ because $E_H(\sigma)$ is an element of $ST_H(X, Y)$.

It remains to prove (3.9). For $x \in X$ and $P \subseteq Y$,

$$\begin{aligned} x \in \omega(E_S(\sigma))(P) &\Leftrightarrow E_S(\sigma)(x) \subseteq P \\ &\Leftrightarrow \sigma(x) \subseteq P \\ &\Leftrightarrow \bot \notin \sigma(x) \;\&\; \sigma(x) \setminus \{\bot\} \subseteq P \\ &\Leftrightarrow E_S(\sigma)(x) \subseteq Y \;\&\; E_H(\sigma) \subseteq P \\ &\Leftrightarrow x \in \omega(E_S(\sigma))(Y) \cap \omega(E_H(\sigma))(P). \end{aligned}$$

□

A Nelson predicate transformer $\langle \pi_1, \pi_2 \rangle \in PT_N(Y, X)$ determines uniquely an Egli-Milner state transformer $\eta^{-1}(\langle \pi_1, \pi_2 \rangle)$ by putting, for $x \in X$,

$$\eta^{-1}(\langle \pi_1, \pi_2 \rangle)(x) = \omega^{-1}(\pi_2)(x) \cup \{\bot \mid x \notin \pi_1(Y)\}.$$

According to the intuition behind the pair $\langle \pi_1, \pi_2 \rangle$, we use the predicate transformer π_1 to determine non-terminating computations, whereas we use the predicate transformer π_2 to calculate their final outcomes.

Theorem 3.3.14 *The function $\eta : ST_E(X, Y) \to PT_N(Y, X)$ is a bijection with inverse η^{-1}.*

Proof. Let $\sigma \in ST_E(X, Y)$ and $x \in X$. We have

$$\begin{array}{ll}
& \eta^{-1}(\eta(\sigma))(x) \\
= & \eta^{-1}(\langle \omega(E_S(\sigma)), \omega(E_H(\sigma))\rangle)(x) \quad [\text{definition } \eta] \\
= & \omega^{-1}(\omega(E_H(\sigma)))(x) \cup \{\bot \mid x \notin \omega(E_S(\sigma))(Y)\} \quad [\text{definition } \eta^{-1}] \\
= & E_H(\sigma)(x) \cup \{\bot \mid E_S(\sigma)(x) = Y_\bot\} \quad [\text{Theorem 3.3.10 and definition } \omega] \\
= & (\sigma(x) \setminus \{\bot\}) \cup \{\bot \mid \bot \in \sigma(x)\} \quad [\text{definition } E_H \text{ and } E_S] \\
= & \sigma(x).
\end{array}$$

Conversely, for $\langle \pi_1, \pi_2 \rangle \in PT_N(Y, X)$, $P \subseteq Y$, and $x \in X$,

$$\begin{array}{ll}
& x \in \omega(E_S(\eta^{-1}(\langle \pi_1, \pi_2 \rangle)))(P) \\
\Leftrightarrow & E_S(\eta^{-1}(\langle \pi_1, \pi_2 \rangle))(x) \subseteq P \quad [\text{definition } \omega] \\
\Leftrightarrow & \bot \notin \eta^{-1}(\langle \pi_1, \pi_2 \rangle))(x) \;\&\; \eta^{-1}(\langle \pi_1, \pi_2 \rangle)(x) \subseteq P \quad [\text{definition } E_S] \\
\Leftrightarrow & x \in \pi_1(Y) \;\&\; \omega^{-1}(\pi_2)(x) \subseteq P \quad [\text{definition } \eta^{-1}] \\
\Leftrightarrow & x \in \pi_1(Y) \;\&\; x \in \pi_2(P) \quad [\text{Lemma 3.3.5}] \\
\Leftrightarrow & x \in \pi_1(P). \quad [\text{Equation (3.9)}]
\end{array}$$

Hence

$$\begin{array}{ll}
& \eta(\eta^{-1}(\langle \pi_1, \pi_2 \rangle)) \\
= & \langle \omega(E_S(\eta^{-1}(\langle \pi_1, \pi_2 \rangle))), \omega(E_H(\eta^{-1}(\langle \pi_1, \pi_2 \rangle)))\rangle \quad [\text{definition } \eta] \\
= & \langle \pi_1, \omega(\eta^{-1}(\langle \pi_1, \pi_2 \rangle) \setminus \{\bot\})\rangle \quad [\text{above calculation and definition } E_H] \\
= & \langle \pi_1, \omega(\omega^{-1}(\pi_2))\rangle \quad [\text{definition } \eta^{-1}] \\
= & \langle \pi_1, \pi_2 \rangle. \quad [\text{Theorem 3.3.10}] \quad \Box
\end{array}$$

The set of Nelson predicate transformers $PT_N(Y, X)$ can now be turned into a partial order by the order induced by η^{-1} on $PT_N(Y, X)$: for $\langle \pi_1, \pi_2 \rangle$ and $\langle \pi_3, \pi_4 \rangle$ in $PT_N(Y, X)$, define

$$\langle \pi_1, \pi_2 \rangle \leq \langle \pi_3, \pi_4 \rangle \text{ if and only if } \eta^{-1}(\langle \pi_1, \pi_2 \rangle) \leq \eta^{-1}(\langle \pi_3, \pi_4 \rangle).$$

The order on $PT_N(Y, X)$ satisfies the following equation.

Lemma 3.3.15 *For all* $\langle \pi_1, \pi_2 \rangle$ *and* $\langle \pi_3, \pi_4 \rangle$ *in* $PT_N(Y, X)$,

$$\langle \pi_1, \pi_2 \rangle \leq \langle \pi_3, \pi_4 \rangle \;\Leftrightarrow\; \forall P \subseteq Y : \pi_1(P) \subseteq \pi_3(P) \;\&\; \pi_2(P) \supseteq \pi_4(P).$$

Proof. Let us use σ as shorthand for $\eta^{-1}(\langle \pi_1, \pi_2 \rangle)$ and τ as shorthand for $\eta^{-1}(\langle \pi_3, \pi_4 \rangle)$. Assume first $\sigma \leq \tau$ in $ST_E(X, Y)$ and let $P \subseteq Y$.

If $x \in \pi_1(P)$ then $\bot \notin \sigma(x)$. Since $\sigma \leq \tau$, $\sigma(x) = \tau(x)$. Because $x \in \pi_1(P) = \omega(E_S(\sigma))(P)$ it follows that $x \in \pi_3(P) = \omega(E_S(\tau))(P)$. Thus $\pi_1(P) \subseteq \pi_3(P)$.

If $x \in \pi_4(P)$ we have to consider two cases depending on the presence of $\bot$ in $\sigma(x)$. In case $\bot \notin \sigma(x)$, $\sigma \leq \tau$ implies $\sigma(x) = \tau(x)$. Hence $x \in \pi_4(P) = \omega(E_H(\tau))(P)$ implies $x \in \omega(E_H(\sigma))(P) = \pi_2(P)$. In the other case $\bot \in \sigma(x)$. Since $\sigma \leq \tau$ then $\sigma(x) \setminus \{\bot\} \subseteq \tau(x)$. Thus $\sigma(x) \setminus \{\bot\} \subseteq \tau(x) \setminus \{\bot\}$, that is, $E_H(\sigma)(x) \subseteq E_H(\tau)(x)$. Hence $x \in \pi_4(P) = \omega(E_H(\tau))(P)$ implies that x is an element of $\omega(E_H(\sigma))(P) = \pi_2(P)$. Therefore $\pi_2(P) \supseteq \pi_4(P)$.

For the converse, assume that $\pi_1(P) \subseteq \pi_3(P)$ and $\pi_2(P) \supseteq \pi_4(P)$ for all $P \subseteq Y$. First note that for every $x \in X$,

$$\omega^{-1}(\pi_2)(x) \subseteq \omega^{-1}(\pi_4)(x) \tag{3.11}$$

because $\pi_4(P) \subseteq \pi_2(P)$ for all $P \subseteq Y$. Next we distinguish two cases.

If $\bot \notin \sigma(x)$ then by definition of η^{-1} $x \in \pi_1(Y)$ and $\sigma(x) = \omega^{-1}(\pi_2)(x)$. Since $\pi_1(Y) \subseteq \pi_3(Y)$, $x \in \pi_3(Y)$. Thus $\bot \notin \tau(x)$ and $\tau(x) = \omega^{-1}(\pi_4)(x)$. By (3.11) it follows $\sigma(x) \subseteq \tau(x)$. We still need to prove the reverse inclusion. Because $\langle \pi_1, \pi_2 \rangle$ is a Nelson predicate transformer, $x \in \pi_1(Y)$ and, by Lemma 3.3.5, x is an element of $\pi_2(\omega^{-1}(\pi_2)(x))$, it follows that $x \in \pi_1(\omega^{-1}(\pi_2)(x))$. Hence x is in $\pi_3(\omega^{-1}(\pi_2)(x))$. Because $\langle \pi_3, \pi_4 \rangle$ is a Nelson predicate transformer too, x is in $\pi_4(\omega^{-1}(\pi_2)(x))$. Thus, by Lemma 3.3.5, $\omega^{-1}(\pi_4)(x) = q(x, \pi_4) \subseteq \omega^{-1}(\pi_2)(x)$. Therefore $\tau(x) \subseteq \sigma(x)$.

If $\bot \in \sigma(x)$ then $\sigma(x) \setminus \{\bot\} = \omega^{-1}(\pi_2)(x)$ by definition of η^{-1}. Thus, by equation (3.11), $\sigma(x) \setminus \{\bot\} \subseteq \omega^{-1}(\pi_4)(x)$. Since $\omega^{-1}(\pi_4)(x) \subseteq \tau(x)$ by definition of η^{-1}, we obtain that $\sigma(x) \setminus \{\bot\} \subseteq \tau(x)$. $\square$

The above characterization of the order between Nelson predicate transformers is used in [167] to give an early treatment of recursion in the original weakest

precondition calculus of Dijkstra [56], based on continuity of the weakest preconditions. A more detailed treatment of the recursion is given in [91] and [20].

We conclude this section by showing that the Egli-Milner state transformer semantics of $\mathcal{L}_0$ corresponds to the pair of weakest precondition and weakest liberal precondition semantics. For $\langle d, S\rangle \in \mathcal{L}_0$ we have

$$\begin{array}{ll} & \eta(St_E[\![\langle d, S\rangle]\!]) \\ = & \langle \omega(E_S(St_E[\![\langle d, S\rangle]\!])), \omega(E_H(St_E[\![\langle d, S\rangle]\!]))\rangle \\ = & \langle \omega(St_S[\![\langle d, S\rangle]\!])), \omega(St_H[\![\langle d, S\rangle]\!]))\rangle \quad \text{[Theorem 3.2.14]} \\ = & \langle Wp_0[\![\langle d, S\rangle]\!], Wlp_0[\![\langle d, S\rangle]\!]\rangle. \quad \text{[Theorems 3.3.8 and 3.3.11]} \end{array}$$

As a consequence of the above, we obtain that the weakest precondition semantics $Wp_0[\![\langle d, S\rangle]\!]$ and the weakest liberal precondition semantics $Wlp_0[\![\langle d, S\rangle]\!]$ of a program $\langle d, S\rangle \in \mathcal{L}_0$ satisfy the pairing condition (3.9).

3.4 Can a backtrack operator be added to $\mathcal{L}_0$?

In this section we study the incorporation of a backtrack operator into our language $\mathcal{L}_0$. The backtrack operator is a binary operator '$\boxtimes$' which backtracks to the second component if the first component deadlocks. We define it in the domain of Egli-Milner state transformers to derive its weakest precondition semantics. Maybe surprisingly, the backtrack operator is not monotone with respect to the order of the total correctness predicate transformers. To repair the problem a new order can be defined which refines the ordinary order on predicate transformers and such that the backtrack operator becomes monotone. However, sequential composition is not monotone with respect to this new order. In order to justify the well-definedness of a weakest precondition semantics for $\mathcal{L}_0$ extended with a backtrack operator we prove that under certain conditions the least fixed point of a non-monotone function exists.

Our extension of $\mathcal{L}_0$ is a variation of the language studied in [154]. In this article a weakest precondition semantics together with a weakest liberal precondition semantics for a language with a backtrack operator is given. Below we will concentrate only on a weakest precondition semantics.

Definition 3.4.1 (i) *The set* $(S \in)\, Stat_B$ *of* statements *is given by*

$$S ::= v := e \mid b\!\rightarrow \mid x \mid S\,;S \mid S \,\square\, S \mid S \boxtimes S\,.$$

(ii) *The set* $(d \in)\, Decl_B$ *of* declarations *is defined by* $PVar \rightarrow Stat_B$.

(iii) *The language* $\mathcal{L}_B$ *is given by* $Decl_B \times Stat_B$.

To guide the intuition about the backtrack operator '$\boxtimes$' we define the corresponding semantical operator in the domain of the Egli-Milner state transformers. For $\sigma_1, \sigma_2 \in ST_E(X, Y)$ define $\sigma_1 \boxtimes \sigma_2$ by

$$(\sigma_1 \boxtimes \sigma_2)(x) = \begin{cases} \sigma_2(x) \text{ if } \sigma_1(x) = \emptyset \\ \sigma_1(x) \text{ otherwise,} \end{cases}$$

for $x \in X$. A similar definition can be given for the Smyth state transformers and for the Hoare state transformers. It is a straightforward verification to see that

$$\boxtimes : ST_E(X, Y) \times ST_E(X, Y) \to ST_E(X, Y)$$

is a monotone function. However this is not true with respect to the order of the Smyth state transformers $ST_S(X, Y)$. Indeed if $y_1, y_2 \in Y$ then

$$\lambda x.\{y_1\} \leq \lambda x.\emptyset$$

in $ST_S(X, Y)$, but,

$$\begin{aligned} \lambda x.\{y_1\} \boxtimes \lambda x.\{y_2\} &= \lambda x.\{y_1\} \\ &\not\leq \lambda x.\{y_2\} \\ &= \lambda x.\emptyset \boxtimes \lambda x.\{y_2\} \end{aligned}$$

The above monotonicity problem is caused by the fact that the function $\lambda x.\emptyset$ is the top element of $ST_S(X, Y)$. In $ST_E(X, Y)$ this is not the case, and indeed the backtrack operator is monotone. We can try to define a new domain of state transformers between $ST_S(X, Y)$ and $ST_E(X, Y)$ by introducing a new order on the Smyth state transformers which preserves deadlock. The idea is that a state transformer which does not deadlock cannot be substituted by another which does, even if more can be guaranteed for it.

Definition 3.4.2 *Define* $ST_D(X, Y)$ *to be the set of all functions from* X *to* $\mathcal{P}(Y) \cup \{Y_\perp\}$ *ordered as follows. For* $\sigma, \tau \in ST_D(X, Y)$,

$$\begin{aligned} \sigma \leq \tau \text{ if and only if } \forall x \in X : (\tau(x) \neq \emptyset \;\&\; \sigma(x) \supseteq \tau(x)) \text{ or} \\ (\tau(x) = \emptyset \;\&\; (\sigma(x) = \emptyset \text{ or } \sigma(x) = Y_\perp)). \end{aligned}$$

As for $ST_S(X, Y)$, the above domain $ST_D(X, Y)$ is a partial order with the function $\lambda x.\{Y_\perp\}$ as bottom element. However $ST_D(X, Y)$ need not to be a cpo. For example let $\mathbb{N}$ be the set of natural numbers, and consider in $ST_D(X, \mathbb{N})$ the following directed set

$$\lambda x.\mathbb{N} \leq \lambda x.\mathbb{N} \setminus \{0\} \leq \lambda x.\mathbb{N} \setminus \{0, 1\} \leq \ldots .$$

It has no upper bound in $ST_D(X, \mathbb{N})$ (in $ST_S(X, \mathbb{N})$ it would have the function $\lambda x.\emptyset$ as a least upper bound).

It is now easy to see that the backtrack operator '$\boxtimes$' is monotonic with respect to the new domain $ST_D(X, Y)$. However the composition function ';', defined exactly as for $ST_S(X, Y)$, is not monotone anymore. For $y_1, y_2 \in Y$,

$$\lambda x.\{y_1, y_2\} \leq \lambda x.\{y_1\}$$

in $ST_D(X, Y)$. If we compose them with the function $\sigma \in ST_S(Y, Z)$ which maps y_2 to $\{z\} \subseteq Z$ and every other $y \in Y$ to $\emptyset$ we obtain

$$\begin{aligned} \lambda x.\{y_1, y_2\} \,;\, \sigma &= \lambda x.\{z\} \\ &\not\leq \lambda x.\emptyset \\ &= \lambda x.\{y_1\} \,;\, \sigma. \end{aligned}$$

Next we turn to a weakest precondition semantics for $\mathcal{L}_B$. First we use the isomorphism of Theorem 3.3.6 to derive the semantical backtrack operator in the domain of total correctness predicate transformers. For $\sigma_1, \sigma_2 \in ST_S(X, Y)$ let

$$\pi_1 = \omega(\sigma_1) \text{ and } \pi_2 = \omega(\sigma_2).$$

Then $\sigma_1 = \omega^{-1}(\pi_1)$ and $\sigma_2 = \omega^{-1}(\pi_2)$. For $P \subseteq Y$,

$$\begin{aligned} &\omega(\sigma_1 \boxtimes \sigma_2)(P) \\ = \;& \{x \in X \mid (\sigma_1 \boxtimes \sigma_2)(x) \subseteq P\} \\ = \;& \{x \in X \mid \sigma_1(x) = \emptyset \;\&\; \sigma_2(x) \subseteq P\} \cup \\ & \{x \in X \mid \sigma_1(x) \neq \emptyset \;\&\; \sigma_1(x) \subseteq P\} \\ = \;& (\{x \in X \mid \sigma_1(x) \subseteq \emptyset\} \cap \{x \in X \mid \sigma_2(x) \subseteq P\}) \cup \\ & (X \setminus \{x \in X \mid \sigma_1(x) \subseteq \emptyset\} \cap \{x \in X \mid \sigma_1(x) \subseteq P\}) \end{aligned}$$

$$
\begin{array}{ll}
= & (\omega(E_S(\sigma_1))(\emptyset) \cap \omega(E_S(\sigma_2))(P)) \cup \\
& ((X \setminus \omega(E_S(\sigma_1))(\emptyset)) \cap \omega(E_S(\sigma_1))(P)) \\
= & (\pi_1(\emptyset) \cap \pi_2(P)) \cup ((X \setminus \pi_1(\emptyset)) \cap \pi_1(P)) \\
= & \pi_1(P) \cap (\pi_1(\emptyset) \;\Rightarrow\; \pi_2(P)),
\end{array}
$$

where $P \;\Rightarrow\; Q$ is a shorthand for $(P \cap Q) \cup (X \setminus P)$. The above justifies the following definition.

Definition 3.4.3 *For $\pi_1, \pi_2 \in PT_T(Y, X)$ define $\pi_1 \boxtimes \pi_2 \in PT_T(Y, X)$ by*

$$(\pi_1 \boxtimes \pi_2)(P) = \pi_1(P) \cap (\pi_1(\emptyset) \;\Rightarrow\; \pi_2(P)),$$

for all $P \subseteq Y$.

Since ω is an order-preserving isomorphism '$\boxtimes$' is not monotone with respect to the order in $PT_T(Y, X)$. Nevertheless we want to define the weakest precondition semantics of $\mathcal{L}_B$ in the same way as we did in Lemma 3.3.3 for the weakest precondition semantics of $\mathcal{L}_0$: as the least fixed point of a higher order transformation.

Definition 3.4.4 *Let $F \in Sem_B = \mathcal{L}_B \to PT_T(\mathtt{St}, \mathtt{St})$ and define the function $\Psi_B : Sem_B \to Sem_B$ inductively by*

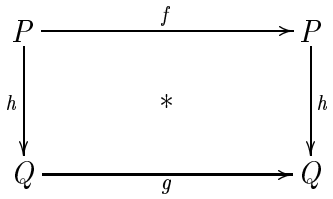
$$
\begin{array}{lll}
\Psi_B(F)(\langle d, v := e\rangle) & = & Wp_0[\![\langle d, v := e\rangle]\!], \\
\Psi_B(F)(\langle d, b{\to}\rangle) & = & Wp_0[\![\langle d, b{\to}\rangle]\!], \\
\Psi_B(F)(\langle d, x\rangle) & = & F(\langle d, d(x)\rangle), \\
\Psi_B(F)(\langle d, S_1 \,;\, S_2\rangle) & = & \Psi_B(F)(\langle d, S_1\rangle) \circ \Psi_B(F)(\langle d, S_2\rangle), \\
\Psi_B(F)(\langle d, S_1 \,\Box\, S_2\rangle) & = & \Psi_B(F)(\langle d, S_1\rangle) \wedge \Psi_B(F)(\langle d, S_2\rangle), \\
\Psi_B(F)(\langle d, S_1 \boxtimes S_2\rangle)(P) & = & \Psi_B(F)(\langle d, S_1\rangle) \boxtimes \Psi_B(F)(\langle d, S_2\rangle).
\end{array}
$$

Well-definedness of Ψ_B is straightforwardly checked, since it is based on the well-definedness of the corresponding semantical operators in $PT_B(\mathtt{St}, \mathtt{St})$. Since the semantical operator '$\boxtimes$' is not monotone, also Ψ_B is not monotone. At first sight it seems that we cannot define a weakest precondition semantics for $\mathcal{L}_B$ as the least fixed point of Ψ_B because the ordinary fix-point methods require Ψ_B to be at least monotone.

However, we show that, under certain conditions, the least fixed point of a

non-monotonic function on a poset (which need not to be complete) exists and that it can be calculated by iteration.

Proposition 3.4.5 *Let P be a cpo and let Q be a poset such that there there is an onto and continuous function $h : P \to Q$. Assume also that, for every $y \in Q$ there is a top element in $h^{-1}(y)$, that is, there exists $z \in h^{-1}(y)$ such that $x \leq z$ for all $x \in h^{-1}(y)$. If $f : P \to P$ is a monotone function then every function $g : Q \to Q$ making the following diagram commute*



has a least fixed point. Moreover, for every ordinal α, $g^{\langle\alpha\rangle}$ exists and equals $h(f^{\langle\alpha\rangle})$.

Proof. By Proposition 2.2.3 f has as least fixed point $f^{\langle\lambda\rangle}$, for some ordinal λ. We have:

$$h(f^{\langle\lambda\rangle}) = h(f^{\langle\lambda+1\rangle}) = h(f(f^{\langle\lambda\rangle})) = g(h(f^{\langle\lambda\rangle})).$$

So $h(f^{\langle\lambda\rangle})$ is a fixed point of g. Next we prove $h(f^{\langle\lambda\rangle})$ is also the least one.

Let $y \in Q$ be such that $g(y) = y$ and let z be the top element in $h^{-1}(y)$. We prove by induction on ordinals that $f^{\langle\alpha\rangle} \leq z$ for every ordinal α. In the proof below we need the fact that $f(z) \leq z$ which can justified by the following

$$h(f(z)) = g(h(z)) = g(y) = y.$$

If $\alpha = 0$ then $f^{\langle\alpha\rangle} = f(\bot) \leq f(z) \leq z$. Assume now that $f^{\langle\beta\rangle} \leq z$ for all ordinals $\beta < \alpha$. We have

$$\begin{aligned}
(\forall \beta < \alpha : f^{\langle\beta\rangle} \leq z) &\Rightarrow \bigvee\{f^{\langle\beta\rangle} \mid \beta < \alpha\} \leq z \\
&\Rightarrow f(\bigvee\{f^{\langle\beta\rangle} \mid \beta < \alpha\}) \leq f(z) \quad [f \text{ is monotone}] \\
&\Rightarrow f^{\langle\alpha\rangle} \leq z. \quad [\text{definition of } f^{\langle\alpha\rangle} \text{ and } f(z) \leq z]
\end{aligned}$$

It follows that $f^{\langle\lambda\rangle} \leq z$. Hence, by monotonicity of h,

$$h(f^{\langle\lambda\rangle}) \leq h(z) = y,$$

from which we can conclude that $h(f^{\langle\lambda\rangle})$ is the least fixed point of g.

It remains to prove that $g^{\langle\alpha\rangle} = h(f^{\langle\alpha\rangle})$ for every ordinal α. Since h is onto and monotone, it is also strict. Hence, for $\alpha = 0$,

$$h(f^{\langle\alpha\rangle}) = h(f(\bot)) = g(h(\bot)) = g(\bot) = g^{\langle\alpha\rangle}.$$

Using induction on ordinals we have for $\alpha > 0$

$$\begin{aligned} h(f^{\langle\alpha\rangle}) &= h(f(\bigvee\{f^{\langle\beta\rangle} \mid \beta < \alpha\})) \\ &= g(h(\bigvee\{f^{\langle\beta\rangle} \mid \beta < \alpha\})) \quad [\text{commutativity}] \\ &= g(\bigvee\{h(f^{\langle\beta\rangle}) \mid \beta < \alpha\}) \quad [h \text{ is continuous}] \\ &= g(\bigvee\{g^{\langle\beta\rangle} \mid \beta < \alpha\}) \quad [\text{induction hypothesis}] \\ &= g^{\langle\alpha\rangle}. \quad [\text{by definition}] \end{aligned}$$

□

In order to apply the above proposition consider the complete partial order $Sem_E = \mathcal{L}_0 \to ST_E(\mathtt{St}, \mathtt{St})$, and define the transformation $\Phi : Sem_E \to Sem_B$ by

$$\Phi(F)(\langle d, S\rangle) = \omega(E_S(F(\langle d, S\rangle))).$$

Since $E_S : ST_E(\mathtt{St}, \mathtt{St}) \to ST_B(\mathtt{St}, \mathtt{St})$ is strict, onto and continuous, and $\omega : ST_B(\mathtt{St}, \mathtt{St}) \to PT_T(\mathtt{St}, \mathtt{St})$ is an order isomorphism, Φ is onto and continuous. Moreover, if $\sigma \in ST_B(\mathtt{St}, \mathtt{St})$ then σ is also a function in $ST_E(\mathtt{St}, \mathtt{St})$ and $E_S(\sigma) = \sigma$. Clearly σ is the top element of $E_S^{-1}(\sigma)$. Hence also $\Phi^{-1}(F)$ has a top element for every $F \in Sem_B$.

Theorem 3.4.6 *The function $\Psi_B : Sem_B \to Sem_B$ has a least fixed point which can be calculated by iteration from the bottom element of Sem_B.*

Proof. Define $\Psi_E : Sem_E \to Sem_E$ inductively by

$$\begin{aligned}
\Psi_E(F)(\langle d, v := e\rangle) &= St_E[\![\langle d, v := e\rangle]\!],\\
\Psi_E(F)(\langle d, b\rightarrow\rangle) &= St_E[\![\langle d, b\rightarrow\rangle]\!],\\
\Psi_E(F)(\langle d, x\rangle) &= F(\langle d, d(x)\rangle),\\
\Psi_E(F)(\langle d, S_1 \,;\, S_2\rangle) &= \Psi_E(F)(\langle d, S_1\rangle) \,;\, \Psi_E(F)(\langle d, S_2\rangle),\\
\Psi_E(F)(\langle d, S_1 \,\square\, S_2\rangle) &= \Psi_E(F)(\langle d, S_1\rangle) \,\square\, \Psi_E(F)(\langle d, S_2\rangle),\\
\Psi_E(F)(\langle d, S_1 \boxtimes S_2\rangle)(P) &= \Psi_E(F)(\langle d, S_1\rangle) \boxtimes \Psi_E(F)(\langle d, S_2\rangle).
\end{aligned}$$

Well-definedness and monotonicity of Ψ_E can be straightforwardly checked. It is ultimately based on the monotonicity of the corresponding state transformer constructors. Moreover, by induction on the structure of S, and using Theorem 3.2.14, Theorem 3.3.8, and the definition of '$\boxtimes$' we have that

$$\Phi(\Psi_E(F))(\langle d, S\rangle) = \Psi_B(\Phi(F))(\langle d, S\rangle)$$

for all $\langle d, S\rangle \in \mathcal{L}_B$. Therefore by Proposition 3.4.5 Ψ_B has a least fixed point which can be calculated by iteration from the bottom element of Sem_B. $\square$

The least fixed point of Ψ_B defines the weakest precondition semantics for $\mathcal{L}_B$.

3.5 Concluding notes

The predicate transformer semantics we presented in this chapter is formulated using higher-order transformations. Hence predicate transformers are regarded as basic objects in contrast to the more traditional view which regards predicates on states as basic objects. Accordingly, we treated recursion at the level of predicate transformers whereas for example Dijkstra and Scholten [58] treat recursion at the level of predicates.

Several semantic domains we introduced in this chapter are general enough to support both recursion and unbounded non-determinism. For example our Egli-Milner state transformer domain $ST_E(X, Y)$ is more general than the similar domain for countable non-determinism of Apt and Plotkin [10], while our predicate transformers domain $PT_T(Y, X)$ is equivalent to the domain of predicate transformers for unbounded non-determinism treated in [57,96].

We have not used the capability of the domains to express unbounded non-determinism. In this chapter we only treated a language without specification constructs. An extension of the language $\mathcal{L}_0$ with this kind of constructs is treated in Chapter 4.

The results of this chapter can be extended to capture the semantics of more general programs than the sequential ones. In Chapter 7 we treat an example of a program which interacts with its environment by extending $\mathcal{L}_0$ with a parallel operator. The key step towards this goal is a refinement of our definition of predicates. In Chapter 5 affirmative predicates are introduced as open sets of a topological space, and in Chapter 6 we introduce two kinds of topological predicate transformers which generalize the total and the partial correctness predicate transformers. Dualities between state transformers and topological predicate transformers are also studied in Chapter 6.