Joining Specification Statements

K. Rustan M. Leino
Digital's Systems Research Center
130 Lytton Avenue
Palo Alto, CA 94301. U.S.A.
rustan@pa.dec.com

Rajit Manohar*
Department of Computer Science
California Institute of Technology
Pasadena, CA 91125. U.S.A.
rajit@cs.caltech.edu

March 2, 1998

Abstract: *The specification statement allows us to easily express what a program statement does. This paper shows how refinement of specification statements can be directly expressed using the predicate calculus. It also shows that the specification statements interpreted as predicate transformers form a complete lattice, and that this lattice is the lattice of conjunctive predicate transformers. The join operator of this lattice is constructed as a specification statement. The join operators of two interesting sublattices of the set of specification statements are also investigated.*
Keywords: *Specification statement; Conjunctive predicate transformers; Program semantics; Complete lattices*

1. Introduction

We are motivated by programs operating on a state space whose finite collection of coordinates are called variables. A predicate is a function from the state of a program to a boolean. A predicate transformer is a function from one predicate to another. The behavior of a program statement is captured by its *weakest precondition* semantics. The weakest precondition semantics of a statement is a predicate transformer that maps a predicate on the final state of the statement to a predicate on the initial state. We identify a program statement with its weakest precondition semantics. Given a predicate Q and a program statement S, $S.Q$ is the weakest precondition that guarantees that S will terminate in a state in which Q holds.

A powerful and convenient program statement is the specification statement introduced by Carroll Morgan [14]. It has three components, and is written as

$$w{:}[\mathit{Pre}, \mathit{Post}]$$

where w (the *frame*) is a list of variables, and Pre (the *precondition*) and Post (the *postcondition*) are predicates. If started in a state satisfying Pre, the statement terminates in a state satisfying Post, modifying only those variables listed in the frame w.

In this paper we formulate refinement in the lattice of specification statements using the predicate calculus. We show that the lattice of specification statements and the lattice of conjunctive predicate transformers are the same, and construct the join in the lattice. We show that the join of a set of specification statements S is the least refined conjunctive predicate transformer that refines the angelic choice of the statements from S. We investigate properties of the lattice of universally conjunctive predicate transformers, and the relation between its join and angelic choice. Finally, we discuss the meaning of the join operations in the various lattices, and suggest applications of the join to program development.

Our proof format is from [7]. We use $P[v := e]$ to denote the predicate that holds for the expression e just when the predicate P holds for the variable v. The expression

$$\langle Q\ x \mid R \triangleright T\rangle$$

where x is an unordered list of identifier names (*dummies*), R (the *range*) is a predicate, and T (the *term*) is an expression of the appropriate type, is used to denote quantification. When R is true everywhere or understood, we omit the range and write the quantification as:

$$\langle Q\ x \triangleright T\rangle$$

A common quantifier is the set constructor, written as

$$\{x \mid R \triangleright T\}$$

*The research described in this report was sponsored in part by the Defence and Advanced Research Projects Agency and monitored by the Office of Army Research.

which denotes the set containing all the terms T where x satisfies R. We use [ ] for *everywhere* brackets, which denote universal quantification over all variables.

A predicate transformer is *positively conjunctive* if it distributes over any nonempty, possibly infinite conjunction. It is called *universally conjunctive* if it also distributes over the empty conjunction. We will use *conjunctive* to mean positively conjunctive.

We assume that the reader has a rudimentary knowledge of lattice theory. Birkhoff [6] contains a thorough treatment of the subject, and Van de Snepscheut [16] provides a self-contained introduction to the aspects of lattice theory extensively used in program semantics. The join operation in a lattice is denoted by $\uparrow$, and the meet is denoted by $\downarrow$. We write $\uparrow Z$, where Z is a set, to mean the join of all the elements in Z, and similar for $\downarrow Z$.

The definition of the simple specification statement, for any predicate Q, is given by

$$(w{:}[Pre, Post]).Q \;=\; Pre \wedge \langle \forall\, w \mid Post \triangleright Q \rangle \tag{1}$$

Notationally, it is often convenient to be able to express the postcondition in terms of the initial values of variables. We therefore permit *Post* to be a predicate on both initial and final states. We adopt the convention that a variable subscripted with 0 in the postcondition refers to the initial value of the variable. *Pre* is a predicate on initial states only. Therefore, we assume that *Pre* is independent of initial-value variables. Since the only variables that are modified are those from the frame, the final value of all variables not in w is the same as their initial value. Consequently, we assume that all initial-value variables on which *Post* depends are from w_0. Like Morgan [14], we have

$$(w{:}[Pre, Post]).Q \;=\; Pre \wedge \langle \forall\, w \mid Post \triangleright Q \rangle [w_0 := w] \tag{2}$$

We use $\mathcal{S}_0$ to denote the set of specification statements with a fixed frame w, and $\mathcal{S}$ to denote specification statements from $\mathcal{S}_0$ whose postconditions are independent of initial-value variables.

Demonic choice $\Box$ is defined as follows: Given any predicate Q,

$$\langle \Box\, i \triangleright S_i \rangle.Q \;=\; \langle \forall\, i \triangleright S_i.Q \rangle \tag{3}$$

The following lemma gives a simple expression for the specification statement that corresponds to the demonic choice of a set of specification statements.

Lemma.
For specification statements from $\mathcal{S}_0$ (or from $\mathcal{S}$),

$$\langle \Box\, i \triangleright w{:}[P_i, Q_i] \rangle \;=\; w{:}[\langle \forall\, i \triangleright P_i \rangle, \langle \exists\, i \triangleright Q_i \rangle] \tag{4}$$

Proof: Follows from (2), (3). □

The refinement relation $\sqsubseteq$ on statements (cf. [3]) is defined as follows:

$$S \sqsubseteq T \;\equiv\; \langle \forall\, R \triangleright [S.R \Rightarrow T.R] \rangle \tag{5}$$

2. Lattice of Specifications

In this section we show that $\mathcal{S}$ is a complete lattice. We use the following lemma from lattice theory, which can be found in [6].

Lemma. (*lattices*) (6)

Let $\langle X, \leq, \downarrow \rangle$ be a complete semi-lattice with a maximum element. For any $Z \subseteq X$, let $\uparrow Z$ be defined to be $\downarrow \{x \mid \langle \forall\, z \mid z \in Z \triangleright z \leq x \rangle \triangleright x\}$. Then $\langle X, \leq, \downarrow, \uparrow \rangle$ is a complete lattice.

The join operation constructed by this lemma is the only join operation that is consistent with the $\leq$ relation and $\downarrow$.

Theorem. (7)

The $\mathcal{S}$-specification statements form a complete lattice.

Proof: The structure $\langle \mathcal{S}, \Box, \sqsubseteq \rangle$ forms a complete semi-lattice since (4) ensures that the meet of every nonempty subset of specification statements from $\mathcal{S}$ exists. The element $w{:}[true, false] \in \mathcal{S}$ is above every element in $\mathcal{S}$. The result follows from (6). □

3. Conjunctive Predicate Transformers

In this section we show that $\mathcal{S}_0$ is the set of conjunctive predicate transformers. As a result, we also know that $\mathcal{S}_0$ forms a complete lattice [5]. We will also show that $\mathcal{S}$ is a proper subset of $\mathcal{S}_0$.

Lemma. (8)

Every statement from $\mathcal{S}_0$ is a conjunctive predicate transformer.

Proof: Let $\{i \triangleright P_i\}$ be a nonempty set of predicates. We will show that any statement from $\mathcal{S}_0$ is conjunctive.

$$\begin{array}{ll}
 & (w{:}[P, Q]).\langle \forall\ i \triangleright P_i \rangle \\
= & \quad \{\ (2)\text{: def. of the specification statement}\ \} \\
 & P \wedge \langle \forall\ w \mid Q \triangleright \langle \forall\ i \triangleright P_i \rangle\rangle[w_0 := w] \\
= & \quad \{\ \text{nesting of quantifiers}\ \} \\
 & P \wedge \langle \forall\ i \triangleright \langle \forall\ w \mid Q \triangleright P_i \rangle\rangle[w_0 := w] \\
= & \quad \{\ \text{distribution of substitution}\ \} \\
 & P \wedge \langle \forall\ i \triangleright \langle \forall\ w \mid Q \triangleright P_i \rangle[w_0 := w]\rangle \\
= & \quad \{\ \text{distribution of } \wedge \text{ over } \forall\text{—nonempty range}\ \} \\
 & \langle \forall\ i \triangleright P \wedge \langle \forall\ w \mid Q \triangleright P_i \rangle[w_0 := w]\rangle \\
= & \quad \{\ (2)\text{: def. of the specification statement}\ \} \\
 & \langle \forall\ i \triangleright (w{:}[P, Q]).P_i \rangle
\end{array}$$

□

Lemma. (9)

Every conjunctive predicate transformer is in $\mathcal{S}_0$.

Proof: The proof of this lemma follows from the observation that any conjunctive predicate transformer can be generated using the primitive commands assignment, assert, and assume, which are the specification statements

$v{:}[\mathit{true}, v = e[v := v_0]]$

$:[v = e, \mathit{true}]$

$:[\mathit{true}, v = e]$

respectively, and the constructors demonic choice and sequential composition [5]. The frame of a specification statement can be enlarged according to the following theorem from Morgan [14]:

$x{:}[P, Q] = x, y{:}[P, Q \wedge y_0 = y]$

Finally, specification statements are closed under demonic choice (see lemma (4)) and sequential composition. □

The following theorem follows immediately from the two lemmas above.

Theorem. (10)

The $\mathcal{S}_0$-specification statements are exactly the conjunctive predicate transformers.

Corollary. (11)

The $\mathcal{S}_0$-specification statements form a complete lattice.

Proof: Follows from the fact that the conjunctive predicate transformers form a complete lattice [5].

Having shown that the lattice of specification statements with initial-value variables is the complete lattice of conjunctive predicate transformers, we now show that introducing initial-value variables strictly increases the expressive power of specification statements.

Theorem.

$\mathcal{S} \neq \mathcal{S}_0$ (12)

i.e., $\mathcal{S}$ is a *proper* subset of $\mathcal{S}_0$.

Proof: We will show that $x{:}[true, x = x_0 + 1] \in \mathcal{S}_0$ has no counterpart in $\mathcal{S}$. Assume that it does have a counterpart, and let the counterpart be $w{:}[P, Q]$. Then, by definition, we have for all R

$$[(x{:}[true, x = x_0 + 1]).R \;\equiv\; (w{:}[P, Q]).R]$$

Using the definition of the specification statements, this simplifies to

$$[R[x := x + 1] \;\equiv\; P \wedge \langle \forall\, w \mid Q \triangleright R\rangle] \tag{13}$$

Instantiating (13) with $R := true$, we get $[true \;\equiv\; P]$. We now consider two cases: w is empty and w is nonempty.

If w is empty, instantiating (13) with $R := (x = 6)$ gives us

$$[x = 5 \;\equiv\; Q \Rightarrow x = 6] \tag{14}$$

However, $[x = 6 \Rightarrow$ RHS of (14)$]$, whereas $\neg[x = 6 \Rightarrow$ LHS of (14)$]$—a contradiction.

If w is nonempty, then it contains a variable, say y. Instantiating (13) with $R := (y = 6)$ yields

$$[(y = 6)[x := x + 1] \;\equiv\; \langle \forall\, w \mid Q \triangleright y = 6\rangle]$$

Notice that y is free on the left-hand side of the above (even if x and y coincide), but not on the right-hand side. Therefore the two predicates cannot be equal—a contradiction.

Both cases lead to a contradiction, completing the proof. □

4. Refinement

In this section, we use predicate calculus to directly formulate the refinement relation on $\mathcal{S}_0$-specification statements. As a special case, the theorem applies equally well to $\mathcal{S}$-specification statements.

Theorem. (*refinement*)

$$w{:}[P, Q] \sqsubseteq w{:}[P', Q'] \;\equiv\; [P \Rightarrow P'] \wedge [P \Rightarrow \langle \forall\, w \triangleright Q' \Rightarrow Q\rangle[w_0 := w]] \tag{15}$$

Proof: We prove this using a ping-pong argument.

Ping.

$$\begin{array}{ll}
 & w{:}[P, Q] \sqsubseteq w{:}[P', Q'] \\
= & \quad \{\ (5)\text{: def. of } \sqsubseteq\ \} \\
 & \langle \forall\, R \triangleright [(w{:}[P, Q]).R \Rightarrow (w{:}[P', Q']).R]\rangle \\
= & \quad \{\ (2)\text{: def. of the specification statement}\ \} \\
 & \langle \forall\, R \triangleright [P \wedge \langle \forall\, w \mid Q \triangleright R\rangle[w_0 := w] \Rightarrow P' \wedge \langle \forall\, w \mid Q' \triangleright R\rangle[w_0 := w]]\rangle \\
\Leftarrow & \quad \{\ \text{pred. calc.}\ \} \\
 & \langle \forall\, R \triangleright [(P \Rightarrow P') \wedge (P \wedge \langle \forall\, w \mid Q \triangleright R\rangle[w_0 := w] \Rightarrow \langle \forall\, w \mid Q' \triangleright R\rangle[w_0 := w])]\rangle \\
= & \quad \{\ \text{distribution of } \wedge \text{ over } [\;]\ \} \\
 & \langle \forall\, R \triangleright [P \Rightarrow P'] \wedge [P \wedge \langle \forall\, w \mid Q \triangleright R\rangle[w_0 := w] \Rightarrow \langle \forall\, w \mid Q' \triangleright R\rangle[w_0 := w]]\rangle \\
= & \quad \{\ \text{pred. calc.; distribution of substitution over } \Rightarrow\ \} \\
 & \langle \forall\, R \triangleright [P \Rightarrow P'] \wedge [P \Rightarrow (\langle \forall\, w \mid Q \triangleright R\rangle \Rightarrow \langle \forall\, w \mid Q' \triangleright R\rangle)[w_0 := w]]\rangle \\
\Leftarrow & \quad \{\ \text{substitution is monotonic; pred. calc.}\ \} \\
 & \langle \forall\, R \triangleright [P \Rightarrow P'] \wedge [P \Rightarrow \langle \forall\, w \triangleright Q' \Rightarrow Q\rangle[w_0 := w]]\rangle \\
= & \quad \{\ \text{pred. calc.}\ \} \\
 & [P \Rightarrow P'] \wedge [P \Rightarrow \langle \forall\, w \triangleright Q' \Rightarrow Q\rangle[w_0 := w]]
\end{array}$$

Pong.

$$\begin{array}{ll}
 & w{:}[P, Q] \sqsubseteq w{:}[P', Q'] \\
= & \quad \{\ (5)\text{: def. of } \sqsubseteq\text{; (2): def. of the specification statement}\ \} \\
 & \langle \forall\, R \triangleright [P \wedge \langle \forall\, w \mid Q \triangleright R\rangle[w_0 := w] \Rightarrow P' \wedge \langle \forall\, w \mid Q' \triangleright R\rangle[w_0 := w]]\rangle \\
= & \quad \{\ \text{interchange of quantifiers}\ \} \\
 & [\langle \forall\, R \triangleright P \wedge \langle \forall\, w \mid Q \triangleright R\rangle[w_0 := w] \Rightarrow P' \wedge \langle \forall\, w \mid Q' \triangleright R\rangle[w_0 := w]\rangle]
\end{array}$$

$$
\begin{array}{ll}
\Rightarrow & \{ \text{ pick } k \text{ fresh for } w,\ P,\ P',\ Q,\ Q',\ w_0 \ \} \\
& [\langle \forall\ k \mid k = w \triangleright \langle \forall\ R \triangleright P \wedge \langle \forall\ w \mid Q \triangleright R\rangle[w_0 := w] \Rightarrow P' \wedge \langle \forall\ w \mid Q' \triangleright R\rangle[w_0 := w]\rangle\rangle] \\
\Rightarrow & \{ \text{ instantiate with } R := Q[w_0 := k] \ \} \\
& [\langle \forall\ k \mid k = w \triangleright P \wedge \langle \forall\ w \mid Q \triangleright Q[w_0 := k]\rangle[w_0 := w] \Rightarrow P' \wedge \langle \forall\ w \mid Q' \triangleright Q[w_0 := k]\rangle[w_0 := w]\rangle] \\
= & \{ \text{ one-point rule—} k \text{ is fresh in } P,\ P',\ w,\ w_0 \ \} \\
& [P \wedge \langle \forall\ w \mid Q \triangleright Q[w_0 := k]\rangle[k, w_0 := w, w] \Rightarrow P' \wedge \langle \forall\ w \mid Q' \triangleright Q[w_0 := k]\rangle[k, w_0 := w, w]] \\
= & \{ \text{ dummy renaming, pick } t \text{ fresh for } Q,\ Q',\ w,\ w_0,\ k\text{; distribute substitution in } \} \\
& [P \wedge \langle \forall\ t \mid Q[w := t] \triangleright Q[w := t][w_0 := k]\rangle[k, w_0 := w, w] \\
& \quad \Rightarrow P' \wedge \langle \forall\ t \mid Q'[w := t] \triangleright Q[w := t][w_0 := k]\rangle[k, w_0 := w, w]] \\
= & \{ \text{ distribute substitution in—} k \text{ is fresh in } Q,\ Q',\ t \ \} \\
& [P \wedge \langle \forall\ t \mid Q[w := t][w_0 := w] \triangleright Q[w := t][w_0 := w]\rangle \\
& \quad \Rightarrow P' \wedge \langle \forall\ t \mid Q'[w := t][w_0 := w] \triangleright Q[w := t][w_0 := w]\rangle] \\
= & \{ \text{ pred. calc. } \} \\
& [P \wedge \mathit{true} \Rightarrow P' \wedge \langle \forall\ t \mid Q'[w := t] \triangleright Q[w := t]\rangle[w_0 := w]] \\
= & \{ \text{ pred. calc.; dummy renaming } \} \\
& [P \Rightarrow P'] \wedge [P \Rightarrow \langle \forall\ w \mid Q' \triangleright Q\rangle[w_0 := w]]
\end{array}
$$

□

As a notational convenience, we define the operator $\sim$ on predicates by:

$$[\sim P \ \equiv\ P[w, w_0 := w_0, w]] \tag{16}$$

We will frequently use the following properties of $\sim$:

$\sim$ is an involution, i.e., $\sim$ is its own inverse. (17)

$[P] \ \equiv\ [\sim P]$, since $[\ ]$ quantifies over all variables. (18)

$\sim$, like any substitution, distributes over all connectives. (19)

Corollary. (*refinement in $\mathcal{S}_0$*)
For $\mathcal{S}_0$-specification statements,

$$w{:}[P, Q] \sqsubseteq w{:}[P', Q'] \ \equiv\ [P \Rightarrow P'] \wedge [\sim P \wedge Q' \Rightarrow Q] \tag{20}$$

Proof:

$$
\begin{array}{ll}
& w{:}[P, Q] \sqsubseteq w{:}[P', Q'] \\
= & \{ \text{ (15): refinement theorem } \} \\
& [P \Rightarrow P'] \wedge [P \Rightarrow \langle \forall\ w \mid Q' \triangleright Q\rangle[w_0 := w]] \\
= & \{ \text{ (16): def. of } \sim\text{—}\langle \forall\ w \mid Q' \triangleright Q\rangle \text{ is independent of } w \ \} \\
& [P \Rightarrow P'] \wedge [P \Rightarrow \sim\langle \forall\ w \mid Q' \triangleright Q\rangle] \\
= & \{ \text{ (18); (19); (17); pred. calc. } \} \\
& [P \Rightarrow P'] \wedge [\sim P \Rightarrow \langle \forall\ w \triangleright Q' \Rightarrow Q\rangle] \\
= & \{ \text{ distribution of } \Rightarrow \text{ over } \forall\text{—}\sim P \text{ is independent of } w\text{, since } P \text{ is independent of } w_0 \ \} \\
& [P \Rightarrow P'] \wedge [\langle \forall\ w \triangleright \sim P \Rightarrow (Q' \Rightarrow Q)\rangle] \\
= & \{ \text{ [ ] quantifies over } w \ \} \\
& [P \Rightarrow P'] \wedge [\sim P \Rightarrow (Q' \Rightarrow Q)] \\
= & \{ \text{ pred. calc. } \} \\
& [P \Rightarrow P'] \wedge [\sim P \wedge Q' \Rightarrow Q]
\end{array}
$$

□

Corollary. (*refinement in $\mathcal{S}$*)
For $\mathcal{S}$-specification statements,

$$w{:}[P, Q] \sqsubseteq w{:}[P', Q'] \;\equiv\; [P \Rightarrow P'] \wedge [\langle \exists\, w \triangleright P\rangle \wedge Q' \Rightarrow Q] \tag{21}$$

Proof:

$$\begin{array}{ll}
 & w{:}[P, Q] \sqsubseteq w{:}[P', Q'] \\
= & \quad \{\ (20)\text{: refinement in } \mathcal{S}_0\ \} \\
 & [P \Rightarrow P'] \wedge [\sim P \wedge Q' \Rightarrow Q] \\
= & \quad \{\ [\ ]\text{ quantifies over } w_0\ \} \\
 & [P \Rightarrow P'] \wedge [\langle \forall\, w_0 \triangleright \sim P \wedge Q' \Rightarrow Q\rangle] \\
= & \quad \{\ \text{pred. calc.—} Q',\ Q \text{ are independent of } w_0\ \} \\
 & [P \Rightarrow P'] \wedge [\langle \exists\, w_0 \triangleright \sim P\rangle \wedge Q' \Rightarrow Q] \\
= & \quad \{\ (16)\text{: def. of } \sim\text{; dummy renaming—} P \text{ is independent of } w_0\ \} \\
 & [P \Rightarrow P'] \wedge [\langle \exists\, w \triangleright P\rangle \wedge Q' \Rightarrow Q]
\end{array}$$

□

5. Join Composition

As a direct consequence of (7) and (11), we know that there exists a choice operator that is the dual of demonic choice in the complete lattices of specification statements. In fact, we can use the construction in (6) to define the dual operation, which we denote by $\between$.

Definition. (*join composition*)

$$\between Z = \sqcap \{P, Q \mid \langle \forall\, P', Q' \mid w{:}[P', Q'] \in Z \triangleright w{:}[P', Q'] \sqsubseteq w{:}[P, Q]\rangle \triangleright w{:}[P, Q]\} \tag{22}$$

Since the specification statements form a complete lattice, there is some specification statement that corresponds to $\between Z$. The rest of this section is devoted to computing the closed form of this specification statement.

Theorem. (*join composition in* $\mathcal{S}_0$)
For $\mathcal{S}_0$-specification statements,

$$\between\{i \triangleright w{:}[P_i, Q_i]\} = w{:}[\langle \exists\, i \triangleright P_i\rangle, \langle \forall\, i \triangleright \sim P_i \Rightarrow Q_i\rangle] \tag{23}$$

Proof: For any arbitrary statement $w{:}[P, Q]$, we calculate,

$$\begin{array}{ll}
 & \between\{i \triangleright w{:}[P_i, Q_i]\} \sqsubseteq w{:}[P, Q] \\
= & \quad \{\ \text{def. of join (lattice theory)}\ \} \\
 & \langle \forall\, i \triangleright w{:}[P_i, Q_i] \sqsubseteq w{:}[P, Q]\rangle \\
= & \quad \{\ (20)\text{: refinement in } \mathcal{S}_0\ \} \\
 & \langle \forall\, i \triangleright [P_i \Rightarrow P] \wedge [\sim P_i \wedge Q \Rightarrow Q_i]\rangle \\
= & \quad \{\ \text{pred. calc. (term split; interchange of quantifiers)}\ \} \\
 & [\langle \forall\, i \triangleright P_i \Rightarrow P\rangle] \wedge [\langle \forall\, i \triangleright \sim P_i \wedge Q \Rightarrow Q_i\rangle] \\
= & \quad \{\ \text{pred. calc.}\ \} \\
 & [\langle \exists\, i \triangleright P_i\rangle \Rightarrow P] \wedge [Q \Rightarrow \langle \forall\, i \triangleright \sim P_i \Rightarrow Q_i\rangle] \\
= & \quad \{\ [\langle \forall\, i \triangleright \sim P_i \Rightarrow \mathit{false}\rangle \Rightarrow \langle \forall\, i \triangleright \sim P_i \Rightarrow Q_i\rangle]\ \} \\
 & [\langle \exists\, i \triangleright P_i\rangle \Rightarrow P] \wedge [Q \Rightarrow \langle \forall\, i \triangleright \sim P_i \Rightarrow \mathit{false}\rangle \vee \langle \forall\, i \triangleright \sim P_i \Rightarrow Q_i\rangle] \\
= & \quad \{\ \text{pred. calc. (shunting; De Morgan)}\ \} \\
 & [\langle \exists\, i \triangleright P_i\rangle \Rightarrow P] \wedge [Q \wedge \langle \exists\, i \triangleright \sim P_i\rangle \Rightarrow \langle \forall\, i \triangleright \sim P_i \Rightarrow Q_i\rangle] \\
= & \quad \{\ (19)\text{: distribution of } \sim \text{ over } \exists\ \} \\
 & [\langle \exists\, i \triangleright P_i\rangle \Rightarrow P] \wedge [Q \wedge \sim\langle \exists\, i \triangleright P_i\rangle \Rightarrow \langle \forall\, i \triangleright \sim P_i \Rightarrow Q_i\rangle] \\
= & \quad \{\ (20)\text{: refinement in } \mathcal{S}_0\ \} \\
 & w{:}[\langle \exists\, i \triangleright P_i\rangle, \langle \forall\, i \triangleright \sim P_i \Rightarrow Q_i\rangle] \sqsubseteq w{:}[P, Q]
\end{array}$$

Since $w{:}[P, Q]$ was arbitrary, we have established (23). □

The postcondition of the join of $\mathcal{S}_0$-specification statements may depend on initial-value variables even if the postconditions of the statements being joined do not. Thus, the join in $\mathcal{S}_0$ is not necessarily the same join as the one in $\mathcal{S}$. We continue by showing the closed form of the join in $\mathcal{S}$.

Theorem. (*join composition in* $\mathcal{S}$)
For $\mathcal{S}$-specification statements,

$$\triangle\{i \triangleright w{:}[P_i, Q_i]\} = w{:}[\langle \exists\ i \triangleright P_i \rangle, \langle \forall\ i \triangleright \langle \exists\ w \triangleright P_i \rangle \Rightarrow Q_i \rangle] \tag{24}$$

Proof: Similar to proof of (23). □

> *Remark.* Given a set of specification statements Z without initial-value variables, $\triangle Z$ in $\mathcal{S}_0$ is refined (in the lattice of all predicate transformers) by $\triangle Z$ in $\mathcal{S}$. (*End of Remark.*)

6. Universally Conjunctive Predicate Transformers

Back and von Wright show that the universally conjunctive predicate transformers form a complete lattice (referred to as $\mathcal{C}^{\top}_{\wedge}$) [5]. In this section we show that given any nonempty $Z \subseteq \mathcal{C}^{\top}_{\wedge}$, the join of the elements from Z in $\mathcal{C}^{\top}_{\wedge}$ is the same as their join in $\mathcal{S}_0$. We also relate the join of $\mathcal{C}^{\top}_{\wedge}$ to angelic choice, the join in the lattice of all predicate transformers. Throughout this section, we use $\triangle$ to denote the join in $\mathcal{S}_0$.

6.1. Joining statements in $\mathcal{C}^{\top}_{\wedge}$

We show that the lattice of universally conjunctive predicate transformers, $\mathcal{C}^{\top}_{\wedge}$, is a proper sublattice of the lattice of conjunctive predicate transformers.

Since $\mathcal{C}^{\top}_{\wedge}$ is a subset of $\mathcal{S}_0$, every universally conjunctive predicate transformer is a specification statement. The following lemma shows exactly which specification statements are universally conjunctive.

Lemma.

$$w{:}[P, Q] \text{ is universally conjunctive } \equiv [P \equiv \mathit{true}] \tag{25}$$

Proof: All specification statements in $\mathcal{S}_0$ are conjunctive. For a specification statement to be universally conjunctive, it must also distribute the empty conjunction. By (2), $(w{:}[P, Q]).\mathit{true} = P$, from which the lemma follows. □

The following lemma shows that $\mathcal{C}^{\top}_{\wedge}$ is closed under nonempty $\triangle$, the join from $\mathcal{S}_0$.

Lemma. (26)
If $Z \subseteq \mathcal{S}_0$ and there is an element $z \in Z$ such that z is universally conjunctive, then $(\triangle Z)$ is universally conjunctive.

Proof: From (23), the precondition of $(\triangle Z)$ is given by the disjunction of the preconditions of statements in Z. By (25), the precondition for z is *true*, which implies that the precondition for $(\triangle Z)$ is *true*. Applying (25) once more concludes the proof. □

Theorem.

$$\langle \mathcal{C}^{\top}_{\wedge}, \sqsubseteq, [\,], \triangle \rangle \text{ is a proper sublattice of } \langle \mathcal{S}_0, \sqsubseteq, [\,], \triangle \rangle \tag{27}$$

Proof: $\mathcal{C}^{\top}_{\wedge}$ is a subset of $\mathcal{S}_0$, and hence $\sqsubseteq$ is a partial order on $\mathcal{C}^{\top}_{\wedge}$. The dual of Theorem 2(f) from [5] shows that the meet operation in $\mathcal{C}^{\top}_{\wedge}$ is $[\,]$. Let Z be any nonempty subset of $\mathcal{C}^{\top}_{\wedge}$. By (26), $\triangle Z$ is universally conjunctive. Therefore it is the join in $\mathcal{C}^{\top}_{\wedge}$. We conclude that $\mathcal{C}^{\top}_{\wedge}$ is a sublattice of $\mathcal{S}_0$. $\mathcal{C}^{\top}_{\wedge}$ is a *proper* sublattice of $\mathcal{S}_0$ since, for example, $w{:}[\mathit{false}, \mathit{true}] \notin \mathcal{C}^{\top}_{\wedge}$. □

6.2. Join and angelic choice

Let $\sqcup$ denote angelic choice, i.e., the join operator in the complete lattice of all predicate transformers. When two conjunctive predicate transformers are composed using $\sqcup$, the result need not be conjunctive. Morgan shows a function $\square$ that maps an arbitrary predicate transformer S to the least-refined universally conjunctive transformer that refines S [13]. Formally, $\square$ is characterized, for any predicate transformer S, by

$$\square.S \in \mathcal{C}^{\top}_{\wedge} \quad \text{and} \tag{28}$$

$$\langle\forall\ c \mid c \in \mathcal{C}_\wedge^\top \triangleright S \sqsubseteq c \ \equiv\ \square.S \sqsubseteq c\rangle \tag{29}$$

We now show that

Theorem.
For any nonempty subset Z of universally conjunctive predicate transformers, we have

$$\mathbin{\Delta} Z = \square.(\sqcup\ Z) \tag{30}$$

Proof: From lattice theory we have for any lattice $\langle X, \leq, \downarrow, \uparrow\rangle$, with $Z \subseteq X$ and $u \in X$,

$$\uparrow Z = u \ \equiv\ \langle\forall\ x \mid x \in X \triangleright u \leq x \ \equiv\ \langle\forall\ z \mid z \in Z \triangleright z \leq x\rangle\rangle \tag{31}$$

With $u := \uparrow Z$, we obtain

$$\langle\forall\ x \mid x \in X \triangleright \uparrow Z \leq x \ \equiv\ \langle\forall\ z \mid z \in Z \triangleright z \leq x\rangle\rangle \tag{32}$$

For nonempty Z,

$$\begin{aligned}
& \mathbin{\Delta} Z = \square.(\sqcup\ Z) \\
= \quad & \{\ (27)\text{: } \mathbin{\Delta} \text{ is the join in } \mathcal{C}_\wedge^\top\text{, and (31) with } u := \square.(\sqcup\ Z)\text{, and (28) }\} \\
& \langle\forall\ c \mid c \in \mathcal{C}_\wedge^\top \triangleright \square.(\sqcup\ Z) \sqsubseteq c \ \equiv\ \langle\forall\ z \mid z \in Z \triangleright z \sqsubseteq c\rangle\rangle \\
= \quad & \{\ (29)\ \} \\
& \langle\forall\ c \mid c \in \mathcal{C}_\wedge^\top \triangleright (\sqcup\ Z) \sqsubseteq c \ \equiv\ \langle\forall\ z \mid z \in Z \triangleright z \sqsubseteq c\rangle\rangle \\
= \quad & \{\ (32) \text{ for the complete lattice of all pred. transformers }\} \\
& \langle\forall\ c \mid c \in \mathcal{C}_\wedge^\top \triangleright \langle\forall\ z \mid z \in Z \triangleright z \sqsubseteq c\rangle \ \equiv\ \langle\forall\ z \mid z \in Z \triangleright z \sqsubseteq c\rangle\rangle \\
= \quad & \{\ \text{pred. calc. }\} \\
& \mathit{true}
\end{aligned}$$

□

We also define $\square'$ for an arbitrary predicate transformer S by

$$\square'.S.Q \ \equiv\ \langle\exists\ R \triangleright S.R\rangle \wedge \square.S.Q$$

$\square'$ maps an arbitrary predicate transformer S to the least-refined transformer that is both conjunctive (but not necessarily universally conjunctive) and refines S.

Lemma.

$$\square'.S \in \mathcal{S}_0 \quad \text{and} \tag{33}$$

$$\langle\forall\ c \mid c \in \mathcal{S}_0 \triangleright S \sqsubseteq c \ \equiv\ \square'.S \sqsubseteq c\rangle \tag{34}$$

Proof: (33) follows from the definition of $\square'$, Morgan's (28), (25), and (10). The proof of (34) uses the results described in [13], and we give the following hint: One can define $\underline{B}$ and $\overline{B}$ in terms of Morgan's $\underline{A}$ and $\overline{A}$ (see [13]) and then show that $(\underline{B}, \overline{B})$ forms a Galois connection and that $\square' = \underline{B} \circ \overline{B}$. □

Theorem.
For any subset Z of conjunctive predicate transformers, we have

$$\mathbin{\Delta} Z = \square'.(\sqcup\ Z) \tag{35}$$

Proof: The proof is the same as for (30), but with $\mathcal{S}_0$ for $\mathcal{C}_\wedge^\top$, (33) for (28), and (34) for (29). □

7. Discussion

In this section, we examine the behavior of statements that are composed using $\mathbin{\Delta}$. We show that the join can be used in program development, and discuss its relationship with relational program semantics and parallel programs.

Letting Z denote the set $\{i \triangleright w{:}[P_i, Q_i]\}$, we examine $\mathbin{\Delta} Z$. According to (23) and (24), the precondition for $\mathbin{\Delta} Z$ is

$$\langle\exists\ i \triangleright P_i\rangle$$

which allows the statement to be executed whenever the precondition for any statement in Z is true. Contrast this with demonic choice (3) where all the preconditions have to be true.

The postcondition for $\mathcal{S}_0$-specification statements is given by

$\langle \forall\ i \triangleright \sim P_i \Rightarrow Q_i \rangle$

which can be read as follows: for every statement $w{:}[P_i, Q_i]$ whose precondition is true in the initial state, its postcondition is true on termination of $\between Z$. $\between Z$ does not necessarily guarantee that every Q_i holds on termination, but only that those Q_i for which the corresponding P_i holds initially, hold upon termination. Since the precondition of $\between Z$ implies that *some* P_i holds initially, *some* Q_i will hold in the final state.

The postcondition for $\mathcal{S}$-specification statements is given by

$\langle \forall\ i \triangleright \langle \exists\ w \triangleright P_i \rangle \Rightarrow Q_i \rangle$

which can be read as follows: for every statement $w{:}[P_i, Q_i]$ whose precondition is true for some value of w, its postcondition is true on termination of $\between Z$. Note that since the specification statement is allowed to change only those variables mentioned in the frame, $\langle \exists\ w \triangleright P_i \rangle$ will be true just when there is some assignment of w for which P_i holds in the *initial* state. In other words, $\langle \exists\ w \triangleright P_i \rangle$ will be true if, from the post-state it is possible to arrive back at a state that satisfies P_i by modifying only those variables mentioned in w.

Back and Butler introduce a *fusion* operator, denoted $\odot$, that is similar to our join and that satisfies similar properties [4]. However the join of two statements differs from their fusion, as can be seen by the following result from Back and Butler [4]:

$w{:}[P, Q] \odot w{:}[P', Q'] = w{:}[P \wedge P', Q \wedge Q']$

The fusion of two statements is refined by their join, with equality holding just when the two statements have equal preconditions. Universally conjunctive predicate transformers have *true* as their precondition; therefore the fusion and join of two universally conjunctive predicate transformers are equal.

7.1. Examples

To improve our understanding of $\between$, we give some examples of its operation. Consider the join of $x{:}[x = 0, x \bmod 2 = 0]$, $x{:}[x = 1, 10 \leq x]$, and $x{:}[y = 0, x = 20]$. The first of these statements is the statement that, if started in a state in which $x = 0$, ends in a state where x is even, having modified only x. The second statement requires precondition $x = 1$, and modifies the value of x to be at least 10. The third statement requires $y = 0$ and sets x to 20. Since these statements do not mention initial-value variables, they are in both $\mathcal{S}$ and $\mathcal{S}_0$.

We calculate the join of these statements in $\mathcal{S}_0$.

$$\begin{array}{ll}
 & x{:}[x = 0, x \bmod 2 = 0] \between x{:}[x = 1, 10 \leq x] \between x{:}[y = 0, x = 20] \\
= & \quad \{ \text{ (23): def. of } \between \text{ in } \mathcal{S}_0 \ \} \\
 & x{:}[x = 0 \vee x = 1 \vee y = 0, \\
 & \quad (\sim(x = 0) \Rightarrow x \bmod 2 = 0) \wedge (\sim(x = 1) \Rightarrow 10 \leq x) \wedge (\sim(y = 0) \Rightarrow x = 20)] \\
= & \quad \{ \text{ (16): def. of } \sim \ \} \\
 & x{:}[x = 0 \vee x = 1 \vee y = 0, (x_0 = 0 \Rightarrow x \bmod 2 = 0) \wedge (x_0 = 1 \Rightarrow 10 \leq x) \wedge (y = 0 \Rightarrow x = 20)] \ . \quad (36)
\end{array}$$

This is the statement that can start from an initial state where any of the three previous preconditions is satisfied. The statement guarantees that upon termination, x is even if $x = 0$ initially, x is at least 10 if $x = 1$ initially, and x is 20 if $y = 0$ initially. We now calculate the join of these three statements in $\mathcal{S}$.

$$\begin{array}{ll}
 & x{:}[x = 0, x \bmod 2 = 0] \between x{:}[x = 1, 10 \leq x] \between x{:}[y = 0, x = 20] \\
= & \quad \{ \text{ (24): def. of } \between \text{ in } \mathcal{S} \ \} \\
 & x{:}[x = 0 \vee x = 1 \vee y = 0, \\
 & \quad (\langle \exists\ x \triangleright x = 0 \rangle \Rightarrow x \bmod 2 = 0) \wedge (\langle \exists\ x \triangleright x = 1 \rangle \Rightarrow 10 \leq x) \wedge (\langle \exists\ x \triangleright y = 0 \rangle \Rightarrow x = 20)] \\
= & \quad \{ \text{ pred. calc. } \} \\
 & x{:}[x = 0 \vee x = 1 \vee y = 0, x \bmod 2 = 0 \wedge 10 \leq x \wedge (y = 0 \Rightarrow x = 20)] \quad . \qquad (37)
\end{array}$$

The precondition of this statement is the same as the precondition of (36). However, statement (37) always guarantees that upon termination, x is even and at least 10. Stated differently, although $\mathcal{S}$ is a subset of $\mathcal{S}_0$, $\mathcal{S}$ is not a sublattice of $\mathcal{S}_0$.

As a second example, the join of the statements $t{:}[\mathit{true}, t = 0]$ and $t{:}[\mathit{true}, t = 1]$ (which represent the assignment statements $t := 0$ and $t := 1$) is given by

$$
\begin{array}{cl}
 & t{:}[\mathit{true}, t = 0] \curlywedge t{:}[\mathit{true}, t = 1] \\
= & \quad \{ \text{ (23) or (24): def. of } \curlywedge \ \} \\
 & t{:}[\mathit{true}, t = 0 \wedge t = 1] \\
= & \quad \{ \text{ pred. calc. } \} \\
 & t{:}[\mathit{true}, \mathit{false}] \quad .
\end{array}
$$

The example shows that joining two specification statements that have disjoint postconditions produces the top element of $\mathcal{S}$ (or $\mathcal{S}_0$), a statement whose execution is always miraculous [14]. Contrast this to the angelic choice of the same two statements which angelically chooses to set t to 0 or 1 [5].

In certain circumstances, the join of a set of specification statements can be constructed in a simple manner. Let $\{i \triangleright w{:}[P_i, Q_i]\}$ be a set of specification statements with pairwise disjoint preconditions. Then the specification statement

$$\curlywedge\{i \triangleright w{:}[P_i, Q_i]\}$$

is the specification statement that corresponds to the statement

$$\textbf{if } \langle \square \ i \triangleright P_i \rightarrow w{:}[P_i, Q_i] \rangle \textbf{ fi}$$

(see [7]).

Although the join in the lattices $\mathcal{S}$ and $\mathcal{S}_0$ are distinct, in certain cases their application can result in the same specification statement. Let $\{i \triangleright Q_i\}$ be a set of predicates that are independent of initial-value variables. Then,

$$\langle \curlywedge \ i \triangleright w{:}[\mathit{true}, Q_i] \rangle$$

is the same in $\mathcal{S}$ and $\mathcal{S}_0$.

Consider the assignment statement $x := x + 1$ that is specified in $\mathcal{S}_0$ by $x{:}[\mathit{true}, x = x_0 + 1]$. An alternative method for specifying such statements in $\mathcal{S}_0$ is to use specification constants. The statement $x := x + 1$ would be written as follows using the specification constant X:

$$x{:}[x = X, x = X + 1]$$

Given a particular value of X, the statement above provides a partial specification for $x := x + 1$, namely the specification when the statement is begun in a state in which $x = X$. If we consider the join of the set of partial specifications obtained by assigning various values to X, we obtain $x := x + 1$. We calculate:

$$
\begin{array}{cl}
 & \langle \curlywedge \ X \triangleright x{:}[x = X, x = X + 1] \rangle \\
= & \quad \{ \text{ (23): def. of } \curlywedge \text{ in } \mathcal{S}_0 \ \} \\
 & x{:}[\langle \exists \ X \triangleright x = X \rangle, \langle \forall \ X \triangleright x_0 = X \Rightarrow x = X + 1 \rangle] \\
= & \quad \{ \ X \text{ ranges over the type of } x\text{; pred. calc. } \} \\
 & x{:}[\mathit{true}, x = x_0 + 1]
\end{array}
$$

The specification constant can be thought of as being implicitly bound by a $\curlywedge$-quantification in $\mathcal{S}_0$. A similar calculation in $\mathcal{S}$ results in $x{:}[\mathit{true}, \mathit{false}]$, which serves as an alternative proof that $\mathcal{S} \neq \mathcal{S}_0$.

7.2. Parallel programs

Consider the two statements $x, y{:}[\mathit{true}, x = 5]$ and $x, y{:}[\mathit{true}, y = 7]$. Their join (in both $\mathcal{S}$ and $\mathcal{S}_0$) is

$$x, y{:}[\mathit{true}, x = 5 \wedge y = 7]$$

In this case, the join produces a statement that has the effect of both of the statements being composed. Stated differently, in this case, the join corresponds to the parallel composition of the two statements.

The join does not always correspond to our operational understanding of parallel composition. For example, the join of $t := 0$ and $t := 1$ (see above) is miraculous; the parallel composition of these statements does not "make sense." As a comparison, in the temporal logic of actions (TLA) [12], specification are, crudely speaking, the postcondition component of $\mathcal{S}_0$-specification statements (except that TLA allows temporal logic formulas). Conjunction in TLA is thus related to our join operator. TLA uses conjunction to model parallel composition (cf. [2]), and, as in our case, the issue of this not always corresponding to parallel composition is present.

One can consider doing away with "interference" between statements to be composed in parallel by restricting their frames. For example, the statements

$$x{:}[P, Q] \text{ and } y{:}[P', Q'] \tag{38}$$

do not interfere with each other if x and y are disjoint, Q does not mention y or y_0, and Q' does not mention x or x_0. One can then consider expanding the frame of each of these statements to obtain

$$x, y{:}[P, Q] \text{ and } x, y{:}[P', Q'] \tag{39}$$

after which one can apply our join operator (which then produces the parallel composition). Ideally, we would want the result to be a refinement of each statement in (38). The join is indeed a refinement of each of the statements in (39)—by definition—but between (38) and (39) the refinements that hold:

$$x, y{:}[P, Q] \sqsubseteq x{:}[P, Q] \text{ and } x, y{:}[P', Q'] \sqsubseteq y{:}[P', Q'] \tag{40}$$

are in the opposite direction of what we would like. Thus one cannot hope to get from (38) to (40) via (39), but must go from (38) to (40) directly. A definition of "interference-free" in terms of implementations and their proofs can be found in [15].

7.3. Relational program semantics

There are many relational models of programs. A relational model that captures all conjunctive predicate transformers is that of R.M. Dijkstra [8]. The model uniformly captures both the weakest precondition and weakest *liberal* precondition semantics of programs [7]. However, in that model, relational intersection (conjunction) is not exactly our join. To illustrate, consider a state space consisting of one variable, x, which can take on one of two values, 0 or 1. Then, the join of the statements

$$x{:}[x = 0, x = 1] \tag{41}$$

$$x{:}[x = 1, x = 1] \tag{42}$$

is

$$x{:}[\mathit{true}, x = 1] \tag{43}$$

To model non-termination, the relational program model of Dijkstra uses an extra artificial state, ∞, and requires as a healthiness condition that every statement contain the transition from ∞ in the pre-state to ∞ in the post-state, and contain no other transition from ∞ in the pre-state. A program execution from a given state is seen as demonically choosing one of the possible transitions from that state, the empty demonic choice resulting in miraculous termination. The following depiction of relations correspond to statements (41) and (42), respectively.

$$\begin{array}{rcl} x = 0 & \longrightarrow & x = 1 \\ x = 1 & \longrightarrow & \infty \\ \infty & \longrightarrow & \infty \end{array} \qquad \begin{array}{rcl} x = 0 & \longrightarrow & \infty \\ x = 1 & \longrightarrow & x = 1 \\ \infty & \longrightarrow & \infty \end{array}$$

The conjunction (intersection) of these statements corresponds to

$$\infty \longrightarrow \infty$$

which equals the miraculous statement M (see [8]), and not (43). Thus relational intersection is different from our join.

However, it is also possible to represent statements (41) and (42) as:

$$\begin{array}{rcl} x = 0 & \longrightarrow & x = 1 \\ x = 1 & \longrightarrow & x = 0, x = 1, \infty \\ \infty & \longrightarrow & \infty \end{array} \qquad \begin{array}{rcl} x = 0 & \longrightarrow & x = 0, x = 1, \infty \\ x = 1 & \longrightarrow & x = 1 \\ \infty & \longrightarrow & \infty \end{array}$$

The intersection of these relations does indeed correspond to our join. To be sure the latter pair of representations are used for (41) and (42), one would have to require as an additional healthiness condition that from every normal state that can transition to ∞ (i.e., from which non-termination is possible), there is a possible transition to each of the other post-states. The problem with this condition is that statements are then no longer closed under composition, something that can be corrected by instead choosing the healthiness condition:

1. The transition ∞ to ∞ is present, and

2. For every initial state s (including ∞), if a transition to ∞ is possible from s, then there is also a transition from s to every post-state.

This healthiness condition would lead to a calculus quite different from Dijkstra's [8]. This model is explored in [10], and is the traditional model for relational program semantics.

Other applications of a join operator can be traced from [9], which uses yet another relational model. For example, the miraculous statement $w{:}[\mathit{true}, \mathit{false}]$ is not modeled.

7.4. Program development

The join of two specification statements $w{:}[P, Q]$ and $w{:}[P', Q']$ results in the weakest (conjunctive) specification that is a refinement of both the statements. We suggest how this fact can be used to one's advantage in program development.

Suppose that a large software project has many people working on it. Somehow, persons A and B both write specifications for procedure *proc*. Person A writes

$$w{:}[P, Q]$$

and person B writes

$$w{:}[P', Q']$$

Following this, programs are written that depend on one or the other of these specifications. When the mistake is noticed, a lot of work would be necessary to rectify the errors that were introduced as a result of assuming different specifications—unless one can find an implementation that meets both specifications. If the programming language at hand is one corresponding to the conjunctive predicate transformers, then the weakest specification such an implementation must satisfy is

$$w{:}[P, Q] \curlywedge w{:}[P', Q']$$

which we can now write in closed form. Implementing this specification would correct the errors introduced because of inconsistent specifications.

A similar situation can arise when using source code control systems. Suppose two programmers edit the same procedure and produce different implementations. Using the statements for the two implementations, one can compute the specification statement that describes an implementation that refines both implementations—the join of the two implementations. Methods for integrating two different implementations of programs have been proposed before (see, for instance [11]). The join operation computed here is the basis for reasoning about the correctness of such a method.

The join is also the basis for the paradigm of program construction by parts [9]. The paradigm proposes that a specification S be written as the join of specifications S' and S'', i.e., $S = S' \curlywedge S''$. The specifications S' and S'' capture partial requirements of specification S. Therefore, these component specifications are simpler, leading to a systematic methodology for program development. Frappier et al. give the detailed construction of a program using this paradigm [9].

Ainsworth and Wallis introduce a *union* operator, denoted $\uplus$, that is equivalent to our join [1]. They simply define this operator and note that it is a hybrid of two other operators: disjunction and fusion. They proceed to state other properties of union and use it in their definition of co-refinement by multiple viewpoints. Since we have shown that the union operator is in fact the join in the lattice of specifications, this shows that their work is closely related to the paradigm of program construction by parts.

8. Summary

We showed that the set of all specification statements, $\mathcal{S}_0$, and the subset thereof that does not use initial-value variables, $\mathcal{S}$, each forms a complete lattice. We showed that $\mathcal{S}_0$ is the lattice of conjunctive predicate transformers, and that $\mathcal{S}$ differs from $\mathcal{S}_0$. A simple predicate calculus formulation of refinement in the lattice of specification statements was provided. The join operations in $\mathcal{S}$ and $\mathcal{S}_0$ were calculated and explained. These two operations were shown to be different, proving that $\mathcal{S}$ is not a sublattice of $\mathcal{S}_0$. We showed that the lattice of universally conjunctive predicate transformers, $\mathcal{C}_\wedge^\top$, is a proper sublattice of $\mathcal{S}_0$, and related the join to angelic choice in the lattice of all predicate transformers. Finally, we mentioned related work and showed an application of the join in program development.

Acknowledgments

We would like to thank Rutger Dijkstra for helpful discussions. He suggested that we introduce $\sim$, which simplified a number of proofs. In addition, he suggested the relation to specification constants. We thank the referees for their valuable comments. Rick Hehner and the referees pointed us to some related work. Paul Gardiner's suggestion of defining $\square'$ in terms of $\square$ and his suggested definition led to our definition of $\square'$.

References

[1] Mike Ainsworth and Peter J.L. Wallis. Co-Refinement. *Proceedings of the 6th Refinement Workshop*, pp. 151–166. Springer-Verlag, 1994.

[2] M. Abadi and L. Lamport. Conjoining Specifications. *ACM Transactions on Programming Languages and Systems*, **17**(3):507–534, May 1995.

[3] R.J.R. Back. *On the correctness of refinement steps in program development.* PhD thesis, University of Helsinki, 1978. Report A-1978-4.

[4] R.J.R. Back and M.J. Butler. Exploring Summation and Product Operations in the Refinement Calculus. *Mathematics of Program Construction: Third International Conference*. Lecture Notes in Computer Science 947, pp. 128–158. Springer-Verlag, 1995.

[5] R.J.R. Back and J. von Wright. Combining angels, demons and miracles in program specifications. *Theoretical Computer Science* **100**(2):365–383. 1992

[6] G. Birkhoff. *Lattice Theory.* Colloquium Publications, Volume 25. American Mathematical Society, 1967.

[7] E.W. Dijkstra and C.S. Scholten. *Predicate Calculus and Program Semantics.* Springer-Verlag, 1990.

[8] R.M. Dijkstra. Relational Calculus and Relational Program Semantics. Technical report CS-R9408, University of Groningen, 1994.

[9] M. Frappier, A. Mili, and J. Desharnais. Program Construction by Parts. In *Mathematics of Program Construction: Third International Conference.* Lecture Notes in Computer Science 947, pp. 257–281. Springer-Verlag, 1995.

[10] C.A.R. Hoare and J.F. He. The weakest prespecification. *Fundamenta Informaticæ*, IX:51–84, 1986.

[11] S. Horowitz, J. Prins, and T. Reps. Integrating Noninterfering Versions of Programs. *ACM Transactions on Programming Languages and Systems*, **11**(3):345–387. July, 1989.

[12] L. Lamport. The Temporal Logic of Actions. *ACM Transactions on Programming Languages and Systems*, **16**(3):872–923, May 1994.

[13] C.C. Morgan. The Cuppest Capjunctive Capping, and Galois. In A.W. Roscoe, editor, *A Classical Mind: Essays in Honour of C.A.R. Hoare*, Prentice-Hall International Series in Computer Science, pp. 317–332. Prentice-Hall, 1994.

[14] C.C. Morgan. The specification statement. *ACM Transactions on Programming Languages and Systems*, **10**(3):403–419, July 1988.

[15] S. Owicki and D. Gries. An axiomatic proof technique for parallel programs. *Acta Informatica*, **6**:319–340, 1976.

[16] J.L.A. van de Snepscheut. On Lattice Theory and Program Semantics. Caltech technical report CS-TR-93-19, 1993.