On Refinement Calculus and Partial Correctness (DRAFT)*

R.D. Arthan

2 October 2011

1 Introduction

1.1 Background and Motivation

This note is intended as the first of a series concerned with a style of system specification in which the claims that can be verified have the form "system p satisfies specification s under normal circumstances". This notion of correctness is traditionally referred to as partial correctness, although a more impartial term such as "algorithmic correctness" is preferred by some. A total correctness claim has the form "system p always satisfies specification s".

I should explain what is meant by "normal circumstances": we have in a mind some notion of mathematically rigorous system specification. Such a notion will always be relative to some conventions and assumptions about the relationship between the mathematical model and the physical systems under consideration. For example, in program specification, the modelling conventions might be the abstractions involved in a formal semantics for the programming language and the modelling assumptions might include an assumption that no exceptions are raised causing control to pass from the programming language execution environment to the operating system under which it runs. We take normal circumstances to be those in which the modelling assumptions hold — abnormal circumstances are those in which the model breaks down and something external to the model is required to deal with the situation.

For many practical purposes, partial correctness is a much more useful notion than total correctness. For example, in program analysis, tools other than formal specification and proof may be better suited to automatic analysis of normal termination. Furthermore, partial correctness claims have the clear advantage of sometimes being true in physical reality, whereas total correctness claims never are (e.g., consider what happens when a power supply fails). We believe the partial correctness approach is much closer in spirit to normal engineering practice: no engineered artefact is perfect. Much of the work in any engineering task is in combining potentially faulty subsystems so as to produce a system that is adequately resilient for its intended purpose.

Some technologies, such as programming languages, apparently offer the possibility of formal proof of total correctness. However, such a proof is always relative to some mathematical

model which is invariably an idealisation of the physical reality of the delivered artefact. While this possibility is of considerable theoretical importance, we believe that there is considerable practical value in having a mathematically rigorous theory that reflects the potential for failure in any engineered component.

The original motivation for this work lay in program specification. However, it became clear that with just a little additional abstraction, much of the theory has a much wider domain of application. In this first note, we look at a very general class of systems and system specifications.

The most general notion of the act of specifying a system is first to identify some set P of possible systems and then to identify a subset $A \subseteq P$ of acceptable systems. We may call this the *property-based* approach. It is advocated in Jones [2] as a general method for specifying and verifying critical system properties. The advantages and disadvantages of this view both lie in its generality: on the one hand, it is clearly adequate to deal with any system property that is amenable to any form of mathematical specification, on the other hand it gives no guidance whatsoever as to how to proceed: the modelling conventions and assumptions in any given situation need to be carefully designed and inspected to ensure they are fit for their intended purpose.

An approach that has a very long tradition, particularly in program verification, is to observe that almost any system can be viewed as some kind of binary relation, say between inputs and outputs, or between successive states of a state transition machine. This *relational* approach leads to a notion of *refinement* whereby implementations are viewed as binary relations of the same kind as are used for specifications and refinement rules are given defining what a valid implementation of a specification is. (The corresponding notion for the property-based approach is just set-theoretic inclusion or membership. The refinement rules give more useful insights in the relational approach.)

In a relatively recent work, Hoare & He [1], this approach is advocated as providing a unifying framework for dealing with a wide class of programming languages and programs. However, the approach as described in Hoare & He [1] is heavily biased towards the total correctness viewpoint. This actually leads to considerable technical complications and the solutions taken in Hoare & He [1] lead to counter-intuitive results: for example, they preclude the design of a system which can create order from chaos — a very common engineering requirement.

It is certainly possible to adapt the relational approach to give a notion of partial correctness. Indeed, the approach we will take in this note embraces this as a special case. However, viewing specifications as binary relations leads to a significant loss of expressiveness in the partial correctness version. The problem is that when dealing with the possibility of failure, the specifier may need to demand failure under certain circumstances. Unfortunately, the relational approach gives no way of distinguishing between cases where failure is required and cases where the behaviour is unspecified. Our approach is to augment the binary relation with a set indicating the domain in which the specifier has a definite interest in behaviour. This is, essentially just the pre- and post-condition style of specification that is familiar from Floyd-Hoare logic and notations like VDM. However, the partial correctness interpretation of the pre-condition makes it play a much more important role than it does in a total correctness approach (which is why Hoare & He [1] are able to do without it, except as a syntactic convention).

Please note this is work-in-progress and the current version of this note lacks any serious

attempt to cite the literature or compare the results with other work.

1.2 Notation

We will develop our theory using the Z notation. We will give definitions in Z and state our theorems as Z conjectures. The Z definitions have all been type-checked and the conjectures have all been proved using the ProofPower system. The master of this document is in fact a ProofPower literate script which can be processed automatically to check the definitions and replay the proofs. For brevity, the proof commands are suppressed from the printed form of the document.

We use infix notation for the following relation and function symbols:

$$\vdash \textit{relation } \_ \sqsubseteq \_,\ \_ \models \_,\ \_ \equiv \_$$

$$\vdash \textit{function } 30\ \_ \sqcup \_$$

$$\vdash \textit{function } 40\ \_ \sqcap \_$$

$$\vdash \textit{function } 40\ \ \_ \oplus_R \_$$

We use postfix notation for the following function and generic symbols:

$$\vdash \textit{function } 7\ \_^{\$}$$

$$\vdash \textit{generic } 7\ \_^{\perp}$$

As mentioned above, all of the conjectures stated in this document have been proved with ProofPower and the resulting theory listing is included as an appendix. An index to the Z specification is given at the end of the document.

2 Inputs, Outputs , Pre-conditions, Post-conditions

We wish to construct a semantic model of specification of operations via pre- and post-conditions. Our operations have inputs and outputs that could be memory states, or data streams or anything else appropriate to the modelling task at hand. The internal structure of inputs and outputs is not relevant to our purposes for the time being — our definitions will be generic with respect to the set, X, of inputs and the set, Y, of outputs.

For modelling the semantics of a conventional imperative programming language, the inputs and outputs would both be assignments of values to program variables. We take this as our motivating example in this section.

In our formulation, a predicate on a set, X, is just a subset of X, i.e., the denotation of the usual syntactic notion of a predicate.

$$\mid \boldsymbol{PRED}[X] \mathrel{\widehat{=}} \mathbb{P}X$$

For the programming language example, a pre-condition would be given syntactically as a predicate whose free variables are program variables and a post-condition would be given as a predicate whose free variables are program variables with optional decoration to distinguish values in the before-state from values in the after-state. A pre-condition denotes the set of assignments, α, in which the program variables satisfy the pre-condition and a post-condition denotes the relation, ρ, which holds between a before-state s and an after-state s' when $s \times s'$ satisfies the post-condition after making the appropriate binding of before- and after-values to the variables in the post-condition.

Taken together, we can often think of the pre- and post-condition together as denoting the relation $\alpha \lhd \rho$. However, as we will be making formal presently, our reading of a specification will require the program not to terminate in before-states that satisfy the pre-condition but to which the post-condition cannot respond. Consequently we lose some expressiveness if we throw away the pre-condition, for example, if the post-condition is unsatisfiable, the pre-condition defines a set of states in which the program must not terminate. We therefore keep a separate record of the pre-condition. So a specification is a pair comprising a predicate giving the pre-condition and a binary relation between inputs and outputs giving the post-condition.

$$\mid \boldsymbol{PRE\_COND}[X] \mathrel{\widehat{=}} PRED[X]$$

$$\mid \boldsymbol{POST\_COND}[X,\ Y] \mathrel{\widehat{=}} X \leftrightarrow Y$$

$$\mid \boldsymbol{SPEC}[X,\ Y] \mathrel{\widehat{=}} PRE\_COND[X] \times POST\_COND[X,\ Y]$$

To give examples in the sequel, we will borrow some syntax from the Compliance Notation, in which states comprise assignments of values to program variables representing the execution state of an Ada program. In the Compliance Notation, specifications are written using *specification statements* which have the following general form:

$$\Delta\ \ \mathcal{W}\ [\ \mathcal{P},\ \ \mathcal{Q}\ ]$$

where $\mathcal{W}$ is a list of program variables called the *frame* and $\mathcal{P}$ and $\mathcal{Q}$ are syntactic predicates giving the pre-condition and post-condition respectively. In the post-condition a subscript 0 may be used to distinguish variables that refer to the before-state from variables that refer to the after-state. The frame lists the program variables that may be changed by the code being specified. For example, here are specification statements for *(a)* a fragment of code that exchanges the value of two program variables, X and Y, possibly with a side-effect on a third variable, T; and, *(b)* a fragment of code that divides one variable, Y, by another, X, subject to the pre-condition that X be positive.

$$\Delta\ X,\ Y,\ T\ [true,\ X = Y_0 \wedge Y = X_0] \qquad (a)$$
$$\Delta\ Y\ [X > 0,\ Y = Y_0\ /\ X\ ] \qquad (b)$$

In our semantic view, a syntactic specification statement of this sort denotes a specification in the above semantic sense whose pre-condition is the denotation of the given syntactic pre-condition and whose post-condition is the relation denoted by the given post-condition conjoined with the requirement that any variable not listed in the frame must be unchanged.

3 Refinement

Refinement is a notion that is fundamental to this work. Refinement is the relation that obtains between a specification and a satisfactory implementation of that specification, where, in the present context an "implementation" is simply a specification, typically more definite than the specification it refines. Since we are only concerned with partial correctness, we consider an implementation to be satisfactory even if it fails to respond in some situations where the specification appears to require a response. More formally, we will say that one specification, $s_2 = (prec_2,\ postc_2)$ refines another $s_1 = (prec_1,\ postc_1)$ iff:

- $prec_2$ includes $prec_1$; and:
- the restriction of $postc_2$ to $prec_1$ is contained in $postc_1$.

The first of these conditions is a vestigial analogue of liveness in our partial correctness view of refinement. The traditional notion of liveness allows the pre-condition to be weakened but requires $postc_2$ to be at least as responsive as $postc_1$ in states where $prec_1$ holds. Here, we do not impose the latter requirement. We just say is that s_2 may not strengthen the pre-condition of s_1: this amounts to saying that s_2 must not be less specific than s_1 in its requirements for non-termination.

The second condition is the traditional notion of safety — it says that any response made by s_2 in a state satisfying the pre-condition of s_1 is a response that could also be made by s_1. This includes the possibility that s_2 may be unable to respond where s_1 can.

$$
\begin{array}{|l}
\hline
[X, Y] \\
\_ \sqsubseteq \_ : SPEC[X,\ Y] \leftrightarrow SPEC[X,\ Y] \\
\hline
\forall prec_1,\ prec_2 : PRE\_COND[X];\ postc_1,\ postc_2 : POST\_COND[X,\ Y] \bullet \\
\quad (prec_1,\ postc_1) \sqsubseteq (prec_2,\ postc_2) \\
\Leftrightarrow \quad prec_1 \subseteq prec_2 \\
\wedge \quad prec_1 \lhd postc_2 \subseteq postc_1 \\
\hline
\end{array}
$$

We now explore the properties of the refinement relation. First of all, we note that refinement is a pre-order (i.e., it is reflexive and transitive):

$$
\begin{array}{|l}
refinement\_pre\_order\_cnj\ ?\vdash \\
\quad \forall\ s_1,\ s_2\ ,\ s_3 : SPEC \bullet \\
\qquad s_1 \sqsubseteq s_1 \\
\wedge \qquad (s_1 \sqsubseteq s_2 \wedge s_2 \sqsubseteq s_3 \Rightarrow s_1 \sqsubseteq s_3)
\end{array}
$$

However, refinement is not antisymmetric, i.e., it is possible to have $s_1 \sqsubseteq s_2$ and $s_2 \sqsubseteq s_1$ without having $s_1 = s_2$. This happens because two specifications with the same pre-condition may differ only with respect to responses to before-states that do not satisfy the pre-condition

and are therefore irrelevant in our definition of refinement[1]. To remedy this, where necessary, let us say that two specifications are equivalent if they refine one another:

$$[X, Y]$$

$$\_ \equiv \_ : SPEC[X,\ Y] \leftrightarrow SPEC[X,\ Y]$$

$$\forall s_1,\ s_2 : SPEC[X,\ Y] \bullet$$
$$s_1 \equiv s_2 \Leftrightarrow s_1 \sqsubseteq s_2 \land s_2 \sqsubseteq s_1$$

That $\equiv$ is an equivalence relation follows from the fact that $\sqsubseteq$ is a pre-order. The pre-order induces a pre-order on the equivalence classes which will necessarily also be a partial order, i.e., it will also be antisymmetric. Rather than work with equivalence classes in the sequel, we will work with canonical representatives when necessary. These canonical representatives are defined by the following which reduces a specification to a normal form in which the post-condition is empty outside the pre-condition.

$$[X, Y]$$

$$\_^{\$} : SPEC[X,\ Y] \to SPEC[X,\ Y]$$

$$\forall prec : PRE\_COND[X];\ postc : POST\_COND[X,\ Y] \bullet$$
$$(prec,\ postc)^{\$} = (prec,\ prec \lhd postc)$$

The reduction operator (which is useful enough to be worth the dollar in its name!) is idempotent:

$$reduce\_reduce\_cnj\ ?\vdash\ \forall\ s : SPEC \bullet\ (s^{\$})^{\$} = s^{\$}$$

The following theorem shows that the reduction operator does indeed pick a canonical representative from each $\equiv$-equivalence class:

$$equiv\_cnj\ ?\vdash\ \forall\ s_1,\ s_2 : SPEC \bullet\ s_1 \equiv s_2 \Leftrightarrow {s_1}^{\$} = {s_2}^{\$}$$

Our next theorem says that refinement is independent of choice of representative in each equivalence class:

$$refinement\_reduce\_cnj\ ?\vdash\ \forall\ s_1,\ s_2 : SPEC \bullet\ s_1 \sqsubseteq s_2 \Leftrightarrow {s_1}^{\$} \sqsubseteq {s_2}^{\$}$$

Let us define the *reduced* specifications to be those in the range of the reduction operator:

[1]We could require all our specifications to be such that the pre-condition contains the domain of the post-condition. We prefer not to impose this restriction except where necessary. This is technically more convenient and is also more faithful to actual examples, e.g, in the example, $\Delta Y[X > 0,\ Y = Y_0/X]$ that we have already discussed, the denotation of the post-condition considered in isolation includes the possibility that X be negative in the before-state.

$[X, Y]$

$\boldsymbol{REDUCED} : \mathbb{P} SPEC[X,\ Y]$

$REDUCED\ = \{s : SPEC \bullet\ s^{\$}\}$

When restricted to reduced specifications, refinement does indeed become antisymmetric:

$refinement\_antisym\_cnj\ ?\vdash\ \forall\ s_1,\ s_2 : REDUCED \bullet\ s_1 \sqsubseteq s_2 \land s_2 \sqsubseteq s_1 \Rightarrow s_1 = s_2$

We now want to show that refinement of reduced specifications is a complete lattice. We define constants that help us state this and other results. We use upper-case names for the constituents of specifications and mixed upper-lower case names for specifications.

$[X]$

$\boldsymbol{TRUE}, \boldsymbol{FALSE} : PRE\_COND[X]$

$TRUE = X;$
$FALSE = \varnothing$

$[X, Y]$

$\boldsymbol{ABORT} : POST\_COND[X,\ Y];$
$\boldsymbol{CHAOS} : PRE\_COND[X] \to POST\_COND[X,\ Y];$
$\boldsymbol{Abort}, \boldsymbol{Chaos} : PRE\_COND[X] \to SPEC[X,\ Y];$
$\boldsymbol{Bottom}\ , \boldsymbol{Top} : SPEC[X,\ Y]$

$ABORT = \varnothing;$
$\forall\ prec : PRE\_COND[X] \bullet$
$\qquad CHAOS\ prec = prec \times Y$
$\land \qquad Abort\ prec = (prec,\ ABORT)$
$\land \qquad Chaos\ prec = (prec,\ CHAOS\ prec);$
$Bottom = Chaos\ \varnothing;$
$Top = Abort\ TRUE$

Bottom and top are indeed the bottom and top elements of the pre-order:

$bottom\_top\_cnj\ ?\vdash\ \forall\ s : SPEC \bullet\ Bottom \sqsubseteq s \sqsubseteq Top$

Meets (greatest lower bounds) turn out to be straightforward to define and verify:

$[X, Y]$

$$\textbf{\textit{Meet}} : \mathbb{P} SPEC[X,\ Y] \to SPEC[X,\ Y]$$

$$\forall A : \mathbb{P}\ SPEC[X,\ Y]\bullet\ Meet\ A = (\bigcap\{s : A\bullet\ s.1\},\ \bigcup\{s : A\bullet\ s.2\})$$

Now we can state the conjecture that meets are indeed greatest lower bounds.

refinement_meet_cnj ?⊢

$$\forall\ A : \mathbb{P} SPEC\bullet\ \forall t : A\bullet\ Meet\ A \sqsubseteq t;$$
$$\forall\ A : \mathbb{P} SPEC;\ s : SPEC\bullet\ (\forall t : A\bullet\ s \sqsubseteq t) \Rightarrow s \sqsubseteq Meet\ A$$

Joins turn out to be trickier. We first define a function that maps a specification, s say to the post-condition that allows precisely those state transitions that s forbids:

$[X, Y]$

$$\textbf{\textit{Forbidden}} : SPEC[X,\ Y] \to POST\_COND[X,\ Y]$$

$$\forall s : SPEC[X,\ Y]\bullet$$
$$Forbidden\ s = CHAOS\ (s.1) \setminus s.2$$

A more explicit equation for forbidden sets is useful[2]:

forbidden_def_cnj ?⊢

$$\forall\ s : SPEC\bullet$$
$$Forbidden\ s = \{\ x : \mathbb{U};\ y : \mathbb{U} \mid x \in s.1 \land \neg(x,\ y) \in s.2\}$$

The following conjecture is useful to check the definition of the forbidden function and in proving later results.

forbidden_cnj ?⊢

$$\forall\ s_1,\ s_2 : SPEC\bullet$$
$$s_1 \sqsubseteq s_2 \Leftrightarrow s_1.1 \subseteq s_2.1 \land s_2.2 \cap (Forbidden\ s_1) = \varnothing$$

Now we can define joins. The pre-condition of the join of a set of specifications is just the union of the individual pre-conditions. The post-condition is obtained from the union of the post-conditions by removing all transitions that are forbidden by some specification in the set.

$[X, Y]$

$$\textbf{\textit{Join}} : \mathbb{P} SPEC[X,\ Y] \to SPEC[X,\ Y]$$

$$\forall A : \mathbb{P}\ SPEC[X,\ Y]\bullet$$
$$Join\ A = (\bigcup\{s : A\bullet\ s.1\},\ (\bigcup\{s : A\bullet(s^{\$}).2\}) \setminus (\bigcup\{s : A\bullet Forbidden\ s\}))$$

[2]For convenience in stating this and other theorems, we use the generic constant $\mathbb{U}$ which is defined as part of the ProofPower Z library as if by the generic definition $\mathbb{U}[X] \widehat{=} X$, so that $\mathbb{U}$ with the generic parameter left implicit denotes the universal set of whatever type is required by the context.

In elementary working with join it is often more convenient to have more explicit set comprehensions:

$$
\begin{array}{l}
\mid\ join\_def\_cnj\ ?\vdash \\
\mid\qquad \forall\ A : \mathbb{P} SPEC\bullet\ Join\ A = \\
\mid\qquad\qquad (\{s : A;\ x : \mathbb{U} \mid x \in s.1\bullet\ x\}, \\
\mid\qquad\qquad \{s : A;\ x : \mathbb{U};\ \ y : \mathbb{U} \\
\mid\qquad\qquad \mid\ x \in s.1 \land (x,\ y) \in s.2 \land (\forall t : A\bullet\ x \in t.1 \Rightarrow (x,\ y) \in t.2) \\
\mid\qquad\qquad \bullet\ (x,\ y)\})
\end{array}
$$

It is useful to be able to calculate the forbidden set of a join

$$
\begin{array}{l}
\mid\ forbidden\_join\_cnj\ ?\vdash \\
\mid\qquad \forall\ A : \mathbb{P} SPEC\bullet \\
\mid\qquad\qquad Forbidden(Join\ A) = \bigcup\{s : A\bullet\ Forbidden\ s\}
\end{array}
$$

Now we can state the theorem that our joins are indeed least upper bounds.

$$
\begin{array}{l}
\mid\ refinement\_join\_cnj\ ?\vdash \\
\mid\qquad \forall\ A : \mathbb{P} SPEC\bullet\ \forall t : A\bullet\ t \sqsubseteq Join\ A; \\
\mid\qquad \forall\ A : \mathbb{P} SPEC;\ s : SPEC\bullet\ (\forall t : A\bullet\ t \sqsubseteq s) \Rightarrow Join\ A \sqsubseteq s
\end{array}
$$

Given that arbitrary meets and joins exist, we can now define binary meets and joins in terms of them. At this point, the observant reader who is not familiar with the traditions of our style of specification will note that the passage from the language of set-theoretic inclusion, union and intersection, to the language of refinement, meets and joins involves not only squaring off the corners of the symbols but also looking at them upside down in a mirror. This is perhaps unfortunate, but it is the tradition and we feel obliged to follow it.

$$
\begin{array}{|l}
\hline
[X, Y] \\
\_\sqcap\_,\ \_\sqcup\_ : SPEC[X,\ Y] \times SPEC[X,\ Y] \to SPEC[X,\ Y] \\
\hline
\forall s_1,\ s_2 : SPEC[X,\ Y]\bullet \\
\qquad s_1 \sqcap s_2 = Meet\{s_1,\ s_2\} \\
\land \qquad s_1 \sqcup s_2 = Join\{s_1,\ s_2\} \\
\hline
\end{array}
$$

The following formulae for binary meets and joins are often simpler to use (and probably easier to understand) than the general definitions:

$$
\mid\ meet2\_cnj\ ?\vdash \forall s_1,\ s_2 : SPEC\bullet\ s_1 \sqcap s_2 = (s_1.1 \cap s_2.1,\ s_1.2 \cup s_2.2)
$$

$$
\begin{array}{l}
\mid\ join2\_cnj\ ?\vdash \forall s_1,\ s_2 : SPEC\bullet \\
\mid\qquad s_1 \sqcup s_2 = \\
\mid\qquad (\qquad s_1.1 \cup s_2.1, \\
\mid\qquad\qquad ((s_1.1 \cap s_1.1) \lhd (s_1.2 \cap s_2.2)) \\
\mid\qquad \cup \qquad ((s_1.1 \setminus s_2.1) \lhd s_1.2) \\
\mid\qquad \cup \qquad ((s_2.1 \setminus s_1.1) \lhd s_2.2))
\end{array}
$$

4 Relations with Other Approaches

In Hoare & He [1], a theory based on total correctness is developed. This theory is syntactic and involves various artifices that are on the face of it mainly introduced to obtain various algebraic laws. A semantics and rather simpler account is given in Woodcock and Davies[3]. Total correctness effectively means restricting attention to specification in our sense in which the domain of the post-condition contains the pre-condition. This has the advantage that, if we take the canonical representation, we can discard the pre-condition since it may be recovered as the domain of the post-condition.

Woodcock and Davies[3] proceed to point out that for total relations, the notion of refinement reduces to set-theoretic inclusion of relations. In this section we show that this idea generalises to our partial correctness notion. To do this we first introduce some more notation: $X^\perp$ is to be the result of augmenting a set X with an additional element $\perp$. We represent $X^\perp$ as the set of all subsets of X with at most one element:

$$\boldsymbol{X}^{\perp} \mathrel{\widehat=} \{\ A : \mathbb{P}X \mid \forall x,\ y : A \bullet\ x\ = y\}$$

ι and $\perp$ are then defined as follows (and act very much as if they were constructors of a generic free type).

$[X]$

$$\perp : X^{\perp}\ ;$$
$$\boldsymbol{\iota} : X \to X^{\perp}$$

$$\perp = \varnothing;$$
$$\forall x : X \bullet \quad \iota\ x = \{x\}$$

The function *lift* maps a specification in our sense to the relation between augmented inputs and augmented outputs which agrees with the post-condition where the pre-condition holds and is chaos outside the pre-condition. This definition agrees with the one given in Woodcock and Davies[3] on specifications that happen to be total[3].

$[X, Y]$

$$\boldsymbol{lift} : SPEC[X,\ Y] \to X^{\perp} \leftrightarrow Y^{\perp}$$

$$\forall prec : PRE\_COND[X];\ postc : POST\_COND[X,\ Y] \bullet$$
$$lift(prec,\ postc) =$$
$$\{\ xa : X^{\perp};\ ya : Y^{\perp}$$
$$\mid \quad xa = \perp$$
$$\vee \quad (\exists x : X \bullet \neg x \in prec \land xa = \iota\ x)$$
$$\vee \quad (\exists x : X;\ y : Y \bullet\ x \in prec \land (x,\ y) \in postc \land xa = \iota\ x \land ya = \iota\ y)\ \}$$

[3] In fact, we do not actually need to augment the input set X, an analogue of *lift* could be defined mapping specifications to relations between X and $Y^\perp$ and this would carry refinements to inclusions just like the formulation here. We have chosen to augment X for uniformity with Woodcock and Davies[3] and because this formulation seems likely to have nicer compositionality properties.

We can now claim the theorem that refinement as we have defined it is equivalent to inclusion of the lifted relations.

$$\mid \mathit{refinement\_lift\_cnj}\ ?\vdash \forall s_1,\ s_2 \colon \mathit{SPEC}\bullet\ s_1 \sqsubseteq s_2 \Leftrightarrow \mathit{lift}\ s_2 \subseteq \mathit{lift}\ s_1$$

Together with earlier results this shows that our refinement ordering is isomorphic to a complete sublattice of the lattice of relations on the sets of augmented inputs and outputs. Our final theorem characterises this sublattice.

$$\begin{array}{l}
\mid \mathit{ran_lift_cnj}\ ?\vdash \\
\mid \qquad \mathit{ran}\ \mathit{lift} = \\
\mid \qquad \{ \qquad r : (_^{\perp}) \leftrightarrow (_^{\perp}) \\
\mid \qquad \mid \qquad \forall x{:}(_^{\perp});\ y : (_^{\perp})\bullet\ (\perp,\ y) \in r \wedge ((x,\ \perp) \in r \Rightarrow (x,\ y) \in r)\}
\end{array}$$

References

[1] He Jifeng and C. A. R. Hoare. *Unifying theories of programming.* Springer-Verlag, 1988.

[2] R.B. Jones. Methods and Tools for the Verification of Critical Properties. In R.Shaw, editor, *5th Refinement Workshop*, Workshops in Computing, pages 88–118. Springer-Verlag/BCS-FACS, 1992.

[3] Jim Woodcock and Jim Davies. *Using Z: Specification, Refinement, and Proof.* Prentice/Hall International, 1996.

[4] LEMMA1/ZED/WRK070. *On Correctness of Imperative Programs — Precondition Calculation.* R.D. Arthan, Lemma 1 Ltd., `rda@lemma-one.com`.

A WEAKEST PRE-CONDITIONS FOR RELATIONS

In [4], we give an account of pre-condition calculation for a small programming language. In subsequent work, we plan to explore weakest pre-condition calculation in less familiar situations. It is therefore helpful to formalise the abstract concept. In this appendix we give an account of weakest pre-conditions arbitrary relations in Z.

What is done here is motivated by the notion of a traced monoidal category, but no knowledge of these is required. Readers familiar with the basic ideas of category theory will appreciate that the class of all relations in Z under relational composition (which we sometimes call "horizontal composition") form a category (but not one that we could formalise in Z). This category becomes a monoidal category under a "vertical" composition operator given by a relational product and then becomes a traced monoidal category under a trace operator which we will define by existential quantification. Our aim is to show how the weakest pre-condition behaves with respect to these operators.

The simplest notion of weakest pre-condition pulls a simple predicate back through a relation. Thinking of predicates as sets, this notion is defined as follows:

$[Y, Z]$

$$\boldsymbol{WPS} : (Y \leftrightarrow Z) \times \mathbb{P}Z \to \mathbb{P}Y$$

$$\forall S : Y \leftrightarrow Z;\ C : \mathbb{P}Z \bullet\ WPS(S,\ C) = \{y : Y \mid S\ (\!|\{y\}|\!) \subseteq C\}$$

The following theorem shows that our explicit definition for $WPS(S, C)$ does indeed give the largest set (i.e., weakest predicate), B, such that $S(\!|B|\!) \subseteq C$.

wps_correct_thm ?⊢
$$\forall S : \mathbb{U} \leftrightarrow \mathbb{U};\ C : \mathbb{P}\mathbb{U} \bullet\ S(\!|WPS(S,\ C)|\!) \subseteq C;$$
$$\forall S : \mathbb{U} \leftrightarrow \mathbb{U};\ B : \mathbb{P}\mathbb{U};\ C : \mathbb{U} \mid S(\!|B|\!) \subseteq C \bullet B \subseteq WPS(S,\ C)$$

(Note: $\mathbb{U}$ is defined in the ProofPower-Z library to be the generic object which gives the universe of all elements of a type. In a declaration, it gives a similar effect to a type variable in HOL or ML, as a set thought of as a predicate it corresponds to the predicate *true*, and as a relation it denotes the total relation or *chaos*.)

The following theorem shows that the relational semantics can be recovered from the predicate-transformer semantics given by weakest pre-conditions:

rel_image_wps_thm ?⊢
$$\forall S : \mathbb{U} \leftrightarrow \mathbb{U};\ A : \mathbb{P}\mathbb{U} \bullet\ S(\!|A|\!) = \bigcap\{C : \mathbb{U} \mid A \subseteq WPS(S,\ C)\}$$

Weakest pre-conditions through a given relation R commute with intersections (conjunctions):

wps_cap_thm ?⊢ $\forall R : \mathbb{U} \leftrightarrow \mathbb{U};\ B,\ C : \mathbb{P}\mathbb{U} \bullet$
$$WPS(R,\ B \cap C) = WPS(R,\ B) \cap WPS(R,\ C)$$

Weakest pre-conditions through a given relation R do not commute with unions (disjunctions) in general, but the following does hold:

$$\begin{array}{|l} wps\_cup\_thm\ ?\vdash \forall R : \mathbb{U} \leftrightarrow \mathbb{U};\ B,\ C : \mathbb{P}\mathbb{U}\bullet \\ \quad\quad WPS(R,\ B) \cup WPS(R,\ C) \subseteq WPS(R,\ B \cup C) \end{array}$$

The weakest pre-condition for a relational composition ("horizontal" composition) is given by an appropriate form of functional composition.

$$\begin{array}{|l} wps\_comp\_thm\ ?\vdash \forall R : \mathbb{U} \leftrightarrow \mathbb{U};\ S : \mathbb{U} \leftrightarrow \mathbb{U};\ A : \mathbb{P}\mathbb{U}\bullet \\ \quad\quad WPS(R \fatsemi S,\ A) = WPS(R,\ WPS(S,\ A)) \end{array}$$

The following theorem may be helpful in understanding some of the results which follow. It shows that any point not in the domain of the relation S satisfies any weakest pre-condition through S.

$$\begin{array}{|l} wps\_compl\_dom\_thm\ ?\vdash \forall S : \mathbb{U} \leftrightarrow \mathbb{U};\ C : \mathbb{P}\mathbb{U} \bullet \mathbb{U} \setminus dom\ S \subseteq WPS(S,\ C) \end{array}$$

(Here $\mathbb{U} \setminus X$ gives the complement of X).

We now define the "vertical" composition of two arbitrary relations.

$$\begin{array}{|l} [X, Y, V, W] \\ \hline - \oplus_{\boldsymbol{R}} - : (X \leftrightarrow Y) \times (V \leftrightarrow W) \to (X \times V \leftrightarrow Y \times W) \\ \hline \forall R : X \leftrightarrow Y;\ S : V \leftrightarrow W\bullet \\ \quad\quad R \oplus_R S = \\ \quad\quad \{x : X;\ y : Y;\ v : V;\ w : W \mid (x,\ y) \in R \wedge (v,\ w) \in S \bullet (\ (x,\ v), (y,\ w)\ )\} \\ \hline \end{array}$$

There does not seem to be a nice, symmetric, general formula for the weakest pre-condition through a vertical composition. We give an asymmetric formulae, which is based on the following lemma.

$$\begin{array}{|l} wps\_prod\_lemma\ ?\vdash \forall R{:}\mathbb{U} \leftrightarrow \mathbb{U};\ S : \mathbb{U} \leftrightarrow \mathbb{U};\ X{:}\ \mathbb{P}\mathbb{U};\ V{:}\ \mathbb{P}\mathbb{U};\ C : \mathbb{P}(\mathbb{U} \times \mathbb{U})\bullet \\ \quad\quad (R \oplus_R S)(\!| X \times V |\!) \subseteq C \Leftrightarrow R(\!| X |\!) \subseteq \{y : \mathbb{U} \mid \forall w{:}S(\!| V |\!)\bullet (y,\ w) \in C\} \end{array}$$

This gives the following general formula for the weakest pre-condition through a vertical product:

$$\begin{array}{|l} wps\_prod\_thm\ ?\vdash\ \forall R{:}\mathbb{U} \leftrightarrow \mathbb{U};\ S : \mathbb{U} \leftrightarrow \mathbb{U};\ C : \mathbb{P}(\mathbb{U} \times \mathbb{U})\bullet \\ \quad\quad WPS(R \oplus_R S,\ C) = \bigcup\{V : \mathbb{U} \bullet WPS(R,\ \{y : \mathbb{U} \mid \forall w{:}S(\!| V |\!)\bullet (y,\ w) \in C\}) \times V\} \end{array}$$

(Note: using *rel_image_wps_thm* the expression $S(\!| V |\!)$ above could be rewritten in terms of weakest pre-conditions. It is not clear whether this would confer any advantage — it would certainly make the expression more complicated.)

Interchanging R and S gives the alternative formulation:

$$wps\_prod\_thm1 \vdash\ \forall R{:}\mathbb{U} \leftrightarrow \mathbb{U};\ S : \mathbb{U} \leftrightarrow \mathbb{U};\ C : \mathbb{P}(\mathbb{U} \times \mathbb{U}) \bullet$$
$$WPS(R \oplus_R S,\ C) = \bigcup\{X : \mathbb{U} \bullet X \times WPS(S,\ \{w : \mathbb{U} \mid \forall y{:}R(\!|X|\!) \bullet (y,\ w) \in C\})\}$$

The special case of a weakest pre-condition through a vertical composite, where the post-condition is given as the product of two sets is much simpler:

$$wps\_simple\_prod\_thm \vdash \forall R : \mathbb{U} \leftrightarrow \mathbb{U};\ S : \mathbb{U} \leftrightarrow \mathbb{U};\ B : \mathbb{P}\mathbb{U};\ C : \mathbb{P}\mathbb{U} \bullet$$
$$WPS(R \oplus_R S,\ B \times C) =$$
$$(WPS(R,\ B) \times WPS(S,\ C)) \cup ((\mathbb{U} \times \mathbb{U}) \setminus (dom\ R \times dom\ S))$$

Now we define the existential trace operator. It can be thought of as finding a fixed point.

$[X, Y, V]$

$$\boldsymbol{Trc}_\exists : (X \times V \leftrightarrow Y \times V) \to (X \leftrightarrow Y)$$

$$\forall R : (X \times V \leftrightarrow Y \times V) \bullet$$
$$Trc_\exists\ R = \{x : X;\ y : Y \mid \exists v : V \bullet ((x,\ v),\ (y,\ v)) \in R\}$$

The following theorem gives a formula for the weakest pre-condition through an existential trace:

$$wps\_trc\_exists\_thm \vdash \forall R : \mathbb{U} \times \mathbb{U} \leftrightarrow \mathbb{U} \times \mathbb{U};\ \ C : \mathbb{P}\mathbb{U} \bullet$$
$$WPS(Trc_\exists\ R,\ C) = \{x{:}\ \mathbb{U} \mid \forall v{:}\mathbb{U} \bullet (x,\ v) \in WPS(R \cap (\mathbb{U} \oplus_R (id\ \_)),\ C \times \mathbb{U})\}$$

As noted in [4], a more general notion of weakest pre-condition calculation, which has useful compositionality properties, deals with relations. In general, if S and P are binary relations, we want to specify $WP(S, P)$ so that it is the weakest (i.e., largest) relation R such that $R \fatsemi S \subseteq P$. We use an explicit formula for WP (cf. the clause for atoms in the function *prec_calc* in [4]).

$[X, Y, Z]$

$$\boldsymbol{WP} : (Y \leftrightarrow Z) \times (X \leftrightarrow Z) \to (X \leftrightarrow Y)$$

$$\forall S : Y \leftrightarrow Z;\ P : X \leftrightarrow Z \bullet\ WP(S,\ P) = \{x : X;\ y : Y \mid S\ (\!|\{y\}|\!) \subseteq P(\!|\{x\}|\!)\}$$

The following theorem shows that our explicit definition for $WP(S, P)$ is indeed the weakests R such that $R \fatsemi S \subseteq P$.

$$wp\_correct\_thm \vdash$$
$$\forall S : \mathbb{U} \leftrightarrow \mathbb{U};\ P : \mathbb{U} \leftrightarrow \mathbb{U} \bullet\ WP(S,\ P) \fatsemi S \subseteq P;$$
$$\forall R : \mathbb{U} \leftrightarrow \mathbb{U};\ S : \mathbb{U} \leftrightarrow \mathbb{U};\ P : \mathbb{U} \leftrightarrow \mathbb{U} \mid R \fatsemi S \subseteq P \bullet R \subseteq WP(S,\ P)$$

The relational formulation of weakest pre-condition may be defined in terms of the set formulation:

wp_wps_thm $?\vdash \forall S : \mathbb{U} \leftrightarrow \mathbb{U};\ P : \mathbb{U} \leftrightarrow \mathbb{U} \bullet$
$\quad WP(S,\ P) = \{x : \mathbb{U};\ y : \mathbb{U} \mid y \in WPS(S,\ P(\!|\{x\}|\!))\}$

From the above, one can calculate relational weakest pre-conditions for intersections, compositions etc. using the corresponding theorems for predicate weakest pre-conditions.

In the following theorem, we think of a post-condition P specifying a system which will be implemented as the relational composition of two subsystems R and S. The theorem then says that for fixed S, the weakest pre-condition $WP(S, P)$ gives the largest R such that the overall system will satisfy its specification P. Cf. the definition of *sound_prec_calc* in [4].

$wp\_refines\_thm$ $?\vdash \forall R : \mathbb{U} \leftrightarrow \mathbb{U};\ S : \mathbb{U} \leftrightarrow \mathbb{U};\ P : \mathbb{U} \leftrightarrow \mathbb{U} \bullet$
$\quad R \subseteq WP(S,\ P) \Rightarrow (dom\ R,\ P) \sqsubseteq (dom\ R,\ R \fatsemi S)$

B THE Z THEORY refcalc

B.1 Parents

$cache'refinement$ $z\_library$

B.2 Global Variables

$\boldsymbol{PRED}[\boldsymbol{X}]$ $\mathbb{P}\ (\mathbb{P}\ X)$

$\boldsymbol{PRE\_COND}[\boldsymbol{X}]$ $\mathbb{P}\ (\mathbb{P}\ X)$

$\boldsymbol{POST\_COND}[\boldsymbol{X}, \boldsymbol{Y}]$
$\mathbb{P}\ (X \leftrightarrow Y)$

$\boldsymbol{SPEC}[\boldsymbol{X}, \boldsymbol{Y}]$ $\mathbb{P}\ X \leftrightarrow X \leftrightarrow Y$

$(\_ \sqsubseteq \_)[\boldsymbol{X}, \boldsymbol{Y}]$
$\mathbb{P}\ X \times (X \leftrightarrow Y) \leftrightarrow \mathbb{P}\ X \times (X \leftrightarrow Y)$

$(\_ \equiv \_)[\boldsymbol{X}, \boldsymbol{Y}]$
$\mathbb{P}\ X \times (X \leftrightarrow Y) \leftrightarrow \mathbb{P}\ X \times (X \leftrightarrow Y)$

$(\_\ ^{\$})[\boldsymbol{X}, \boldsymbol{Y}]$
$\mathbb{P}\ X \times (X \leftrightarrow Y) \leftrightarrow \mathbb{P}\ X \times (X \leftrightarrow Y)$

$\boldsymbol{REDUCED}[\boldsymbol{X}, \boldsymbol{Y}]$
$\mathbb{P}\ X \leftrightarrow X \leftrightarrow Y$

$\boldsymbol{TRUE}[\boldsymbol{X}]$ $\mathbb{P}\ X$

$\boldsymbol{FALSE}[\boldsymbol{X}]$ $\mathbb{P}\ X$

$\boldsymbol{Bottom}[\boldsymbol{X}, \boldsymbol{Y}]$ $\mathbb{P}\ X \times (X \leftrightarrow Y)$

$\boldsymbol{Top}[\boldsymbol{X}, \boldsymbol{Y}]$ $\mathbb{P}\ X \times (X \leftrightarrow Y)$

$\boldsymbol{Abort}[\boldsymbol{X}, \boldsymbol{Y}]$ $\mathbb{P}\ X \leftrightarrow \mathbb{P}\ X \times (X \leftrightarrow Y)$

$\boldsymbol{Chaos}[\boldsymbol{X}, \boldsymbol{Y}]$ $\mathbb{P}\ X \leftrightarrow \mathbb{P}\ X \times (X \leftrightarrow Y)$

$\boldsymbol{CHAOS}[\boldsymbol{X}, \boldsymbol{Y}]$ $\mathbb{P}\ X \leftrightarrow X \leftrightarrow Y$

$\boldsymbol{ABORT}[\boldsymbol{X}, \boldsymbol{Y}]$ $X \leftrightarrow Y$

$\boldsymbol{Meet}[\boldsymbol{X}, \boldsymbol{Y}]$ $(\mathbb{P}\ X \leftrightarrow X \leftrightarrow Y) \leftrightarrow \mathbb{P}\ X \times (X \leftrightarrow Y)$

$\boldsymbol{Forbidden}[\boldsymbol{X}, \boldsymbol{Y}]$
$\mathbb{P}\ X \times (X \leftrightarrow Y) \leftrightarrow X \leftrightarrow Y$

$\boldsymbol{Join}[\boldsymbol{X}, \boldsymbol{Y}]$ $(\mathbb{P}\ X \leftrightarrow X \leftrightarrow Y) \leftrightarrow \mathbb{P}\ X \times (X \leftrightarrow Y)$

$(\_ \sqcap \_)[\boldsymbol{X}, \boldsymbol{Y}]$
$(\mathbb{P}\ X \times (X \leftrightarrow Y)) \times (\mathbb{P}\ X \times (X \leftrightarrow Y)) \leftrightarrow \mathbb{P}\ X \times (X \leftrightarrow Y)$

$(\_ \sqcup \_)[\boldsymbol{X}, \boldsymbol{Y}]$
$(\mathbb{P}\ X \times (X \leftrightarrow Y)) \times (\mathbb{P}\ X \times (X \leftrightarrow Y)) \leftrightarrow \mathbb{P}\ X \times (X \leftrightarrow Y)$

$\boldsymbol{X}\ ^{\perp}$ $\mathbb{P}\ (\mathbb{P}\ X)$

$\boldsymbol{\iota}[\boldsymbol{X}]$ $X \leftrightarrow \mathbb{P}\ X$

$\perp[\boldsymbol{X}]$ $\mathbb{P}\ X$

$\boldsymbol{lift}[\boldsymbol{X}, \boldsymbol{Y}]$ $\mathbb{P}\ X \times (X \leftrightarrow Y) \leftrightarrow \mathbb{P}\ X \leftrightarrow \mathbb{P}\ Y$

$\boldsymbol{WPS}[\boldsymbol{Y}, \boldsymbol{Z}]$ $(Y \leftrightarrow Z) \times \mathbb{P}\ Z \leftrightarrow \mathbb{P}\ Y$

$(\_ \oplus_{\boldsymbol{R}} \_)[\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{V}, \boldsymbol{W}]$
$(X \leftrightarrow Y) \times (V \leftrightarrow W) \leftrightarrow X \times V \leftrightarrow Y \times W$

$\boldsymbol{Trc}_{\exists}[\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{V}]$
$(X \times V \leftrightarrow Y \times V) \leftrightarrow X \leftrightarrow Y$

$\boldsymbol{WP}[\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}]$ $(Y \leftrightarrow Z) \times (X \leftrightarrow Z) \leftrightarrow X \leftrightarrow Y$

B.3 Fixity

fun* 7 *rightassoc
$(\_\ ^{\$})$

fun* 30 *rightassoc
$(\_ \sqcup \_)$

fun* 40 *rightassoc
$(\_ \sqcap \_)\,(\_ \oplus_R \_)$

gen* 7 *rightassoc
$(\_\ ^{\perp})$

rel $(\_ \models \_)$ $(\_ \sqsubseteq \_)(\_ \equiv \_)$

B.4 Axioms

$\_ \sqsubseteq \_$ $\vdash [X, Y]((\_ \sqsubseteq \_)[X,\ Y] \in SPEC[X,\ Y] \leftrightarrow SPEC[X,\ Y]$
$\wedge\ (\forall\ prec_1,\ prec_2 : PRE\_COND[X];$
$postc_1,\ postc_2 : POST\_COND[X,\ Y]$
$\bullet\ ((prec_1,\ postc_1),\ (prec_2,\ postc_2))$
$\in (\_ \sqsubseteq \_)[X,\ Y]$
$\Leftrightarrow prec_1 \subseteq prec_2$
$\wedge\ prec_1 \lhd postc_2 \subseteq postc_1))$

$\_ \equiv \_$ $\vdash [X, Y]((\_ \equiv \_)[X,\ Y] \in SPEC[X,\ Y] \leftrightarrow SPEC[X,\ Y]$
$\wedge\ (\forall\ s_1,\ s_2 : SPEC[X,\ Y]$
$\bullet\ (s_1,\ s_2) \in (\_ \equiv \_)[X,\ Y]$
$\Leftrightarrow s_1 \sqsubseteq s_2 \wedge s_2 \sqsubseteq s_1))$

$\_\ ^{\$}$ $\vdash [X, Y]((\_\ ^{\$})[X,\ Y] \in SPEC[X,\ Y] \to SPEC[X,\ Y]$
$\wedge\ (\forall\ prec : PRE\_COND[X];\ postc : POST\_COND[X,\ Y]$
$\bullet\ (\_\ ^{\$})[X,\ Y]\ (prec,\ postc)$
$= (prec,\ prec \lhd postc)))$

REDUCED $\vdash [X, Y](REDUCED[X,\ Y] \in \mathbb{P}\ SPEC[X,\ Y]$
$\wedge\ REDUCED[X,\ Y] = \{s : SPEC \bullet s\ ^{\$}\})$

TRUE
FALSE $\vdash [X](\{TRUE[X],\ FALSE[X]\} \subseteq PRE\_COND[X]$
$\wedge\ TRUE[X] = X$
$\wedge\ FALSE[X] = \varnothing)$

Bottom
Top
Abort
Chaos

CHAOS

ABORT $\vdash [X, Y]((ABORT[X, Y] \in POST\_COND[X, Y]$
$\land\ CHAOS[X, Y] \in PRE\_COND[X] \to POST\_COND[X, Y]$
$\land\ \{Abort[X, Y], Chaos[X, Y]\}$
$\subseteq PRE\_COND[X] \to SPEC[X, Y]$
$\land\ \{Bottom[X, Y], Top[X, Y]\} \subseteq SPEC[X, Y])$
$\land\ ABORT[X, Y] = \varnothing$
$\land\ (\forall\ prec : PRE\_COND[X]$
$\bullet\ CHAOS[X, Y]\ prec = prec \times Y$
$\land\ Abort[X, Y]\ prec = (prec, ABORT[X, Y])$
$\land\ Chaos[X, Y]\ prec = (prec, CHAOS[X, Y]\ prec))$
$\land\ Bottom[X, Y] = Chaos[X, Y]\ \varnothing$
$\land\ Top[X, Y] = Abort[X, Y]\ TRUE)$

Meet $\vdash [X, Y](Meet[X, Y] \in \mathbb{P}\ SPEC[X, Y] \to SPEC[X, Y]$
$\land\ (\forall\ A : \mathbb{P}\ SPEC[X, Y]$
$\bullet\ Meet[X, Y]\ A$
$= (\bigcap \{s : A \bullet s.1\}, \bigcup \{s : A \bullet s.2\}))))$

Forbidden $\vdash [X, Y](Forbidden[X, Y] \in SPEC[X, Y] \to POST\_COND[X, Y]$
$\land\ (\forall\ s : SPEC[X, Y]$
$\bullet\ Forbidden[X, Y]\ s = CHAOS\ s.1 \setminus s.2))$

Join $\vdash [X, Y](Join[X, Y] \in \mathbb{P}\ SPEC[X, Y] \to SPEC[X, Y]$
$\land\ (\forall\ A : \mathbb{P}\ SPEC[X, Y]$
$\bullet\ Join[X, Y]\ A$
$= (\bigcup \{s : A \bullet s.1\},$
$\bigcup \{s : A \bullet (s\ ^{\$}).2\}$
$\setminus \bigcup \{s : A \bullet Forbidden\ s\})))$

$\_ \sqcap \_$

$\_ \sqcup \_$ $\vdash [X, Y](\{(\_ \sqcap \_)[X, Y], (\_ \sqcup \_)[X, Y]\}$
$\subseteq SPEC[X, Y] \times SPEC[X, Y] \to SPEC[X, Y]$
$\land\ (\forall\ s_1, s_2 : SPEC[X, Y]$
$\bullet\ (\_ \sqcap \_)[X, Y]\ (s_1, s_2)$
$= Meet\ \{s_1, s_2\}$
$\land\ (\_ \sqcup \_)[X, Y]\ (s_1, s_2)$
$= Join\ \{s_1, s_2\}))$

ι

$\bot$ $\vdash [X]((\bot[X] \in X\ ^{\bot}$
$\land\ \iota[X] \in X \to X\ ^{\bot})$
$\land\ \bot[X] = \varnothing$
$\land\ (\forall\ x : X \bullet \iota[X]\ x = \{x\}))$

lift $\vdash [X, Y](lift[X, Y] \in SPEC[X, Y] \to X\ ^{\bot} \leftrightarrow Y\ ^{\bot}$
$\land\ (\forall\ prec : PRE\_COND[X]; postc : POST\_COND[X, Y]$
$\bullet\ lift[X, Y]\ (prec, postc)$
$= \{xa : X\ ^{\bot}; ya : Y\ ^{\bot}$
$\mid xa = \bot$
$\lor\ (\exists\ x : X \bullet \neg\ x \in prec \land xa = \iota\ x)$

$$\vee\ (\exists\ x : X;\ y : Y \bullet x \in prec \wedge (x,\ y) \in postc \wedge xa = \iota\ x \wedge ya = \iota\ y)\}))$$

WPS $\vdash [Y, Z](WPS[Y,\ Z] \in (Y \leftrightarrow Z) \times \mathbb{P}\ Z \to \mathbb{P}\ Y \wedge (\forall\ S : Y \leftrightarrow Z;\ C : \mathbb{P}\ Z \bullet WPS[Y,\ Z]\ (S,\ C) = \{y : Y \mid S \ (\!|\ \{y\}\ |\!) \subseteq C\}))$

_ $\oplus_{R}$ _ $\vdash [X, Y, V, W]((\_\ \oplus_R\ \_)[X,\ Y,\ V,\ W] \in (X \leftrightarrow Y) \times (V \leftrightarrow W) \to X \times V \leftrightarrow Y \times W \wedge (\forall\ R : X \leftrightarrow Y;\ S : V \leftrightarrow W \bullet (\_\ \oplus_R\ \_)[X,\ Y,\ V,\ W]\ (R,\ S) = \{x : X;\ y : Y;\ v : V;\ w : W \mid (x,\ y) \in R \wedge (v,\ w) \in S \bullet ((x,\ v),\ (y,\ w))\}))$

***Trc*$_{\exists}$** $\vdash [X, Y, V](Trc_{\exists}[X,\ Y,\ V] \in (X \times V \leftrightarrow Y \times V) \to X \leftrightarrow Y \wedge (\forall\ R : X \times V \leftrightarrow Y \times V \bullet Trc_{\exists}[X,\ Y,\ V]\ R = \{x : X;\ y : Y \mid \exists\ v : V \bullet ((x,\ v),\ (y,\ v)) \in R\}))$

WP $\vdash [X, Y, Z](WP[X,\ Y,\ Z] \in (Y \leftrightarrow Z) \times (X \leftrightarrow Z) \to X \leftrightarrow Y \wedge (\forall\ S : Y \leftrightarrow Z;\ P : X \leftrightarrow Z \bullet WP[X,\ Y,\ Z]\ (S,\ P) = \{x : X;\ y : Y \mid S \ (\!|\ \{y\}\ |\!) \subseteq P \ (\!|\ \{x\}\ |\!)\}))$

B.5 Definitions

PRED $\vdash [X](PRED[X] = \mathbb{P}\ X)$

PRE_COND $\vdash [X](PRE\_COND[X] = PRED[X])$

POST_COND $\vdash [X, Y](POST\_COND[X,\ Y] = X \leftrightarrow Y)$

SPEC $\vdash [X, Y](SPEC[X,\ Y] = PRE\_COND[X] \times POST\_COND[X,\ Y])$

_ $^{\perp}$ $\vdash [X](X\ ^{\perp} = \{A : \mathbb{P}\ X \mid \forall\ x,\ y : A \bullet x = y\})$

B.6 Conjectures

refinement_pre_order_cnj
$\forall\ s_1,\ s_2,\ s_3 : SPEC \bullet s_1 \sqsubseteq s_1 \wedge (s_1 \sqsubseteq s_2 \wedge s_2 \sqsubseteq s_3 \Rightarrow s_1 \sqsubseteq s_3)$

reduce_reduce_cnj
$\forall\ s : SPEC \bullet (s\ ^{\$})\ ^{\$} = s\ ^{\$}$

equiv_cnj $\forall\ s_1,\ s_2 : SPEC \bullet s_1 \equiv s_2 \Leftrightarrow s_1\ ^{\$} = s_2\ ^{\$}$

refinement_reduce_cnj

$$\forall\ s_1,\ s_2 : SPEC \bullet s_1 \sqsubseteq s_2 \Leftrightarrow s_1\ ^{\$} \sqsubseteq s_2\ ^{\$}$$

refinement_antisym_cnj

$$\forall\ s_1,\ s_2 : REDUCED \bullet s_1 \sqsubseteq s_2 \wedge s_2 \sqsubseteq s_1 \Rightarrow s_1 = s_2$$

bottom_top_cnj

$$\forall\ s : SPEC \bullet Bottom \sqsubseteq s \wedge s \sqsubseteq Top$$

refinement_meet_cnj

$$(\forall\ A : \mathbb{P}\ SPEC \bullet \forall\ t : A \bullet Meet\ A \sqsubseteq t) \wedge (\forall\ A : \mathbb{P}\ SPEC;\ s : SPEC \bullet (\forall\ t : A \bullet s \sqsubseteq t) \Rightarrow s \sqsubseteq Meet\ A)$$

forbidden_def_cnj

$$\forall\ s : SPEC \bullet Forbidden\ s = \{x : \mathbb{U};\ y : \mathbb{U} \mid x \in s.1 \wedge \neg\ (x,\ y) \in s.2\}$$

forbidden_cnj

$$\forall\ s_1,\ s_2 : SPEC \bullet s_1 \sqsubseteq s_2 \Leftrightarrow s_1.1 \subseteq s_2.1 \wedge s_2.2 \cap Forbidden\ s_1 = \varnothing$$

join_def_cnj

$$\forall\ A : \mathbb{P}\ SPEC \bullet Join\ A = (\{s : A;\ x : \mathbb{U} \mid x \in s.1 \bullet x\},\ \{s : A;\ x : \mathbb{U};\ y : \mathbb{U} \mid x \in s.1 \wedge (x,\ y) \in s.2 \wedge (\forall\ t : A \bullet x \in t.1 \Rightarrow (x,\ y) \in t.2) \bullet (x,\ y)\})$$

forbidden_join_cnj

$$\forall\ A : \mathbb{P}\ SPEC \bullet Forbidden\ (Join\ A) = \bigcup \{s : A \bullet Forbidden\ s\}$$

refinement_join_cnj

$$(\forall\ A : \mathbb{P}\ SPEC \bullet \forall\ t : A \bullet t \sqsubseteq Join\ A) \wedge (\forall\ A : \mathbb{P}\ SPEC;\ s : SPEC \bullet (\forall\ t : A \bullet t \sqsubseteq s) \Rightarrow Join\ A \sqsubseteq s)$$

meet2_cnj

$$\forall\ s_1,\ s_2 : SPEC \bullet s_1 \sqcap s_2 = (s_1.1 \cap s_2.1,\ s_1.2 \cup s_2.2)$$

join2_cnj

$$\forall\ s_1,\ s_2 : SPEC \bullet s_1 \sqcup s_2 = (s_1.1 \cup s_2.1,\ (s_1.1 \cap s_1.1) \lhd (s_1.2 \cap s_2.2) \cup (s_1.1 \setminus s_2.1) \lhd s_1.2 \cup (s_2.1 \setminus s_1.1) \lhd s_2.2)$$

refinement_lift_cnj

$$\forall\ s_1,\ s_2 : SPEC \bullet s_1 \sqsubseteq s_2 \Leftrightarrow lift\ s_2 \subseteq lift\ s_1$$

ran_lift_cnj

$$ran\ lift = \{r : (\_\ ^{\perp}) \leftrightarrow (\_\ ^{\perp}) \mid \forall\ x : (\_\ ^{\perp});\ y : (\_\ ^{\perp}) \bullet (\perp,\ y) \in r \wedge ((x,\ \perp) \in r \Rightarrow (x,\ y) \in r)\}$$

wps_correct_thm

$$(\forall\ S : (\_ \leftrightarrow \_);\ C : \mathbb{P}\ \mathbb{U} \bullet S\ (\!|\ WPS\ (S,\ C)\ |\!) \subseteq C)$$
$$\land\ (\forall\ S : (\_ \leftrightarrow \_);\ B : \mathbb{P}\ \mathbb{U};\ C : \mathbb{U}$$
$$\mid S\ (\!|\ B\ |\!) \subseteq C$$
$$\bullet\ B \subseteq WPS\ (S,\ C))$$

rel_image_wps_thm

$$\forall\ S : (\_ \leftrightarrow \_);\ A : \mathbb{P}\ \mathbb{U}$$
$$\bullet\ S\ (\!|\ A\ |\!) = \bigcap \{C : \mathbb{U} \mid A \subseteq WPS\ (S,\ C)\}$$

wps_cap_thm

$$\forall\ R : (\_ \leftrightarrow \_);\ B,\ C : \mathbb{P}\ \mathbb{U}$$
$$\bullet\ WPS\ (R,\ B \cap C) = WPS\ (R,\ B) \cap WPS\ (R,\ C)$$

wps_cup_thm

$$\forall\ R : (\_ \leftrightarrow \_);\ B,\ C : \mathbb{P}\ \mathbb{U}$$
$$\bullet\ WPS\ (R,\ B) \cup WPS\ (R,\ C) \subseteq WPS\ (R,\ B \cup C)$$

wps_comp_thm

$$\forall\ R : (\_ \leftrightarrow \_);\ S : (\_ \leftrightarrow \_);\ A : \mathbb{P}\ \mathbb{U}$$
$$\bullet\ WPS\ (R \fatsemi S,\ A) = WPS\ (R,\ WPS\ (S,\ A))$$

wps_compl_dom_thm

$$\forall\ S : (\_ \leftrightarrow \_);\ C : \mathbb{P}\ \mathbb{U} \bullet \mathbb{U} \setminus dom\ S \subseteq WPS\ (S,\ C)$$

wps_prod_lemma

$$\forall\ R : (\_ \leftrightarrow \_);$$
$$S : (\_ \leftrightarrow \_);$$
$$X : \mathbb{P}\ \mathbb{U};$$
$$V : \mathbb{P}\ \mathbb{U};$$
$$C : \mathbb{P}\ (\mathbb{U} \times \mathbb{U})$$
$$\bullet\ (R \oplus_R S)\ (\!|\ X \times V\ |\!) \subseteq C$$
$$\Leftrightarrow R\ (\!|\ X\ |\!) \subseteq \{y : \mathbb{U} \mid \forall\ w : S\ (\!|\ V\ |\!) \bullet (y,\ w) \in C\}$$

wps_prod_thm

$$\forall\ R : (\_ \leftrightarrow \_);\ S : (\_ \leftrightarrow \_);\ C : \mathbb{P}\ (\mathbb{U} \times \mathbb{U})$$
$$\bullet\ WPS\ (R \oplus_R S,\ C)$$
$$= \bigcup$$
$$\{V : \mathbb{U}$$
$$\bullet\ WPS\ (R,\ \{y : \mathbb{U} \mid \forall\ w : S\ (\!|\ V\ |\!) \bullet (y,\ w) \in C\})$$
$$\times\ V\}$$

wps_prod_thm1

$$\forall\ R : (\_ \leftrightarrow \_);\ S : (\_ \leftrightarrow \_);\ C : \mathbb{P}\ (\mathbb{U} \times \mathbb{U})$$
$$\bullet\ WPS\ (R \oplus_R S,\ C)$$
$$= \bigcup$$
$$\{X : \mathbb{U}$$
$$\bullet\ X$$
$$\times\ WPS$$
$$(S,$$
$$\{w : \mathbb{U}$$
$$\mid \forall\ y : R\ (\!|\ X\ |\!) \bullet (y,\ w) \in C\})\}$$

wps_simple_prod_thm

$$\forall\ R : (\_ \leftrightarrow \_);\ S : (\_ \leftrightarrow \_);\ B : \mathbb{P}\ \mathbb{U};\ C : \mathbb{P}\ \mathbb{U}$$
$$\bullet\ WPS\ (R \oplus_R S,\ B \times C)$$
$$= (WPS\ (R,\ B) \times WPS\ (S,\ C))$$
$$\cup\ ((\mathbb{U} \times \mathbb{U}) \setminus (dom\ R \times dom\ S))$$

wps_trc_exists_thm

$$\forall\ R : \mathbb{U} \times \mathbb{U} \leftrightarrow \mathbb{U} \times \mathbb{U};\ C : \mathbb{P}\ \mathbb{U}$$
$$\bullet\ WPS\ (Trc_\exists\ R,\ C)$$
$$= \{x : \mathbb{U}$$
$$\mid \forall\ v : \mathbb{U}$$
$$\bullet\ (x,\ v) \in WPS\ (R \cap (\mathbb{U} \oplus_R (id\ \_)),\ C \times \mathbb{U})\}$$

wp_correct_thm

$$(\forall\ S : (\_ \leftrightarrow \_);\ P : (\_ \leftrightarrow \_) \bullet WP\ (S,\ P) \fatsemi S \subseteq P) \\ \land\ (\forall\ R : (\_ \leftrightarrow \_);\ S : (\_ \leftrightarrow \_);\ P : (\_ \leftrightarrow \_) \\ \mid R \fatsemi S \subseteq P \\ \bullet R \subseteq WP\ (S,\ P))$$

wp_wps_thm

$$\forall\ S : (\_ \leftrightarrow \_);\ P : (\_ \leftrightarrow \_) \\ \bullet WP\ (S,\ P) = \{x : \mathbb{U};\ y : \mathbb{U} \mid y \in WPS\ (S,\ P\ (\!|\ \{x\}\ |\!))\}$$

wp_refines_thm

$$\forall\ R : (\_ \leftrightarrow \_);\ S : (\_ \leftrightarrow \_);\ P : (\_ \leftrightarrow \_) \\ \bullet R \subseteq WP\ (S,\ P) \Rightarrow (dom\ R,\ P) \sqsubseteq (dom\ R,\ R \fatsemi S)$$

B.7 Theorems

refcalc_u_thm

$$\vdash PRED = \mathbb{U} \land PRE\_COND = \mathbb{U} \land POST\_COND = \mathbb{U} \land SPEC = \mathbb{U}$$

refines_def

$$\vdash \forall\ s_1,\ s_2 : \mathbb{U} \\ \bullet\ s_1 \sqsubseteq s_2 \Leftrightarrow s_1.1 \subseteq s_2.1 \land s_1.1 \lhd s_2.2 \subseteq s_1.2$$

reduce_def

$$\vdash \forall\ s : \mathbb{U} \bullet s\ ^{\$} = (s.1,\ s.1 \lhd s.2)$$

equiv_def

$$\vdash \forall\ s_1,\ s_2 : SPEC \bullet s_1 \equiv s_2 \Leftrightarrow s_1 \sqsubseteq s_2 \land s_2 \sqsubseteq s_1$$

true_def

$$\vdash TRUE = \mathbb{U} \land FALSE = \varnothing$$

abort_def

$$\vdash ABORT = \varnothing \\ \land\ (\forall\ prec : PRE\_COND \\ \bullet\ CHAOS\ prec = prec \times \mathbb{U} \\ \land\ Abort\ prec = (prec,\ ABORT) \\ \land\ Chaos\ prec = (prec,\ CHAOS\ prec)) \\ \land\ Bottom = Chaos\ \varnothing \\ \land\ Top = Abort\ TRUE$$

forbidden_def

$$\vdash \forall\ s : SPEC \bullet Forbidden\ s = CHAOS\ s.1 \setminus s.2$$

meet_def

$$\vdash \forall\ A : \mathbb{P}\ SPEC \\ \bullet\ Meet\ A = (\bigcap \{s : A \bullet s.1\},\ \bigcup \{s : A \bullet s.2\})$$

join_def

$$\vdash \forall\ A : \mathbb{P}\ SPEC \\ \bullet\ Join\ A \\ = (\bigcup \{s : A \bullet s.1\}, \\ \bigcup \{s : A \bullet (s\ ^{\$}).2\} \\ \setminus \bigcup \{s : A \bullet Forbidden\ s\})$$

reduced_def

$$\vdash REDUCED \in \mathbb{P}\ SPEC \land REDUCED = \{s : SPEC \bullet s\ ^{\$}\}$$

join2_def

meet2_def

$$\vdash \forall\ s_1,\ s_2 : SPEC \\ \bullet\ s_1 \sqcap s_2 = Meet\ \{s_1,\ s_2\} \\ \land\ s_1 \sqcup s_2 = Join\ \{s_1,\ s_2\}$$

reduce_reduce_thm

$$\vdash \forall\ s : SPEC \bullet (s\ ^{\$})\ ^{\$} = s\ ^{\$}$$

equiv_thm

$$\vdash \forall\ s_1,\ s_2 : SPEC \bullet s_1 \equiv s_2 \Leftrightarrow s_1\ ^{\$} = s_2\ ^{\$}$$

refinement_pre_order_thm

$$\vdash \forall\ s_1,\ s_2,\ s_3 : SPEC \\ \bullet\ s_1 \sqsubseteq s_1 \land (s_1 \sqsubseteq s_2 \land s_2 \sqsubseteq s_3 \Rightarrow s_1 \sqsubseteq s_3)$$

refinement_reduce_thm

$$\vdash \forall\ s_1,\ s_2 : SPEC \bullet s_1 \sqsubseteq s_2 \Leftrightarrow s_1\ ^{\$} \sqsubseteq s_2\ ^{\$}$$

refinement_antisym_thm

$$\vdash \forall\ s_1,\ s_2 : REDUCED \bullet s_1 \sqsubseteq s_2 \wedge s_2 \sqsubseteq s_1 \Rightarrow s_1 = s_2$$

bottom_top_thm

$$\vdash \forall\ s : SPEC \bullet Bottom \sqsubseteq s \wedge s \sqsubseteq Top$$

refinement_meet_thm

$$\vdash (\forall\ A : \mathbb{P}\ SPEC \bullet \forall\ t : A \bullet Meet\ A \sqsubseteq t) \wedge (\forall\ A : \mathbb{P}\ SPEC;\ s : SPEC \bullet (\forall\ t : A \bullet s \sqsubseteq t) \Rightarrow s \sqsubseteq Meet\ A)$$

forbidden_def1

$$\vdash \forall\ s : SPEC \bullet Forbidden\ s = \{x : \mathbb{U};\ y : \mathbb{U} \mid x \in s.1 \wedge \neg\ (x,\ y) \in s.2\}$$

forbidden_thm

$$\vdash \forall\ s_1,\ s_2 : SPEC \bullet s_1 \sqsubseteq s_2 \Leftrightarrow s_1.1 \subseteq s_2.1 \wedge s_2.2 \cap Forbidden\ s_1 = \varnothing$$

join_def_thm

$$\vdash \forall\ A : \mathbb{P}\ SPEC \bullet Join\ A = (\{s : A;\ x : \mathbb{U} \mid x \in s.1 \bullet x\},\ \{s : A;\ x : \mathbb{U};\ y : \mathbb{U} \mid x \in s.1 \wedge (x,\ y) \in s.2 \wedge (\forall\ t : A \bullet x \in t.1 \Rightarrow (x,\ y) \in t.2) \bullet (x,\ y)\})$$

forbidden_join_thm

$$\vdash \forall\ A : \mathbb{P}\ SPEC \bullet Forbidden\ (Join\ A) = \bigcup \{s : A \bullet Forbidden\ s\}$$

refinement_join_thm

$$\vdash (\forall\ A : \mathbb{P}\ SPEC \bullet \forall\ t : A \bullet t \sqsubseteq Join\ A) \wedge (\forall\ A : \mathbb{P}\ SPEC;\ s : SPEC \bullet (\forall\ t : A \bullet t \sqsubseteq s) \Rightarrow Join\ A \sqsubseteq s)$$

meet2_thm

$$\vdash \forall\ s_1,\ s_2 : SPEC \bullet s_1 \sqcap s_2 = (s_1.1 \cap s_2.1,\ s_1.2 \cup s_2.2)$$

join2_thm

$$\vdash \forall\ s_1,\ s_2 : SPEC \bullet s_1 \sqcup s_2 = (s_1.1 \cup s_2.1,\ (s_1.1 \cap s_1.1) \lhd (s_1.2 \cap s_2.2) \cup (s_1.1 \setminus s_2.1) \lhd s_1.2 \cup (s_2.1 \setminus s_1.1) \lhd s_2.2)$$

ι_clauses

$$\vdash (\forall\ x : \mathbb{U} \bullet \neg\ \iota\ x = \bot \wedge \neg\ \bot = \iota\ x) \wedge (\forall\ x,\ y : \mathbb{U} \bullet \iota\ x = \iota\ y \Leftrightarrow x = y)$$

ι_∈_clauses

$$\vdash (\forall\ x : \mathbb{U} \bullet \iota\ x \in (\_\ ^{\perp})) \wedge \bot \in (\_\ ^{\perp})$$

augmented_cases_thm

$$\vdash \forall\ t : X\ ^{\perp} \bullet t = \bot \vee (\exists\ x : X \bullet t = \iota\ x)$$

augmented_u_cases_thm

$$\vdash \forall\ t : (\_\ ^{\perp}) \bullet t = \bot \vee (\exists\ x : \mathbb{U} \bullet t = \iota\ x)$$

lift_u_def_thm

$$\vdash \forall\ s : SPEC$$

$$
\begin{array}{l}
\quad \bullet\ \mathit{lift}\ s \\
\quad\quad = \{xa : (\_\ ^{\perp}); ya : (\_\ ^{\perp}) \\
\quad\quad\quad \mid xa = \perp \\
\quad\quad\quad\quad \vee (\exists\ x : \mathbb{U} \bullet \neg\ x \in s.1 \wedge xa = \iota\ x) \\
\quad\quad\quad\quad \vee (\exists\ x : \mathbb{U}; y : \mathbb{U} \\
\quad\quad\quad\quad\quad \bullet\ x \in s.1 \\
\quad\quad\quad\quad\quad\quad \wedge (x, y) \in s.2 \\
\quad\quad\quad\quad\quad\quad \wedge\ xa = \iota\ x \\
\quad\quad\quad\quad\quad\quad \wedge\ ya = \iota\ y)\}
\end{array}
$$

refinement_lift_thm

$$\vdash \forall\ s_1, s_2 : SPEC \bullet s_1 \sqsubseteq s_2 \Leftrightarrow \mathit{lift}\ s_2 \subseteq \mathit{lift}\ s_1$$

ran_lift_thm

$$
\begin{array}{l}
\vdash \mathit{ran}\ \mathit{lift} \\
\quad = \{r : (\_\ ^{\perp}) \leftrightarrow (\_\ ^{\perp}) \\
\quad\quad \mid \forall\ x : (\_\ ^{\perp}); y : (\_\ ^{\perp}) \\
\quad\quad\quad \bullet\ (\perp, y) \in r \wedge ((x, \perp) \in r \Rightarrow (x, y) \in r)\}
\end{array}
$$

z_image_singleton_thm

$$\vdash \forall\ R : \mathbb{U}; x : \mathbb{U} \bullet R\ (\!|\ \{x\}\ |\!) = \{z : \mathbb{U} \mid (x, z) \in R\}$$

wps_rw_thm

$$
\begin{array}{l}
\vdash \forall\ S : \mathbb{U}; C : \mathbb{U} \\
\quad \bullet\ \mathit{WPS}\ (S, C) \\
\quad\quad = \{y : \mathbb{U} \\
\quad\quad\quad \mid \forall\ z : \mathbb{U} \bullet (y, z) \in S \Rightarrow z \in C\}
\end{array}
$$

wps_correct_thm

$$
\begin{array}{l}
\vdash (\forall\ S : (\_ \leftrightarrow \_); C : \mathbb{P}\ \mathbb{U} \bullet S\ (\!|\ \mathit{WPS}\ (S, C)\ |\!) \subseteq C) \\
\quad \wedge (\forall\ S : (\_ \leftrightarrow \_); B : \mathbb{P}\ \mathbb{U}; C : \mathbb{U} \\
\quad\quad \mid S\ (\!|\ B\ |\!) \subseteq C \\
\quad\quad \bullet\ B \subseteq \mathit{WPS}\ (S, C))
\end{array}
$$

rel_image_wps_thm

$$
\begin{array}{l}
\vdash \forall\ S : (\_ \leftrightarrow \_); A : \mathbb{P}\ \mathbb{U} \\
\quad \bullet\ S\ (\!|\ A\ |\!) = \bigcap \{C : \mathbb{U} \mid A \subseteq \mathit{WPS}\ (S, C)\}
\end{array}
$$

wps_cap_thm

$$\vdash \forall\ R : (\_ \leftrightarrow \_); B, C : \mathbb{P}\ \mathbb{U} \bullet \mathit{WPS}\ (R, B \cap C) = \mathit{WPS}\ (R, B) \cap \mathit{WPS}\ (R, C)$$

wps_cup_thm

$$\vdash \forall\ R : (\_ \leftrightarrow \_); B, C : \mathbb{P}\ \mathbb{U} \bullet \mathit{WPS}\ (R, B) \cup \mathit{WPS}\ (R, C) \subseteq \mathit{WPS}\ (R, B \cup C)$$

wps_comp_thm

$$\vdash \forall\ R : (\_ \leftrightarrow \_); S : (\_ \leftrightarrow \_); A : \mathbb{P}\ \mathbb{U} \bullet \mathit{WPS}\ (R \fatsemi S, A) = \mathit{WPS}\ (R, \mathit{WPS}\ (S, A))$$

wps_empty_thm

$$\vdash \forall\ C : \mathbb{U} \bullet \mathit{WPS}\ (\{\}, C) = \mathbb{U}$$

wps_compl_dom_thm

$$\vdash \forall\ S : (\_ \leftrightarrow \_); C : \mathbb{P}\ \mathbb{U} \bullet \mathbb{U} \setminus \mathit{dom}\ S \subseteq \mathit{WPS}\ (S, C)$$

rel_prod_rw_thm

$$
\begin{array}{l}
\vdash \forall\ R : \mathbb{U}; S : \mathbb{U}; x1 : \mathbb{U}; y1 : \mathbb{U}; v1 : \mathbb{U}; w1 : \mathbb{U} \\
\quad \bullet\ (\exists\ x : \mathbb{U}; y : \mathbb{U}; v : \mathbb{U}; w : \mathbb{U} \\
\quad\quad \mid (x, y) \in R \wedge (v, w) \in S \\
\quad\quad \bullet\ (x = x1 \wedge v = v1) \wedge y = y1 \wedge w = w1) \\
\quad\quad \Leftrightarrow (x1, y1) \in R \wedge (v1, w1) \in S
\end{array}
$$

dom_rel_prod_thm

$$\vdash \forall\ R : \mathbb{U}; S : \mathbb{U} \bullet \mathit{dom}\ (R \oplus_R S) = \mathit{dom}\ R \times \mathit{dom}\ S$$

rel_prod_×_thm

$$\vdash \forall\ R : (\_ \leftrightarrow \_); X : \mathbb{P}\ \mathbb{U}; V : \mathbb{P}\ \mathbb{U} \bullet (R \oplus_R S)\ (\!|\ X \times V\ |\!) = R\ (\!|\ X\ |\!) \times S\ (\!|\ V\ |\!)$$

wps_prod_lemma

$\vdash \forall\ R : (\_ \leftrightarrow \_);$
$S : (\_ \leftrightarrow \_);$
$X : \mathbb{P}\ \mathbb{U};$
$V : \mathbb{P}\ \mathbb{U};$
$C : \mathbb{P}\ (\mathbb{U} \times \mathbb{U})$
$\bullet\ (R \oplus_R S)\ (\!|\ X \times V\ |\!) \subseteq C$
$\Leftrightarrow R\ (\!|\ X\ |\!) \subseteq \{y : \mathbb{U} \mid \forall\ w : S\ (\!|\ V\ |\!) \bullet (y,\ w) \in C\}$

wps_⋃_×_thm $\vdash \forall\ R : (\_ \leftrightarrow \_);\ S : (\_ \leftrightarrow \_);\ C : \mathbb{P}\ \mathbb{U}$
$\bullet\ WPS\ (R \oplus_R S,\ C)$
$= \bigcup$
$\{X : \mathbb{P}\ \mathbb{U};\ V : \mathbb{P}\ \mathbb{U}$
$\mid (R \oplus_R S)\ (\!|\ X \times V\ |\!) \subseteq C$
$\bullet\ X \times V\}$

wps_prod_thm $\vdash \forall\ R : (\_ \leftrightarrow \_);\ S : (\_ \leftrightarrow \_);\ C : \mathbb{P}\ (\mathbb{U} \times \mathbb{U})$
$\bullet\ WPS\ (R \oplus_R S,\ C)$
$= \bigcup$
$\{V : \mathbb{U}$
$\bullet\ WPS$
$(R,$
$\{y : \mathbb{U}$
$\mid \forall\ w : S\ (\!|\ V\ |\!) \bullet (y,\ w) \in C\})$
$\times\ V\}$

wps_sym_thm $\vdash \forall\ R : (\_ \leftrightarrow \_);\ S : (\_ \leftrightarrow \_);\ C : \mathbb{P}\ \mathbb{U}$
$\bullet\ WPS\ (R \oplus_R S,\ C) = WPS\ (S \oplus_R R,\ C\ ^\sim)\ ^\sim$

⋃_rel_inv_thm
$\vdash \forall\ v : \mathbb{U} \bullet \bigcup v\ ^\sim = \bigcup \{a : v \bullet a\ ^\sim\}$

×_rel_inv_thm
$\vdash \forall\ a : \mathbb{U};\ b : \mathbb{U} \bullet (a \times b)\ ^\sim = b \times a$

wps_prod_thm1
$\vdash \forall\ R : (\_ \leftrightarrow \_);\ S : (\_ \leftrightarrow \_);\ C : \mathbb{P}\ (\mathbb{U} \times \mathbb{U})$
$\bullet\ WPS\ (R \oplus_R S,\ C)$
$= \bigcup$
$\{X : \mathbb{U}$
$\bullet\ X$
$\times\ WPS$
$(S,$
$\{w : \mathbb{U}$
$\mid \forall\ y : R\ (\!|\ X\ |\!) \bullet (y,\ w) \in C\})\}$

wps_simple_prod_thm
$\vdash \forall\ R : (\_ \leftrightarrow \_);\ S : (\_ \leftrightarrow \_);\ B : \mathbb{P}\ \mathbb{U};\ C : \mathbb{P}\ \mathbb{U}$
$\bullet\ WPS\ (R \oplus_R S,\ B \times C)$
$= (WPS\ (R,\ B) \times WPS\ (S,\ C))$
$\cup\ ((\mathbb{U} \times \mathbb{U}) \setminus (dom\ R \times dom\ S))$

wps_trc_exists_thm
$\vdash \forall\ R : \mathbb{U} \times \mathbb{U} \leftrightarrow \mathbb{U} \times \mathbb{U};\ C : \mathbb{P}\ \mathbb{U}$
$\bullet\ WPS\ (Trc_\exists\ R,\ C)$
$= \{x : \mathbb{U}$
$\mid \forall\ v : \mathbb{U}$
$\bullet\ (x,\ v) \in WPS\ (R \cap (\mathbb{U} \oplus_R (id\ \_)),\ C \times \mathbb{U})\}$

wp_rw_thm $\vdash \forall\ S : \mathbb{U};\ P : \mathbb{U}$
$\bullet\ WP\ (S,\ P)$

$$= \{x : \mathbb{U};\ y : \mathbb{U} \mid \forall\ z : \mathbb{U} \bullet (y,\ z) \in S \Rightarrow (x,\ z) \in P\}$$

wp_correct_thm

$$\vdash (\forall\ S : (\_ \leftrightarrow \_);\ P : (\_ \leftrightarrow \_) \bullet WP\ (S,\ P) \fatsemi S \subseteq P) \land (\forall\ R : (\_ \leftrightarrow \_);\ S : (\_ \leftrightarrow \_);\ P : (\_ \leftrightarrow \_) \mid R \fatsemi S \subseteq P \bullet R \subseteq WP\ (S,\ P))$$

wp_wps_thm

$$\vdash \forall\ S : (\_ \leftrightarrow \_);\ P : (\_ \leftrightarrow \_) \bullet WP\ (S,\ P) = \{x : \mathbb{U};\ y : \mathbb{U} \mid y \in WPS\ (S,\ P\ (\!|\ \{x\}\ |\!))\}$$

wp_refines_thm

$$\vdash \forall\ R : (\_ \leftrightarrow \_);\ S : (\_ \leftrightarrow \_);\ P : (\_ \leftrightarrow \_) \bullet R \subseteq WP\ (S,\ P) \Rightarrow (dom\ R,\ P) \sqsubseteq (dom\ R,\ R \fatsemi S)$$

C INDEX