Under consideration for publication in Formal Aspects of Computing

The Specification Logic νZ

Martin C. Henson, Moshe Deutsch and Besnik Kajtazi

Department of Computer Science, University of Essex, Wivenhoe Park, Colchester, Essex CO4 3SQ, UK
E-mail: {hensm, mdeuts, bkajta}@essex.ac.uk

Abstract. This paper introduces a wide-spectrum specification logic νZ. The minimal core logic is extended to a more expressive specification logic which includes a schema calculus similar (but not equivalent) to Z, some new additional schema operators and extensions to a programming and program development logic.

1. Introduction

In this paper we introduce a wide-spectrum logic νZ. This is a very small specification logic based on a total correctness relational semantics with refinement as its fundamental relation.
The language which underlies the logic is Z-like, that is to say, we have schemas and schema operators. A significant difference is that operation schemas have two predicates, so resemble more the specification statements of the Refinement Calculus (*e.g.* [6] and [28]) or designs of the UTP [24]. This is, in fact, a fairly trivial difference, and the language could easily be set up using single predicate schemas, if preferred. On the other hand, there are several significant differences between νZ and Z:

- Z is based on a *partial*-correctness semantics; νZ is based on a *total*-correctness semantics.
- Z permits refinement of over–specifications; νZ does not.
- Z schema operators are not monotonic; νZ schema operators are monotonic (anti-monotonic).
- Z is based on *equality*; νZ is based on *refinement*.
- Z is a *specification* language; νZ is *wide-spectrum*.
- Z is relatively inflexible; νZ is extensible.
- Z is a *language*; νZ is a *logic*.

νZ is very economical and expressive: the core language is very small but capable of further development by definition. After we introduce the theory itself, we go on in the sections that follow to introduce a more expressive specification language (specification of specification constructs in νZ) and then a programming language (specification of programming constructs in νZ). None of these constructs are fixed; it is possible to provide alternative specification infrastructure and indeed alternative programming languages. Because νZis a logic, the various definitions, for specification and programming, inherit this and so we induce an extended specification logic and a programming logic alongside the definitions. These combine to form a mathematical framework for the derivation of programs from specification.
In this paper we will concentrate entirely on the system itself, its mathematical basis, and on methodologies for extending the core framework with additional features for specification, for programming and for program development. In future publications we will explore more pragmatic issues, providing techniques and examples to demonstrate how to effectively specify, refine and implement systems within νZ.

Correspondence and offprint requests to: Martin C. Henson, Department of Computer Science, University of Essex, Wivenhoe Park, Colchester, Essex CO4 3SQ, UK.

2. Core νZ

νZ is interpreted within the logic $\mathcal{Z}_C^{\perp}$, the extension of $\mathcal{Z}_C$ introduced in [20] which includes $\perp$ elements in all types. We assume familiarity with this theory (and notational conventions); all this is also covered in [23].

2.1. Syntax of νZ

The syntax of the core νZ framework is minimal. The type of an *operation schema*, U, is $\mathbb{P}\, T$ (written $U^{\mathbb{P}\, T}$) where T is a schema type which has the form $V \curlyvee V'$. Generally we will, as is usual in Z, write ΔV for $V \curlyvee V'$. We will write $U(v)$ to indicate that variable v may appear free in the schema expression U.[1]

Definition 2.1.

$$
\begin{array}{llll}
U^{\mathbb{P}\, T} & ::= & & \\
& X^{\mathbb{P}\, T} & & \text{– schema variable} \\
& [\, T \mid P \mid Q\,] & & \text{– atomic specifications} \\
& \neg\, U^{\mathbb{P}\, T} & & \text{– negation} \\
& U_0^{\mathbb{P}\, T_0} \vee U_1^{\mathbb{P}\, T_1} & (T = T_0 \curlyvee T_1) & \text{– disjunction} \\
& \exists\, \mathtt{x}^{T_{\mathtt{x}}} \bullet U_0^{\mathbb{P}\, T_0} & (T = T_0 - T_{\mathtt{x}}) & \text{– existential hiding} \\
& \mu\, X^{\mathbb{P}\, T} \bullet U(X)^{\mathbb{P}\, T} & & \text{– recursive schemas}
\end{array}
$$

2.2. Semantics of νZ

We first need to define refinement. In this framework it is simply containment.

Definition 2.2.

$$U_0^{\mathbb{P}\, T} \sqsupseteq U_1^{\mathbb{P}\, T} =_{df} [\![U_0]\!] \subseteq_T [\![U_1]\!]$$

We also need to specify the universe of specification models for a given type. Part (ii) is based on [11] section 8.1.

Definition 2.3.

$$
\begin{array}{llll}
(i) & magic^{\mathbb{P}\, T} & =_{df} & [\, T \mid true \mid false\,] \\
(ii) & W_T & =_{df} & \{ [\![U^{\mathbb{P}\, T}]\!] \mid magic^{\mathbb{P}\, T} \sqsupseteq U \wedge U \,{}_{9}^{\circ}\, magic^{\mathbb{P}(\Delta\, T^{out})} \sqsupseteq U \}
\end{array}
$$

Now we have the semantics of specifications.

Definition 2.4. In what follows, $T^{\star} =_{df} V_{\perp} \star V'_{\perp}$. The types are omitted here, but are taken to be as specified in the syntax above.

$$
\begin{array}{lll}
[\![X]\!] & =_{df} & X \\
[\![[\, T \mid P \mid Q\,]]\!] & =_{df} & \{ z_0 \star z_1' \in T^{\star} \mid z_0.P \Rightarrow z_0.z_1'.Q \} \\
[\![\neg\, U]\!] & =_{df} & \{ z \in T^{\star} \mid z =_V \perp \vee z \notin U \} \\
[\![U_0 \vee U_1]\!] & =_{df} & \{ z \in T^{\star} \mid z \mathrel{\dot{\in}} [\![U_0]\!] \vee z \mathrel{\dot{\in}} [\![U_1]\!] \} \\
[\![\exists\, \mathtt{x} \bullet U_0]\!] & =_{df} & \{ z \in T^{\star} \mid \exists\, y \in T_0^{\star} \bullet y \in [\![U_0]\!] \wedge z = y \upharpoonright T \} \\
[\![\mu\, X \bullet U(X)]\!] & =_{df} & \bigsqcap \{ X \in W_T \mid [\![U(X)]\!] \sqsupseteq X \}
\end{array}
$$

In the case of recursion, the schema variable X must appear in a *positive* position in U. That is: this is monotone recursion. The notation $t.P$ indicates the usual distribution of the binding t through the proposition P so that its component observations $\mathtt{x}$ are replaced by $t.\mathtt{x}$. Note that $\perp .P = false$ for all P ($\perp$ satisfies nothing, in particular it is outside every precondition). Since types can be recovered from the alphabets of P and Q for atomic schemas, we can and will write $[\, P \mid Q\,]$ for $[\, T \mid P \mid Q\,]$ in the sequel (and suppress types) where possible.

---

[1] When the variable has the type $\mathbb{P}\, T$ and T is a schema type (that is: it is a variable over schemas) we shall write it in $\mathcal{Z}_C^{\perp}$, as we do in νZ, in upper-case.

2.3. Logic of νZ

The semantics induces a logic for the constructs, as follows. In this introductory paper we omit the proofs.

2.3.1. Refinement

The rules for operation refinement in νZ are as follows:

Proposition 2.1. Let z be a fresh variable.

$$\frac{z \in U_0 \vdash z \in U_1}{U_0 \sqsupseteq U_1}\ (\sqsupseteq^+) \qquad \frac{U_0 \sqsupseteq U_1 \quad t \in U_0}{t \in U_1}\ (\sqsupseteq^-)$$

□

2.3.2. Atomic Operation Schemas

The rules for atomic operation schema in νZ are as follows:

Proposition 2.2.

$$\frac{t_0.P \vdash t_0.t_1'.Q}{t_0 \star t_1' \in [\,P \mid Q\,]}\ (U^+) \qquad \frac{t_0 \star t_1' \in [\,P \mid Q\,] \quad t_0.P}{t_0.t_1'.Q}\ (U^-)$$

□

The following inequations are derivable:

Proposition 2.3. Weakening of preconditions and strengthening of postconditions (respectively):

$$\frac{t.P_1 \vdash t.P_0}{[\,P_0 \mid Q\,] \sqsupseteq [\,P_1 \mid Q\,]} \qquad \frac{t.Q_0 \vdash t.Q_1}{[\,P \mid Q_0\,] \sqsupseteq [\,P \mid Q_1\,]}$$

□

2.3.3. Negated Schemas

Note that negation in νZ is not the relational complement: it is well-known that the universe of total-correctness relations in this model is not closed under that operation (see *e.g.* [11]). An alternative characterisation of the semantics is available using a combination of relational complement, disjunction and magic.

Definition 2.5.

$$\neg\, U = \overline{U} \vee magic$$

In any event, the rules for negation are derivable:

Proposition 2.4.

$$\frac{t \notin U}{t \in \neg\, U}\ (U^+_{\neg_0}) \qquad \frac{t \mathrel{\dot\in} magic}{t \in \neg\, U}\ (U^+_{\neg_1}) \qquad \frac{t \in \neg\, U \quad t \notin U \vdash P \quad t \mathrel{\dot\in} magic \vdash P}{P}\ (U^-_{\neg})$$

□

Negated schemas are *anti-monotonic* with respect to the refinement relation:

Proposition 2.5.

$$\frac{U_1 \sqsupseteq U_0}{\neg\, U_0 \sqsupseteq \neg\, U_1}$$

□

The notion satisfies double negation and excluded middle.

Proposition 2.6.

$$\frac{t \in U}{t \in \neg\neg U} \qquad \frac{t \in \neg\neg U}{t \in U} \qquad \frac{}{t \in \neg U \vee U}$$

□

2.3.4. Disjunction Schemas

The rules for disjunction schemas in νZ are derivable, as follows:

Proposition 2.7. Let $i \in 2$.

$$\frac{t \mathrel{\dot\in} U_i}{t \in U_0 \vee U_1}\ (U^+_{\vee_i}) \qquad \frac{t \in U_0 \vee U_1 \quad t \mathrel{\dot\in} U_0 \vdash P \quad t \mathrel{\dot\in} U_1 \vdash P}{P}\ (U^-_{\vee})$$

□

Disjunction schemas are *monotonic* with respect to the refinement relation:

Proposition 2.8.

$$\frac{U_0 \sqsupseteq U_2 \quad U_1 \sqsupseteq U_3}{U_0 \vee U_1 \sqsupseteq U_2 \vee U_3}$$

□

The inequational refinement logic of disjunction schemas:

Proposition 2.9.

$$[\, P_0 \mid Q_0 \,] \vee [\, P_1 \mid Q_1 \,] \sqsupseteq [\, P_0 \wedge P_1 \mid Q_0 \vee Q_1 \,]$$

□

Proposition 2.10.

$$[\, P_0 \vee P_1 \mid Q_0 \wedge Q_1 \,] \sqsupseteq [\, P_0 \mid Q_0 \,] \vee [\, P_1 \mid Q_1 \,]$$

□

2.3.5. Existential Hiding Schemas

The rules for existential hiding schemas in νZ are derivable, as follows:

Proposition 2.11.

$$\frac{t \in U}{t \mathrel{\dot\in} \exists\, \mathtt{x} \bullet U}\ (U^+_{\exists}) \qquad \frac{t \in \exists\, \mathtt{x} \bullet U \quad t \star \langle\!| \mathtt{x} \Rrightarrow y |\!\rangle \in U \vdash P}{P}\ (U^-_{\exists})$$

□

The rule for existential hiding involves *binding extension* which is closely connected to binding substitution and to a lemma which will be required extensively in the proofs of the refinement inequations that follow. First we have the definition of substitution for a binding t_0.

Definition 2.6.

$$t_0[\mathtt{x}_0/t_1].\mathtt{x}_1 =_{df} \begin{cases} t_1 & \text{when} \quad \mathtt{x}_0 = \mathtt{x}_1 \\ t_0.\mathtt{x}_1 & \text{otherwise} \end{cases}$$

We employ the notation $b.P$ and $b.t$ (generalising binding selection) adapted from [33]. Suppose that $\{\mathtt{z}_0 \cdots \mathtt{z}_n\}$ is the alphabet set of t, then $t.P$ is $P[\mathtt{z}_0 \cdots \mathtt{z}_n/t.\mathtt{z}_0 \cdots t.\mathtt{z}_n]$.

Lemma 2.12.

$$t_0[\mathtt{x}/t_0.t_1].P = t_0.P[\mathtt{x}/t_1]$$

Proof By induction on the structure of propositions and terms.

□

In view of this, it is possible to express the existential elimination rule as:

$$\frac{t \in \exists\, \mathtt{x} \bullet U \qquad t[\mathtt{x}/y] \in U \vdash P}{P}\ (U^{-}_{\exists})$$

Existential hiding schemas are *monotonic* with respect to the refinement relation:

Proposition 2.13.

$$\frac{U_0 \sqsupseteq U_1}{\exists\, \mathtt{x} \bullet U_0 \sqsupseteq \exists\, \mathtt{x} \bullet U_1}$$

□

There are inequations for refinement involving existential hiding. First, when hiding a before observation:

Proposition 2.14.

$$\exists\, \mathtt{x} \bullet [\, P \mid Q \,] \sqsupseteq [\, \forall\, u \bullet P[\mathtt{x}/u] \mid \exists\, u \bullet Q[\mathtt{x}/u] \,]$$

Proof

$$\dfrac{\dfrac{\dfrac{}{z_0 \star z_1' \in \exists\, \mathtt{x} \bullet [\, P \mid Q \,]}\,(0) \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{z_0 \star \langle\!| \mathtt{x} \Rrightarrow y |\!\rangle \star z_1' \in [\, P \mid Q \,]}\,(2)}{z_0[\mathtt{x}/y] \star z_1' \in [\, P \mid Q \,]} \qquad \dfrac{\dfrac{\dfrac{}{\forall\, u \bullet z_0.P[\mathtt{x}/u]}\,(1)}{z_0.P[\mathtt{x}/y]}}{z_0[\mathtt{x}/y].P}}{z_0[\mathtt{x}/y].z_1'.Q}}{z_0.z_1'.Q[\mathtt{x}/y]}}{\exists\, u \bullet z_0.z_1'.Q[\mathtt{x}/u]}}{\exists\, u \bullet z_0.z_1'.Q[\mathtt{x}/u]}\,(2)}{\dfrac{z_0 \star z_1' \in [\, \forall\, u \bullet P[\mathtt{x}/u] \mid \exists\, u \bullet Q[\mathtt{x}/u] \,]}{\exists\, \mathtt{x} \bullet [\, P \mid Q \,] \sqsupseteq [\, \forall\, u \bullet P[\mathtt{x}/u] \mid \exists\, u \bullet Q[\mathtt{x}/u] \,]}\,(0)}\,(1)$$

□

Second, when hiding an after observation:

Proposition 2.15.

$$\exists\, \mathtt{x}' \bullet [\, P \mid Q \,] \sqsupseteq [\, P \mid \exists\, v \bullet Q[\mathtt{x}'/v] \,]$$

Proof

$$\dfrac{\dfrac{\dfrac{}{z_0 \star z_1' \in \exists\, \mathtt{x}' \bullet [\, P \mid Q \,]}\,(0) \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{z_0 \star z_1' \star \langle\!| \mathtt{x}' \Rrightarrow w |\!\rangle \in [\, P \mid Q \,]}\,(2)}{z_0 \star z_1'[\mathtt{x}'/w] \in [\, P \mid Q \,]} \qquad \dfrac{}{z_0.P}\,(1)}{z_0.z_1'[\mathtt{x}'/w].Q}}{z_0.z_1'.Q[\mathtt{x}'/w]}}{\exists\, v \bullet z_0.z_1'.Q[\mathtt{x}'/v]}}{\exists\, v \bullet z_0.z_1'.Q[\mathtt{x}'/v]}\,(2)}{\dfrac{z_0 \star z_1' \in [\, P \mid \exists\, v \bullet Q[\mathtt{x}'/v] \,]}{\exists\, \mathtt{x}' \bullet [\, P \mid Q \,] \sqsupseteq [\, P \mid \exists\, v \bullet Q[\mathtt{x}'/v] \,]}\,(0)}\,(1)$$

□

And now, in the other direction:

Proposition 2.16.

$$[\, \exists\, u \bullet P[\mathtt{x}/u] \mid \forall\, u \bullet Q[\mathtt{x}/u] \,] \sqsupseteq \exists\, \mathtt{x} \bullet [\, P \mid Q \,]$$

Proof

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\overline{z_0 \star z_1' \in [\, \exists\, u \bullet P[\mathtt{x}/u] \mid \forall\, u \bullet Q[\mathtt{x}/u]\,]}\;(0) \qquad \dfrac{\dfrac{\overline{z_0[\mathtt{x}/y].P}\;(1)}{z_0.P[\mathtt{x}/y]}}{\exists\, u \bullet z_0.P[\mathtt{x}/u]}}{\forall\, u \bullet z_0.z_1'.Q[\mathtt{x}/u]}}{z_0.z_1'.Q[\mathtt{x}/y]}}{z_0[\mathtt{x}/y].z_1'.Q}}{z_0[\mathtt{x}/y] \star z_1' \in [\, P \mid Q\,]}\;(1)}{z_0 \star \langle\!|\, \mathtt{x} \Rrightarrow y \,|\!\rangle \star z_1' \in [\, P \mid Q\,]}}{z_0 \star z_1' \in \exists\, \mathtt{x} \bullet [\, P \mid Q\,]}}{[\, \exists\, u \bullet P[\mathtt{x}/u] \mid \forall\, u \bullet Q[\mathtt{x}/u]\,] \sqsupseteq \exists\, \mathtt{x} \bullet [\, P \mid Q\,]}\;(0)$$

□

Finally:

Proposition 2.17.

$$[\, P \mid \forall\, v \bullet Q[\mathtt{x}'/v]\,] \sqsupseteq \exists\, \mathtt{x}' \bullet [\, P \mid Q\,]$$

Proof

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\overline{z_0 \star z_1' \in [\, P \mid \forall\, v \bullet Q[\mathtt{x}'/v]\,]}\;(0) \qquad \overline{z_0.P}\;(1)}{\forall\, v \bullet z_0.z_1'.Q[\mathtt{x}/v]}}{z_0.z_1'.Q[\mathtt{x}'/w]}}{z_0.z_1'[\mathtt{x}'/w].Q}}{z_0 \star z_1'[\mathtt{x}'/w] \in [\, P \mid Q\,]}\;(1)}{z_0 \star \langle\!|\, \mathtt{x}' \Rrightarrow w \,|\!\rangle \star z_1' \in [\, P \mid Q\,]}}{z_0 \star z_1' \in \exists\, \mathtt{x}' \bullet [\, P \mid Q\,]}}{[\, P \mid \forall\, v \bullet Q[\mathtt{x}'/v]\,] \sqsupseteq \exists\, \mathtt{x}' \bullet [\, P \mid Q\,]}\;(0)$$

□

2.3.6. Recursive Schemas

The rules for recursive schemas in νZ are derivable, as follows:

Proposition 2.18.

$$\dfrac{t \in U(\mu\, X \bullet U(X))}{t \in \mu\, X \bullet U(X)}\;(\mu^+) \qquad \dfrac{t \in \mu\, X \bullet U(X)}{t \in U(\mu\, X \bullet U(X))}\;(\mu^-)$$

□

3. Specifying a Specification Language in νZ

The principles on which νZ is based include *economy* (the core system begin so small) and *extensibility* (the ease with which the core system can be made more expressive). Since the core system is so inexpressive, a first ambition will be to provide additional infrastructure which provides for a considerably more expressive specification language. We cover some aspects of this in this section, beginning with extensions providing other standard schema operators.

Some of the operators which we consider here are familiar from Z (though, because the semantics is differnt, the logic of these operators departs from that in Z). In addition there will be variations on familiar operators, such as composition: in this section we provide a notion of composition which allows *arbitrary* schemas to be composed, even when those schemas do not match for type. Finally, we introduce a range of quite new operators, unfamiliar in

Z, which we will see have some use when we turn to the topic of programming languages and program development logics in later sections.

3.1. Conjunction Schemas

We can define schema conjunction in terms of disjunction and negation, using the usual de Morgan definitions. We omit the proofs, which are a little more involved than usual, due to the more complex notion of negation we are obliged to use.

Definition 3.1.

$$U_0 \wedge U_1 =_{df} \neg(\neg U_0 \vee \neg U_1)$$

The usual rules are derivable.

Proposition 3.1. Let $i \in 2$.

$$\frac{t \mathrel{\dot\in} U_0 \quad t \mathrel{\dot\in} U_1}{t \in U_0 \wedge U_1}\ (U^+_\wedge) \qquad \frac{t \in U_0 \wedge U_1}{t \mathrel{\dot\in} U_i}\ (U^-_{\wedge_i})$$

□

Conjunction schemas are *monotonic* with respect to the refinement relation:

Proposition 3.2.

$$\frac{U_0 \sqsupseteq U_2 \quad U_1 \sqsupseteq U_3}{U_0 \wedge U_1 \sqsupseteq U_2 \wedge U_3}$$

□

The inequational refinement logic of conjunction schemas:

Proposition 3.3.

$$[\, P_0 \mid Q_0 \,] \wedge [\, P_1 \mid Q_1 \,] \sqsupseteq [\, P_0 \wedge P_1 \mid Q_0 \wedge Q_1 \,]$$

□

Proposition 3.4.

$$[\, P_0 \mid Q_0 \,] \wedge [\, P_1 \mid Q_1 \,] \sqsupseteq [\, P_0 \vee P_1 \mid Q_0 \vee Q_1 \,]$$

□

Proposition 3.5.

$$[\, P_0 \vee P_1 \mid Q_0 \wedge Q_1 \,] \sqsupseteq [\, P_0 \mid Q_0 \,] \wedge [\, P_1 \mid Q_1 \,]$$

□

3.2. Implication Schemas

We can define schema implication in terms of disjunction and negation, using the usual de Morgan definitions.

Definition 3.2.

$$U_0 \Rightarrow U_1 =_{df} \neg U_0 \vee U_1$$

With the obvious rules derivable:

Proposition 3.6.

$$\frac{z \mathrel{\dot\in} U_0 \vdash z \mathrel{\dot\in} U_1}{z \in U_0 \Rightarrow U_1}\ (U^+_\Rightarrow) \qquad \frac{t \in U_0 \Rightarrow U_1 \quad t \mathrel{\dot\in} U_0}{t \mathrel{\dot\in} U_1}\ (U^-_\Rightarrow)$$

□

Schema implication is *monotonic* on the right, and *anti-monotonic* on the left with respect to the refinement relation:

Proposition 3.7.

$$\frac{U_2 \sqsupseteq U_0 \quad U_1 \sqsupseteq U_3}{U_0 \Rightarrow U_1 \sqsupseteq U_2 \Rightarrow U_3}$$

□

The inequational refinement logic of schema implication:

Proposition 3.8.

$$[\, P_0 \mid Q_0 \,] \Rightarrow [\, P_1 \mid Q_1 \,] \sqsupseteq [\, P_0 \wedge P_1 \mid Q_0 \Rightarrow Q_1 \,]$$

□

Proposition 3.9.

$$[\, P_0 \Rightarrow P_1 \mid Q_0 \wedge Q_1 \,] \sqsupseteq [\, P_0 \mid Q_0 \,] \Rightarrow [\, P_1 \mid Q_1 \,]$$

□

3.3. Universal Hiding Schemas

Universal hiding is defined in terms of existential hiding and negation, using the standard de Morgan definition. We provide the proofs in detail, in this section, for illustration.

Definition 3.3.

$$\forall\, \mathbf{x} \bullet U =_{df} \neg \exists\, \mathbf{x} \bullet \neg U$$

And then the usual introduction and elimination rules are derivable.

Proposition 3.10. Let z be a fresh variable. We assume that t has the form $t_0 \star t_1'$.

$$\frac{t \star \llparenthesis\, \mathbf{x} \Rrightarrow z \,\rrparenthesis \in U}{t \in \forall\, \mathbf{x} \bullet U}$$

Proof Consider the following derivation, which requires the *law of excluded middle*:

$$\dfrac{\dfrac{}{t_0 = \perp \vee t_0 \neq \perp} \qquad \dfrac{\dfrac{}{t_0 = \perp}\,(0)}{t \in \neg \exists\, \mathbf{x} \bullet \neg U} \qquad \begin{array}{c}\delta_0\\ \vdots\\ t \in \neg \exists\, \mathbf{x} \bullet \neg U\end{array}}{t \in \neg \exists\, \mathbf{x} \bullet \neg U}\,(0)$$

where δ_0 is:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{t \in \exists\, \mathbf{x} \bullet \neg U}\,(1) \qquad \begin{array}{c}\delta_1\\ \vdots\\ \mathit{false}\end{array}}{\mathit{false}}\,(2)}{t \notin \exists\, \mathbf{x} \bullet \neg U}\,(1)}{t \in \neg \exists\, \mathbf{x} \bullet \neg U}$$

and where δ_1 is:

$$\dfrac{\dfrac{}{t \star \llparenthesis\, \mathbf{x} \Rrightarrow z \,\rrparenthesis \in \neg U}\,(2) \qquad \dfrac{\dfrac{}{t \star \llparenthesis\, \mathbf{x} \Rrightarrow z \,\rrparenthesis \notin U}\,(3) \quad t \star \llparenthesis\, \mathbf{x} \Rrightarrow z \,\rrparenthesis \in U}{\mathit{false}} \qquad \dfrac{\dfrac{}{t_0 = \perp}\,(3) \quad \dfrac{}{t_0 \neq \perp}\,(0)}{\mathit{false}}}{\mathit{false}}\,(3)$$

□

Proposition 3.11. Let t have the form $t_0 \star t_1'$.

$$\frac{t \in \forall\, \mathbf{x} \bullet U \quad v \in T_{\mathbf{x}}}{t \star \llparenthesis\, \mathbf{x} \Rrightarrow v \,\rrparenthesis \in U}$$

Proof Consider the following derivation, which requires the *law of excluded middle*:

$$\dfrac{\overline{t_0 = \bot \lor t_0 \neq \bot} \qquad \begin{array}{c}\delta_0\\ \vdots\\ t \star \langle\!| \mathtt{x} \Rrightarrow v |\!\rangle \in U\end{array} \qquad \begin{array}{c}\delta_1\\ \vdots\\ t \star \langle\!| \mathtt{x} \Rrightarrow v |\!\rangle \in U\end{array}}{t \star \langle\!| \mathtt{x} \Rrightarrow v |\!\rangle \in U}\ (0)$$

where δ_0 is:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{t_0 \star \langle\!| \mathtt{x} \Rrightarrow v |\!\rangle \notin U}\ (1)}{t_0 \star \langle\!| \mathtt{x} \Rrightarrow v |\!\rangle \in \neg U}}{t_0 \star \langle\!| \mathtt{x} \Rrightarrow v |\!\rangle \in T^\star}}{t_0 \star \langle\!| \mathtt{x} \Rrightarrow v |\!\rangle \in T_{0_\bot}} \qquad \dfrac{\dfrac{}{t_0 = \bot}\ (0) \qquad \dfrac{\dfrac{\dfrac{}{t_0 \star \langle\!| \mathtt{x} \Rrightarrow v |\!\rangle \in T_0}\ (2)}{t_0 \mathrel{\dot\in} T_0}}{t_0 \neq \bot}}{false} \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{t_0 \star \langle\!| \mathtt{x} \Rrightarrow v |\!\rangle = \bot}\ (2)}{\langle\!| \mathtt{x} \Rrightarrow v |\!\rangle = \bot}}{v = \bot} \qquad \dfrac{v \in T_{\mathtt{x}}}{v \neq \bot}}{false}}{\dfrac{false}{t \star \langle\!| \mathtt{x} \Rrightarrow v |\!\rangle \in U}\ (1)}\ (2)$$

and δ_1 is:

$$\dfrac{t \in \neg \exists \mathtt{x} \bullet \neg U \qquad \dfrac{\dfrac{\dfrac{}{t \star \langle\!| \mathtt{x} \Rrightarrow v |\!\rangle \notin U}\ (4)}{t \star \langle\!| \mathtt{x} \Rrightarrow v |\!\rangle \in \neg U}}{t \in \exists \mathtt{x}^{T_{\mathtt{x}}} \bullet \neg U} \qquad \dfrac{}{t_0 \neq \bot}\ (0)}{\dfrac{false}{t \star \langle\!| \mathtt{x} \Rrightarrow v |\!\rangle \in U}\ (4)}$$

□

Universal hiding schemas are *monotonic* with respect to the refinement relation:

Proposition 3.12.

$$\dfrac{U_0 \sqsupseteq U_1}{\forall \mathtt{x} \bullet U_0 \sqsupseteq \forall \mathtt{x} \bullet U_1}$$

□

We have an inequational logic of refinement for universal hiding.
First, when hiding a before observation:

Proposition 3.13.

$$\forall \mathtt{x} \bullet [\, P \mid Q \,] \sqsupseteq [\, \exists u \bullet P[\mathtt{x}/u] \mid \exists u \bullet Q[\mathtt{x}/u] \,]$$

Proof

$$\dfrac{\dfrac{\dfrac{}{\exists u \bullet z_0.P[\mathtt{x}/u]}\ (1) \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{z_0 \star z_1' \in \forall u \bullet [\, P \mid Q \,]}\ (0)}{z_0[\mathtt{x}/y] \star z_1' \in [\, P \mid Q \,]} \qquad \dfrac{\dfrac{}{z_0.P[\mathtt{x}/y]}\ (2)}{z_0[\mathtt{x}/y].P}}{z_0[\mathtt{x}/y].z_1'.Q}}{z_0.z_1'.Q[\mathtt{x}/y]}}{\exists u \bullet z_0.z_1'.Q[\mathtt{x}/u]}}{\exists u \bullet z_0.z_1'.Q[\mathtt{x}/u]}\ (2)$$

$$\dfrac{\dfrac{\exists u \bullet z_0.z_1'.Q[\mathtt{x}/u]}{z_0 \star z_1' \in [\, \exists u \bullet P[\mathtt{x}/u] \mid \exists u \bullet Q[\mathtt{x}/u] \,]}\ (1)}{\forall \mathtt{x} \bullet [\, P \mid Q \,] \sqsupseteq [\, \exists u \bullet P[\mathtt{x}/u] \mid \exists u \bullet Q[\mathtt{x}/u] \,]}\ (0)$$

□

And:

Proposition 3.14.

$$\forall \mathtt{x} \bullet [\, P \mid Q \,] \sqsupseteq [\, \forall u \bullet P[\mathtt{x}/u] \mid \forall u \bullet Q[\mathtt{x}/u] \,]$$

Proof

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\overline{z_0 \star z_1' \in \forall\, \mathtt{x} \bullet [\, P \mid Q \,]}\;(0)}{z_0[\mathtt{x}/y] \star z_1' \in [\, P \mid Q \,]} \qquad \dfrac{\overline{\forall\, u \bullet z_0.P[\mathtt{x}/u]}\;(1)}{z_0.P[z/y]}}{z_0[\mathtt{x}/y].z_1'.Q}}{z_0.z_1'.Q[\mathtt{x}/y]}}{\forall\, u \bullet z_0.z_1'.Q[\mathtt{x}/u]}}{z_0 \star z_1' \in [\, \forall\, u \bullet P[\mathtt{x}/u] \mid \forall\, u \bullet Q[\mathtt{x}/u] \,]}\;(1)}{\forall\, \mathtt{x} \bullet [\, P \mid Q \,] \sqsupseteq [\, \forall\, u \bullet P[\mathtt{x}/u] \mid \forall\, u \bullet Q[\mathtt{x}/u] \,]}\;(0)
$$

□

Next, when hiding an after observation:

Proposition 3.15.

$$\forall\, \mathtt{x}' \bullet [\, P \mid Q \,] \sqsupseteq [\, P \mid \forall\, v \bullet Q[\mathtt{x}'/v] \,]$$

Proof

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\overline{z_0 \star z_1' \in \forall\, \mathtt{x}' \bullet [\, P \mid Q \,]}}{z_0 \star z_1'[\mathtt{x}'/v] \in [\, P \mid Q \,]} \qquad \overline{z_0.P}\;(1)}{z_0.z_1'[\mathtt{x}'/v].Q}}{z_0.z_1'.Q[\mathtt{x}'/v]}}{\forall\, v \bullet z_0.z_1'.Q[\mathtt{x}'/v]}}{z_0 \star z_1' \in [\, P \mid \forall\, v \bullet Q[\mathtt{x}'/v] \,]}\;(1)}{\forall\, \mathtt{x}' \bullet [\, P \mid Q \,] \sqsupseteq [\, P \mid \forall\, v \bullet Q[\mathtt{x}'/v] \,]}\;(0)
$$

□

In the other direction:

Proposition 3.16.

$$[\, \exists\, u \bullet P[\mathtt{x}/u] \mid \forall\, u \bullet Q[\mathtt{x}/u] \,] \sqsupseteq \forall\, \mathtt{x} \bullet [\, P \mid Q \,]$$

Proof

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\overline{z_0 \star z_1' \in [\, \exists\, u \bullet P[\mathtt{x}/u] \mid \forall\, u \bullet Q[\mathtt{x}/u] \,]}\;(0) \qquad \dfrac{\overline{z_0.P[\mathtt{x}/y]}\;(1)}{\exists\, u \bullet z_0.P[\mathtt{x}/u]}}{\forall\, u \bullet z_0.z_1'.Q[\mathtt{x}/u]}}{z_0.z_1'.Q[\mathtt{x}/y]}}{z_0[\mathtt{x}/y].z_1'.Q}}{z_0[\mathtt{x}/y] \star z_1' \in [\, P \mid Q \,]}\;(1)}{\dfrac{z_0 \star z_1' \in \forall\, \mathtt{x} \bullet [\, P \mid Q \,]}{[\, \exists\, u \bullet P[\mathtt{x}/u] \mid \forall\, u \bullet Q[\mathtt{x}/u] \,] \sqsupseteq \forall\, \mathtt{x} \bullet [\, P \mid Q \,]}\;(0)}
$$

□

And:

Proposition 3.17.

$$[\, P \mid \forall\, v \bullet Q[\mathtt{x}'/v] \,] \sqsupseteq \forall\, \mathtt{x}' \bullet [\, P \mid Q \,]$$

Proof

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{z_0 \star z_1' \in [\, P \mid \forall v \bullet Q[\mathtt{x}'/v] \,]}\,(0) \quad \dfrac{}{z_0.P}\,(1)}{\forall u \bullet z_0.z_1'.Q[\mathtt{x}'/u]}}{z_0.z_1'.Q[\mathtt{x}'/w]}}{z_0.z_1'[\mathtt{x}'/w].Q}}{z_0 \star z_1'[\mathtt{x}'/w] \in [\, P \mid Q \,]}\,(1)}{z_0 \star z_1' \in \forall \mathtt{x}' \bullet [\, P \mid Q \,]}}{[\, P \mid \forall v \bullet Q[\mathtt{x}'/v] \,] \sqsupseteq \forall \mathtt{x}' \bullet [\, P \mid Q \,]}\,(0)$$

□

3.4. Ξ Schemas

We have the usual idea of Ξ-schemas:

Definition 3.4.

$$\Xi T =_{df} [\Delta T \mid true \mid \theta T = \theta' T]$$

The rules are straightforward:

Proposition 3.18.

$$\dfrac{}{t \star t' \in \Xi T} \qquad \dfrac{t_0 \star t_1' \in \Xi T}{t_0 = t_1}$$

□

3.5. The Skip Extension

We use this to define the `skip`-extension of a schema:

Definition 3.5. When T_0 and T_1 are disjoint, we define:

$$U^{\mathbb{P}\, T_0} \diamond T_1 =_{df} U \wedge \Xi T_1$$

Naturally this is well-defined even when the types are not disjoint, but the purpose of this is, as described, to extend a schema with `skip` and the definition has pathological effects in other circumstances.
The rules are straightforward:

Proposition 3.19.

$$\dfrac{t_0 \star t_1' \,\dot{\in}\, U \quad t_0 =_T t_1}{t_0 \star t_1' \in U \diamond T}\,(U_\diamond^+) \qquad \dfrac{t \in U \diamond T}{t \,\dot{\in}\, U}\,(U_{\diamond_0}^-) \qquad \dfrac{t_0 \star t_1' \in U \diamond T}{t_0 =_T t_1}\,(U_{\diamond_1}^-)$$

□

The `skip`-extension is *monotonic* with respect to the refinement relation:

Proposition 3.20.

$$\dfrac{U_0 \sqsupseteq U_1}{U_0 \diamond T \sqsupseteq U_1 \diamond T}$$

□

3.6. Composition Schemas

In νZ we wish to compose *arbitrary* specifications; even when the types of the operations do not match. In this regard νZ differs from Z. For such compositions to make sense, it is necessary to match incompatible types and to ensure that operations do not arbitrarily adjust bindings in the process. The definition of schema composition in νZ is, therefore, a little more complex than in Z. Nevertheless, it is possible to specify composition in the core theory, using the `skip`-extension operator.

Definition 3.6. Let $T_L = T_1 - T_0$ with the form $\Delta T_L = T_L^{in} \curlyvee T_L^{out'}$ and let Let $T_R = T_0 - T_1$ with the form $\Delta T_R = T_R^{in} \curlyvee T_R^{out'}$. Let $\overline{\mathtt{t}}$ be a vector of fresh observations with the size of the alphabet of $T_0^{out'} \curlyvee T_L^{out'}$ (equivalently: $T_1^{in} \curlyvee T_R^{in}$).

$$U_0^{\mathbb{P}(T_0^{in} \curlyvee T_0^{out'})} \fatsemi U_1^{\mathbb{P}(T_1^{in} \curlyvee T_1^{out'})} =_{df} \exists \overline{\mathtt{t}} \bullet \begin{array}{l} (U_0 \diamond T_L)[\alpha(T_0^{out'} \curlyvee T_L^{out'})/\ \overline{\mathtt{t}}] \wedge \\ (U_1 \diamond T_R)[\alpha(T_1^{in} \curlyvee T_R^{in})/\ \overline{\mathtt{t}}] \end{array}$$

The following introduction and elimination rules are derivable for schema composition:

Proposition 3.21.

$$\frac{t_0 \star t_2' \mathrel{\dot\in} U_0 \quad t_0 =_{T_L} t_2 \quad t_2 \star t_1' \mathrel{\dot\in} U_1 \quad t_2 =_{T_R} t_1}{t_0 \star t_1' \in U_0 \fatsemi U_1}\ (U_{\fatsemi}^{+})$$

□

Proposition 3.22.

$$\frac{t_0 \star t_1' \in U_0 \fatsemi U_1 \quad t_0 \star t_2' \mathrel{\dot\in} U_0, t_0 =_{T_L} t_2, t_2 \star t_1' \mathrel{\dot\in} U_1, t_2 =_{T_R} t_1 \vdash P}{P}\ (U_{\fatsemi}^{-})$$

□

Composition schemas are *monotonic* with respect to the refinement relation:

Proposition 3.23.

$$\frac{U_0 \sqsupseteq U_2 \quad U_1 \sqsupseteq U_3}{U_0 \fatsemi U_1 \sqsupseteq U_2 \fatsemi U_3}$$

□

3.7. Restricted Chaos

This definition introduces a restricted form of $chaos$: outside P this schema blocks.

Definition 3.7.

$$chaos_P =_{df} [\, \neg P \mid false \,]$$

This leads to the following logical rules.

Proposition 3.24.

$$\frac{t_0.P}{t_0 \star t_1' \in chaos_P}\ (chaos_P^{+}) \qquad \frac{t_0 \star t_1' \in chaos_P \quad \neg t_0.P}{false}\ (chaos_P^{-})$$

□

3.8. Schema Specialisation

We use restricted chaos to introduce the specialisation of a schema at a particular observation (it blocks elsewhere).

Definition 3.8. Let E_T be the schema type corresponding to the observations contained in E. Let $\mathbb{P}\ T$ be the schema type of U, and let $\Delta[\mathtt{x}^{T_{\mathtt{x}}}] \preceq T$.

$$U[\mathtt{x} \Rrightarrow E] =_{df} chaos_{(\mathtt{x}=\mathtt{E})} \wedge U$$

This induces the following rules:

Proposition 3.25.

$$\frac{t \mathrel{\dot\in} U \quad t.\mathtt{x} = t.E}{t \in U[\mathtt{x}{\Rrightarrow}E]} \qquad \frac{t \in U[\mathtt{x}{\Rrightarrow}E]}{t \mathrel{\dot\in} U} \qquad \frac{t \in U[\mathtt{x}{\Rrightarrow}E]}{t.\mathtt{x} = t.E}$$

□

Specialisation schemas are *monotonic* with respect to the refinement relation:

Proposition 3.26.

$$\frac{U_0 \sqsupseteq U_1}{U_0[\mathtt{x}{\Rrightarrow}E] \sqsupseteq U_1[\mathtt{x}{\Rrightarrow}E]}$$

□

3.9. Strengthening Preconditions

This operator has the effect of (in general) strengthening the precondition of a schema U by stipulating an additional condition P.

Definition 3.9. Let T_P be the schema type corresponding to the observations contained in P. Let $\mathbb{P}\ T$ be the schema type of U, and let $T_P \preceq T$.

$$U \uparrow P =_{df} chaos_P \Rightarrow U$$

The operator is governed by induced logical rules.

Proposition 3.27.

$$\frac{t.P \vdash t \in U}{t \in U \uparrow P} \qquad \frac{t \in U \uparrow P \quad t.P}{t \in U}$$

□

Strengthening preconditions is *monotonic* with respect to the refinement relation:

Proposition 3.28.

$$\frac{U_0 \sqsupseteq U_1}{U_0 \uparrow P \sqsupseteq U_1 \uparrow P}$$

□

4. Specifying a Programming Language in νZ

It is central to the methodology of νZ that it smoothly integrates specification and programming, and that it is possible to develop programs from specifications. This is achieved by firstly *specifying* a programming language in νZ and then inducing a corresponding program logic: refinement then automatically permits development from specifications to programs. We will develop such a language incrementally in this section.

4.1. Skip

Definition 4.1. For any type T.

$$\mathtt{skip} =_{df} \Xi T$$

Rules for `skip`:

Proposition 4.1.

$$\frac{}{t \star t' \in \mathtt{skip}}\ (\mathtt{skip}^+) \qquad \frac{t_0 \star t_1' \in \mathtt{skip}}{t_0 = t_1}\ (\mathtt{skip}^-)$$

□

The inequational refinement logic of `skip`:

Proposition 4.2.

$$\frac{\theta T = \theta' T \vdash t.Q}{\mathtt{skip} \sqsupseteq [\ T \mid true \mid Q\ ]}$$

□

4.2. Assignment

Definition 4.2. Let $V = T_E - [\mathtt{x}^{T_\mathtt{x}}]$

$$\mathtt{x} := E =_{df} [\ true \mid \mathtt{x}' = E\ ] \wedge \Xi V$$

Rules for assignment:

Proposition 4.3.

$$\frac{}{t \star t'[\mathtt{x}'/t.E] \in \mathtt{x} := E}\ (:=^{+}) \qquad \frac{t_0 \star t_1' \in \mathtt{x} := E}{t_0[\mathtt{x}/t_0.E] = t_1}\ (:=^{-})$$

□

The inequational refinement logic for assignment:

Proposition 4.4. Let z be fresh.

$$\frac{z.z'[\mathtt{x}'/z.E].Q}{\mathtt{x} := E \sqsupseteq [\ true \mid Q\ ]}$$

□

4.3. Conditional

We define a new operator, a conditional schema, in terms of conjunction and strengthening of preconditions:

Definition 4.3. Let $\mathbb{P}\ T_0$ and $\mathbb{P}\ T_1$ be the schema types of U_0 and U_1 respectively. Let $T_D \preceq T_0 \curlywedge T_1$.

$$\mathtt{if}\ D\ \mathtt{then}\ U_0\ \mathtt{else}\ U_1 =_{df} U_0 \uparrow D \wedge U_1 \uparrow \neg D$$

Rules for the conditional:

Proposition 4.5.

$$\frac{t.D \vdash z \mathrel{\dot\in} U_0 \quad \neg t.D \vdash t \mathrel{\dot\in} U_1}{t \in \mathtt{if}\ D\ \mathtt{then}\ U_0\ \mathtt{else}\ U_1}\ (\mathtt{if}^{+})$$

$$\frac{t \in \mathtt{if}\ D\ \mathtt{then}\ U_0\ \mathtt{else}\ U_1 \quad t.D}{t \mathrel{\dot\in} U_0}\ (\mathtt{if}_0^{-}) \qquad \frac{t \in \mathtt{if}\ D\ \mathtt{then}\ U_0\ \mathtt{else}\ U_1 \quad t.(\neg D)}{t \mathrel{\dot\in} U_1}\ (\mathtt{if}_1^{-})$$

□

Equations and inequations:

Proposition 4.6.

$$\mathtt{if}\ true\ \mathtt{then}\ U_0\ \mathtt{else}\ U_1 \doteq U_0$$

Proof Follow from specialisations of the introduction rule and the first elimination rule:

$$\frac{z \mathrel{\dot\in} U_0 \quad \dfrac{\dfrac{}{false}\ (1)}{z \mathrel{\dot\in} U_1}}{z \in \mathtt{if}\ D\ \mathtt{then}\ U_0\ \mathtt{else}\ U_1}\ (1)$$

$$\frac{z \in \texttt{if}\ D\ \texttt{then}\ U_0\ \texttt{else}\ U_1}{z \mathrel{\dot\in} U_0}$$

□

Proposition 4.7.

$$\texttt{if}\ \mathit{false}\ \texttt{then}\ U_0\ \texttt{else}\ U_1 \doteq U_1$$

□

Proposition 4.8.

$$\texttt{if}\ D\ \texttt{then}\ [\, P \mid D \land Q \,]\ \texttt{else}\ [\, P \mid \neg D \land Q \,] \sqsupseteq [\, P \mid Q \,]$$

Proof In what follows we write ϕ for

$$z \in \texttt{if}\ D\ \texttt{then}\ [\, P \mid D \land Q \,]\ \texttt{else}\ [\, P \mid \neg D \land Q \,] \sqsupseteq [\, P \mid Q \,]$$

$$\dfrac{\dfrac{\dfrac{\overline{D \lor \neg D} \qquad \dfrac{\dfrac{\overline{z.P}\ (1) \qquad \dfrac{\phi \qquad \overline{z.D}\ (2)}{z \in [\, P \mid D \land Q \,]}}{z.(D \land Q)}}{z.Q} \qquad \dfrac{\dfrac{\overline{z.P}\ (1) \qquad \dfrac{\phi \qquad \overline{z.(\neg D)}\ (2)}{z \in [\, P \mid \neg D \land Q \,]}}{z.(\neg D \land Q)}}{z.Q}}{z.Q}\ (2)}{z \in [\, P \mid Q \,]}\ (1)$$

□

4.4. Cases

The previous section can easily be generalised to case commands. We define a new operator, a case schema, in terms of conjunction and strengthening of preconditions:

Definition 4.4. Let $T = \{\cdots c_i \cdots\}$.

$$\texttt{cases}\ E^T\ \texttt{in}\ c_0 : U_0^{\mathbb{P}\,T_0}\ \cdots\ c_n : U_n^{\mathbb{P}\,T_n}\ \texttt{endcases} =_{df} U_0 \uparrow E = c_0 \land \cdots \land U_n \uparrow E = c_n$$

Rules for the cases:

Proposition 4.9. Let $i \in n + 1$.

$$\frac{\cdots \quad t.(E = c_i) \vdash t \mathrel{\dot\in} U_i \quad \cdots}{t \in \texttt{cases}\ E\ \texttt{in}\ c_0 : U_0\ \cdots\ c_n : U_n\ \texttt{endcases}}\ (\texttt{cases}^+)$$

$$\frac{t \in \texttt{cases}\ E\ \texttt{in}\ c_0 : U_0\ \cdots\ c_n : U_n\ \texttt{endcases} \qquad t.(E = c_i)}{t \mathrel{\dot\in} U_i}\ (\texttt{cases}_i^-)$$

□

Inequation:

Proposition 4.10. Let $T = \{\cdots c_i \cdots\}$.

$$\texttt{cases}\ E^T\ \texttt{in}\ c_0 : [\, T \mid P \mid E = c_0 \land Q \,] \cdots\ c_n : [\, T \mid P \mid E = c_n \land Q \,]\ \texttt{endcases} \sqsupseteq [\, T \mid P \mid Q \,]$$

□

4.5. Scope

Definition 4.5.

$$\texttt{begin var x} : T_{\texttt{x}} ; U \texttt{ end} =_{df} \exists \texttt{x}, \texttt{x}' \bullet U$$

Proposition 4.11.

$$\frac{t \in U}{t \mathrel{\dot\in} \texttt{begin var x} : T_{\texttt{x}} ; U \texttt{ end}} \ (\texttt{begin}^{+})$$

$$\frac{t \in \texttt{begin var x} : T_{\texttt{x}} ; U \texttt{ end} \quad t \star \langle\!| \texttt{x} \Rrightarrow y_0, \texttt{x}' \Rrightarrow y_1 |\!\rangle \in U \vdash P}{P} \ (\texttt{begin}^{-})$$

□

We have refinement inequations for the block:

Proposition 4.12.

$$\texttt{begin var x};\ [\, P \mid Q \,] \texttt{end} \sqsupseteq [\, \forall u \bullet P[\texttt{x}/u] \mid \exists u, v \bullet Q[\texttt{x}, \texttt{x}'/u, v] \,]$$

Proof Follows from propositions 2.14 and 2.15.

□

Proposition 4.13.

$$[\, \exists u \bullet P[\texttt{x}/u] \mid \forall u, v \bullet Q[\texttt{x}, \texttt{x}/u, v]) \,] \sqsupseteq \texttt{begin var x};\ [\, P \mid Q \,] \texttt{end}$$

Proof Follows from propositions 2.16 and 2.17.

□

4.6. Procedure Call

This and the interpretation of procedures themselves are mutually dependent. Suppose that f is a procedure (we will see an example in the next section), then procedure call is trivially defined:

Definition 4.6.

$$f(E) =_{df} f[\texttt{x} \Rrightarrow E]$$

This leads to inference rules:

Proposition 4.14.

$$\frac{t \mathrel{\dot\in} f \quad t.\texttt{x} = t.E}{t \in f(E)} \qquad \frac{t \in f(E)}{t \mathrel{\dot\in} f} \qquad \frac{t \in f(E)}{t.\texttt{x} = t.E}$$

□

It is necessary to analyse this in advance of procedures themselves, as it is implicated in the definition, as we will now see.

4.7. Primitive Recursive Procedures Over Numbers

We define a new schema operator, primitive recursion over the natural numbers, in terms of conjunction, strengthening of preconditions, existential hiding, schema specialisation and recursive schemas.

Definition 4.7.

$$\begin{array}{l}\texttt{proc } f(\texttt{x}) \texttt{ cases x in } 0 : U_0;\ \texttt{m} + 1 : U_1(f(\texttt{m})) \texttt{ endcases} =_{df} \\ \ \mu X \bullet U_0 \uparrow \texttt{x} = 0 \wedge \exists \texttt{m} \bullet U_1(X[\texttt{x} \Rrightarrow \texttt{m}]) \uparrow \texttt{x} = \texttt{m} + 1\end{array}$$

The idea is that U_1 is a schema whose alphabet includes $\mathtt{m}$ and which contains a free schema variable X whose type is the type of the entire procedure.
And the rules.

Proposition 4.15. Introduction:

$$\frac{t.\mathtt{x} = 0 \vdash t \mathrel{\dot\in} U_0 \qquad t.\mathtt{x} = t.\mathtt{m} + 1 \vdash t \mathrel{\dot\in} U_1(f(\mathtt{m}))}{t \mathrel{\dot\in} f}$$

Proof

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}\overline{t.\mathtt{x} = 0}\ (1)\\ \vdots \\ t \mathrel{\dot\in} U_0\end{array}}{t \mathrel{\dot\in} U_0 \upharpoonright \mathtt{x} = 0}\ (1) \qquad \dfrac{\dfrac{\begin{array}{c}\overline{t.\mathtt{x} = t.\mathtt{m} + 1}\ (2)\\ \vdots \\ t \mathrel{\dot\in} U_1(f(\mathtt{m}))\end{array}}{t \mathrel{\dot\in} U_1(f(\mathtt{m})) \upharpoonright \mathtt{x} = \mathtt{m} + 1}\ (2)}{t \mathrel{\dot\in} \exists\, \mathtt{m} \bullet U_1(f(\mathtt{m})) \upharpoonright \mathtt{x} = \mathtt{m} + 1}}{t \mathrel{\dot\in} U_0 \upharpoonright \mathtt{x} = 0 \lor \exists\, \mathtt{m} \bullet U_1(f(\mathtt{m})) \upharpoonright \mathtt{x} = \mathtt{m} + 1}}{t \mathrel{\dot\in} f}\ (\mu^+)$$

□

Proposition 4.16. Elimination:

$$\frac{t \mathrel{\dot\in} f \quad t.\mathtt{x} = 0}{t \mathrel{\dot\in} U_0} \qquad \frac{t \mathrel{\dot\in} f \quad t.\mathtt{x} = \mathtt{m} + 1}{t \mathrel{\dot\in} U_1(f(\mathtt{m}))}$$

□

In what follows, we write $U[E]$ for $U[\mathtt{x} \Rrightarrow E]$, when $\mathtt{x}$ is understood.

Proposition 4.17. The following rule is derivable:

$$\frac{n \in \mathbb{N} \vdash f(n) \sqsupseteq U[n]}{f \sqsupseteq U}$$

Proof Consider the following derivation:

$$\dfrac{\dfrac{\dfrac{\dfrac{\overline{z \in f}\ (1) \qquad \overline{z.\mathtt{x} = z.\mathtt{x}}}{z \in f(z.\mathtt{x})} \qquad \begin{array}{c}\overline{z.\mathtt{x} \in \mathbb{N}}\\ \vdots \\ f(z.\mathtt{x}) \sqsupseteq U[z.\mathtt{x}]\end{array}}{z \in U[z.\mathtt{x}]}}{z \in U}}{f \sqsupseteq U}\ (1)$$

□

And now, the key rule for program development for recursive programming: the rule for recursive synthesis:

Proposition 4.18. The following rule is derivable:

$$\frac{U_0 \sqsupseteq U[0] \qquad f(\mathtt{m}) \sqsupseteq U[\mathtt{m}] \vdash U_1(f(\mathtt{m})) \sqsupseteq U[\mathtt{m} + 1]}{f \sqsupseteq U}$$

Proof Consider the following derivation:

$$\dfrac{\dfrac{\dfrac{U_0 \sqsupseteq U[0] \quad \dfrac{\dfrac{\dfrac{}{z \in f(0)}\,(2)}{z.\mathtt{x} = 0} \quad \dfrac{\dfrac{}{z \in f(0)}\,(2)}{z \in f}}{z \in U_0}}{z \in U[0]}}{f(0) \sqsupseteq U[0]}\,(2) \qquad \begin{array}{c}\delta\\ \vdots\\ f(\mathtt{m}+1) \sqsupseteq U[\mathtt{m}+1]\end{array}}{\dfrac{f(n) \sqsupseteq U[n]}{f \sqsupseteq U}\,(0)}\,(1)$$

where δ is:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{z \in f(\mathtt{m}+1)}\,(3)}{z.\mathtt{x} = \mathtt{m}+1} \quad \dfrac{\dfrac{}{z \in f(\mathtt{m}+1)}\,(3)}{z \in f}}{z \in U_1(f(\mathtt{m}))} \qquad \begin{array}{c}\dfrac{}{f(\mathtt{m}) \sqsupseteq U[\mathtt{m}]}\,(1)\\ \vdots\\ U_1(f(\mathtt{m})) \sqsupseteq U[\mathtt{m}+1]\end{array}}{z \in U[\mathtt{m}+1]}}{f(\mathtt{m}+1) \sqsupseteq U[\mathtt{m}+1]}\,(3)$$

□

4.8. Primitive Recursion Over Lists

The technique is easy to generalise. For example:

Definition 4.8.

$$\begin{array}{l}\mathtt{proc}\; f(\mathtt{x})\; \mathtt{cases}\; \mathtt{x}\; \mathtt{in}\; \mathtt{Nil}\; : U_0;\; \mathtt{Cons}\; \mathtt{m}_0\; \mathtt{m}_1\; : U_1(f(\mathtt{m}_1))\; \mathtt{endcases} =_{df} \\ \mu X \bullet U_0 \uparrow \mathtt{x} = \mathtt{Nil} \wedge \exists \mathtt{m}_0, \mathtt{m}_1 \bullet U_1(X[\mathtt{x} \Rrightarrow \mathtt{m}_1]) \uparrow \mathtt{x} = \mathtt{Cons}\; \mathtt{m}_0\; \mathtt{m}_1\end{array}$$

The rule for recursive synthesis over lists:

Proposition 4.19. The following rule is derivable:

$$\dfrac{U_0 \sqsupseteq U[\mathtt{Nil}] \quad f(\mathtt{m}_1) \sqsupseteq U[\mathtt{m}_1] \vdash U_1(f(\mathtt{m}_1)) \sqsupseteq U[\mathtt{Cons}\; \mathtt{m}_0\; \mathtt{m}_1]}{f \sqsupseteq U}$$

□

4.9. Primitive Recursion Over Trees

Similarly for trees:

Definition 4.9.

$$\begin{array}{l}\mathtt{proc}\; f(\mathtt{x})\; \mathtt{cases}\; \mathtt{x}\; \mathtt{in}\; \mathtt{Leaf}\; \mathtt{m}_0 : U_0;\; \mathtt{Node}\; \mathtt{m}_1\; \mathtt{m}_2 : U_1(f(\mathtt{m}_1), f(\mathtt{m}_2))\; \mathtt{endcases} =_{df} \\ \mu X \bullet \exists \mathtt{m}_0 \bullet U_0 \uparrow \mathtt{x} = \mathtt{Leaf}\; m_0 \wedge \exists \mathtt{m}_1, \mathtt{m}_2 \bullet U_1(X[\mathtt{x} \Rrightarrow \mathtt{m}_1], X[\mathtt{x} \Rrightarrow \mathtt{m}_2]) \uparrow \mathtt{x} = \mathtt{Node}\; \mathtt{m}_1\; \mathtt{m}_2\end{array}$$

The rule for recursive synthesis over trees:

Proposition 4.20. The following rule is derivable:

$$\dfrac{U_0 \sqsupseteq U[\mathtt{Leaf}\; \mathtt{m}_0] \quad f(\mathtt{m}_1) \sqsupseteq U[\mathtt{m}_1], f(\mathtt{m}_2) \sqsupseteq U[\mathtt{m}_2] \vdash U_1(f(\mathtt{m}_1, \mathtt{m}_2)) \sqsupseteq U[\mathtt{Node}\; \mathtt{m}_1\; \mathtt{m}_2]}{f \sqsupseteq U}$$

□

4.10. Primitive Recursion Over Arbitrary Free-Types

All these special cases can be generalised to syntax-directed *free types*.

Types of the form Υ are the names of the free types and are given by equations of the form:

$$\Upsilon ::= \cdots \mid c_i \langle\!\langle \cdots \Upsilon_{ij} \cdots \rangle\!\rangle \mid \cdots$$

The terms of free-type:

$$t^{\Upsilon} ::= c_i \cdots t^{\Upsilon_{ij}} \cdots$$

The logic of free types permits the introduction of values in the type, equality reasoning and finally, elimination (generally by induction).

Proposition 4.21.

$$\frac{\cdots z_{ij} \in \Upsilon_{ij} \cdots}{c_i \cdots z_{ij} \cdots \in \Upsilon}\ (\Upsilon^+) \qquad \frac{\cdots z_{ij} \in \Upsilon_{ij} \cdots \quad \cdots z_{kl} \in \Upsilon_{kl} \cdots}{c_i \cdots z_{ij} \cdots \neq c_k \cdots z_{kl} \cdots}\ (\Upsilon_{\neq})$$

$$\frac{c_i \cdots z_{ij} \cdots = c_i \cdots y_{ij} \cdots}{z_{ij} = y_{ij}}\ (\Upsilon_{=})$$

$$\frac{\cdots \quad \cdots z_{ij} \in \Upsilon_{ij} \cdots, \cdots P[z/y_k] \cdots \vdash P[z/c_i \cdots z_{ij} \cdots] \quad \cdots}{z \in \Upsilon \vdash P}\ (\Upsilon^-)$$

where the y_k are all those variables occurring in the z_{ij} with type Υ.
□

Given a general free type Υ, the corresponding recursive program scheme is:

Definition 4.10.

$$\texttt{proc}_{\Upsilon}\, f(x)\ \texttt{cases x in}\ \cdots\ H_i\ \cdots\ \texttt{endcases}$$

where the H_i are the component cases:

$$H_i =_{df} c_i \cdots \texttt{m}_i\ \cdots\ :\ U_i(\cdots f(\texttt{w}_k) \cdots)$$

where the $\texttt{w}_k$ are those observations among the $\texttt{m}_i$ with type Υ.

The semantics in the general case is given by:

Definition 4.11.

$$\texttt{proc}_{\Upsilon}\, f(x)\ \texttt{cases x in}\ \cdots\ H_i\ \cdots\ \texttt{endcases} =_{df} \mu X \bullet \cdots \wedge K_i(X) \wedge \cdots$$

where:

$$K_i(X) =_{df} \exists \cdots \texttt{m}_i \cdots \bullet\ U_i(\cdots X[\texttt{x} \Rrightarrow \texttt{w}_k] \cdots) \uparrow \texttt{x} = c_i \cdots \texttt{m}_i\ \cdots$$

4.11. Guarded Commands

In this section, we extend νZ with the notion of *guarded commands*. Our motivation lies in the investigation of *action systems* in formalisms such as the Refinement Calculus, the B-Method [1, 29] and Z [34, 31].
The formalism of *action systems* was developed by Back *et al.* [4, 5] (as an extension of Dijkstra's language of guarded commands [22]) within the Refinement Calculus. These concepts were adapted within the B-Method by, for example, Abrial [2], Butler *et al.* [9, 8] and Waldén *et al.* [32]. Similar work (mainly related to the specification of reactive systems) in Z was done by, for example, Josephs [25], Strulo [30] and Miarka *et al.* [27]. In all these frameworks, the main concern is the issue of accommodating *both* ***refusals*** and ***underspecification*** in the same account. In other words, guards and preconditions must be able to coexist in the same specification, so as to employ both the *chaotic* and the *abortive* paradigms for refinement simultaneously.[2]
We shall demonstrate that the approach we have taken in νZ (motivated by our investigation in [21] and [14, ch.6]),

[2] The chaotic and the abortive paradigms for refinement are sometimes also known as the *contractual* and *behavioural* approaches (respectively) [12, ch.2-3]. We have, in previous work, examined thoroughly the concepts of both operation-refinement and data-refinement in these two paradigms. See *e.g.* [20, 17, 16, 15] for the investigation in the chaotic paradigm and *e.g.* [18, 19][14, ch.5,9] for the investigation in the abortive paradigm.

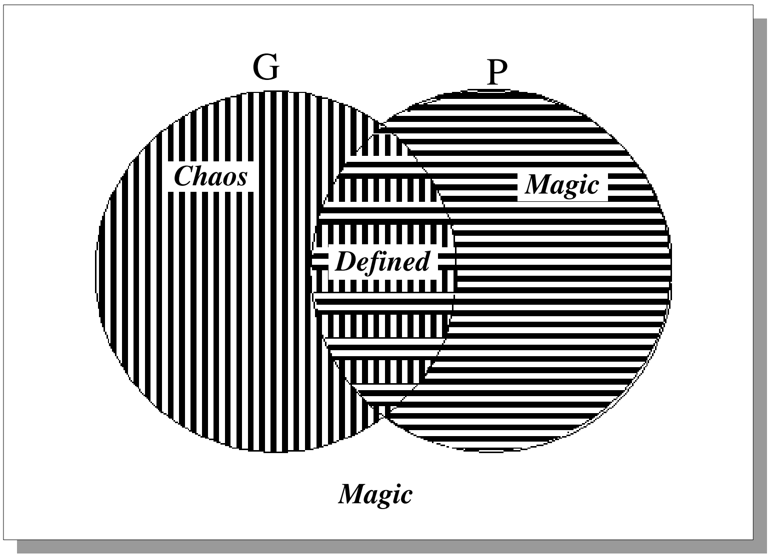

Fig.1. The possible regions of operation behaviour in our framework of guarded commands.

in which the refinement logic is logically prior to the schema logic, enables us to establish a logical framework for guarded commands which encompasses the strong characteristics of the above frameworks: mutual existence of both guards and preconditions in the same operation, accompanied by a powerful, and *fully-monotonic*, Z-like calculus of schema operations.

4.11.1. Logic and Semantics

The approach we take in establishing the logic of guarded commands in νZ is more liberal than the approach employed in [27]: firstly, we use classical logic, as opposed to the non-standard *three-valued* logic employed in *ibid.*; secondly, we do not insist on the guard necessarily being weaker than the precondition. Thus, the realistic description of the possible regions of the behaviour of a guarded operation is given in Fig. 1: The region in which both the guard and the precondition hold is *defined* by the operation; outside the guard, the operation behaves *magically* (regardless of whether or not its precondition holds); and when the guard holds but the precondition doesn't, the operation behaves *chaotically*. These concepts are captured by the following definition:

Definition 4.12.

$$G \longrightarrow [\, P \mid Q \,] =_{df} [\, \neg G \mid G \,] \wedge [\, P \mid Q \,]$$

Notice that when the guard is false the first component schema will always be *magic*, thus the whole schema expression becomes magic; whereas when the guard is true the first component schema will always be *chaos*, thus the conjunction with the actual operation denotes a selection of specified behaviours (which, of course, depends on its precondition and postcondition).
This leads directly to the following introduction and elimination rules (we consider the more general case allowing schema sets):

Proposition 4.22.

$$\frac{t.G \quad t \,\dot{\in}\, U}{t \in G \longrightarrow U}\ (\longrightarrow^{+}) \qquad \frac{t \in G \longrightarrow U}{t.G}\ (\longrightarrow^{-}_{0}) \qquad \frac{t \in G \longrightarrow U}{t \,\dot{\in}\, U}\ (\longrightarrow^{-}_{1})$$

□

Recasting these ideas within a single specification leads to a schema in which the guard implies the precondition and is conjoined with the postcondition. Thus, the following equation holds:

Proposition 4.23.

$$G \longrightarrow [\, P \mid Q \,] = [\, G \Rightarrow P \mid G \wedge Q \,]$$

Proof

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{z_0 \star z_1' \in G \longrightarrow [\,P \mid Q\,]}\,(1)}{z_0.G} \qquad \begin{array}{c}\delta\\ \vdots \\ z_0.z_1'.Q\end{array}}{z_0.z_1'.G \wedge Q}}{z_0 \star z_1' \in [\,G \Rightarrow P \mid G \wedge Q\,]}\,(2)}{G \longrightarrow [\,P \mid Q\,] \sqsupseteq [\,G \Rightarrow P \mid G \wedge Q\,]}\,(1)$$

Where δ stands for the following branch:

$$\dfrac{\dfrac{\dfrac{}{z_0 \star z_1' \in G \longrightarrow [\,P \mid Q\,]}\,(1)}{z_0 \star z_1' \mathrel{\dot\in} [\,P \mid Q\,]} \qquad \dfrac{\dfrac{}{z_0.G \Rightarrow P}\,(2) \quad \dfrac{\dfrac{}{z_0 \star z_1' \in G \longrightarrow [\,P \mid Q\,]}\,(1)}{z_0.G}}{z_0.P}}{z_0.z_1'.Q}$$

For the other direction, consider the following derivation which requires the *law of excluded middle*:

$$\dfrac{\dfrac{\dfrac{}{z_0.G \Rightarrow P \vee z_0.G \wedge \neg P}\,(LEM) \quad \begin{array}{c}\delta_0\\ \vdots \\ z_0 \star z_1' \in G \longrightarrow [\,P \mid Q\,]\end{array} \quad \begin{array}{c}\delta_1\\ \vdots \\ z_0 \star z_1' \in G \longrightarrow [\,P \mid Q\,]\end{array}}{z_0 \star z_1' \in G \longrightarrow [\,P \mid Q\,]}\,(2)}{[\,G \Rightarrow P \mid G \wedge Q\,] \sqsupseteq G \longrightarrow [\,P \mid Q\,]}\,(1)$$

Where δ_0 is:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{z_0 \star z_1' \in [\,G \Rightarrow P \mid G \wedge Q\,]}\,(1) \quad \dfrac{}{z_0.G \Rightarrow P}\,(2)}{z_0.z_1'.G \wedge Q}}{z_0.G} \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{z_0 \star z_1' \in [\,G \Rightarrow P \mid G \wedge Q\,]}\,(1) \quad \dfrac{}{z_0.G \Rightarrow P}\,(2)}{z_0.z_1'.G \wedge Q}}{z_0.z_1'.Q}}{z_0 \star z_1' \mathrel{\dot\in} [\,P \mid Q\,]}}{z_0 \star z_1' \in G \longrightarrow [\,P \mid Q\,]}$$

and δ_1 is:

$$\dfrac{\dfrac{\dfrac{}{z_0.G \wedge \neg P}\,(2)}{z_0.G} \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{z_0.P}\,(3) \quad \dfrac{\dfrac{}{z_0.G \wedge \neg P}\,(2)}{\neg\, z_0.P}}{false}}{z_0.z_1'.Q}}{z_0 \star z_1' \mathrel{\dot\in} [\,P \mid Q\,]}\,(3)}{z_0 \star z_1' \in G \longrightarrow [\,P \mid Q\,]}$$

□

4.11.2. Refinement Logic

In the approach developed in [27], an operation behaves chaotically (*i.e.* divergence including $\perp$) when its guard holds and its precondition doesn't hold, but it behaves *abortively* (*i.e.* strictly $\perp$) outside its guard. This gives rise to a notion of refinement in which not only preconditions may weaken and postconditions may strengthen, but also the guard may be strengthen. This, of course, is very intuitive because strengthening the guard merely means substituting undefined behaviour with abortive behaviour. However, in such an approach, the refinement rules must guarantee that "the precondition is the *upper bound* for strengthening the guard and the guard is the *lower bound* for weakening the precondition" [27]. This is in order to prevent abortive behaviour from substituting defined behaviour, on one hand, and chaotic behaviour from substituting abortive behaviour, on the other hand.

Conversely, in our framework the behaviour outside the guard is *magical* (as shown in Fig. 1). In which case, not only is it possible to strengthen the guard beyond the precondition in a refinement step (because the specification *magic* lies at the bottom of the refinement hierarchy in every framework which employs explicit preconditions and postconditions), but also it is possible to weaken the precondition beyond the guard (because, either way, any new behaviour that is outside the guard will be magical). Hence, we get the following basic refinement inequations for guarded commands.

Proposition 4.24. Weakening preconditions:

$$\frac{z.P_1 \vdash z.P_0}{G \longrightarrow [\, P_0 \mid Q \,] \sqsupseteq G \longrightarrow [\, P_1 \mid Q \,]}$$

Proof

$$\dfrac{\dfrac{\dfrac{\overline{z_0 \star z_1' \in G \longrightarrow [\, P_0 \mid Q \,]}\;(1)}{z_0.G} \qquad \dfrac{\dfrac{\dfrac{\overline{z_0 \star z_1' \in G \longrightarrow [\, P_0 \mid Q \,]}\;(1)}{z_0 \star z_1' \,\dot{\in}\, [\, P_0 \mid Q \,]} \qquad \begin{array}{c}\overline{z_0.P_1}\;(2)\\ \vdots \\ z_0.P_0\end{array}}{z_0.z_1'.Q}}{z_0 \star z_1' \,\dot{\in}\, [\, P_1 \mid Q \,]}\;(2)}{z_0 \star z_1' \in G \longrightarrow [\, P_1 \mid Q \,]}}{G \longrightarrow [\, P_0 \mid Q \,] \sqsupseteq G \longrightarrow [\, P_1 \mid Q \,]}\;(1)$$

□

Proposition 4.25. Strengthening postconditions:

$$\frac{z.Q_0 \vdash z.Q_1}{G \longrightarrow [\, P \mid Q_0 \,] \sqsupseteq G \longrightarrow [\, P \mid Q_1 \,]}$$

Proof

$$\dfrac{\dfrac{\dfrac{\overline{z_0 \star z_1' \in G \longrightarrow [\, P \mid Q_0 \,]}\;(1)}{z_0.G} \qquad \dfrac{\begin{array}{c}\dfrac{\dfrac{\overline{z_0 \star z_1' \in G \longrightarrow [\, P \mid Q_0 \,]}\;(1)}{z_0 \star z_1' \,\dot{\in}\, [\, P \mid Q_0 \,]} \qquad \overline{z_0.P}\;(2)}{z_0.z_1'.Q_0}\\ \vdots \\ z_0.z_1'.Q_1\end{array}}{z_0 \star z_1' \,\dot{\in}\, [\, P \mid Q_1 \,]}\;(2)}{z_0 \star z_1' \in G \longrightarrow [\, P \mid Q_1 \,]}}{G \longrightarrow [\, P \mid Q_0 \,] \sqsupseteq G \longrightarrow [\, P \mid Q_1 \,]}\;(1)$$

□

Proposition 4.26. Strengthening the guard:

$$\frac{z.G_0 \vdash z.G_1}{G_0 \longrightarrow [\, P \mid Q \,] \sqsupseteq G_1 \longrightarrow [\, P \mid Q \,]}$$

Proof

$$\dfrac{\dfrac{\begin{array}{c}\dfrac{\overline{z_0 \star z_1' \in G_0 \longrightarrow [\, P \mid Q \,]}\;(1)}{z_0.G_0}\\ \vdots \\ z_0.G_1\end{array} \qquad \dfrac{\overline{z_0 \star z_1' \in G_0 \longrightarrow [\, P \mid Q \,]}\;(1)}{z_0 \star z_1' \,\dot{\in}\, [\, P \mid Q \,]}}{z_0 \star z_1' \in G_1 \longrightarrow [\, P \mid Q \,]}}{G_0 \longrightarrow [\, P \mid Q \,] \sqsupseteq G_1 \longrightarrow [\, P \mid Q \,]}\;(1)$$

□

Given the nice properties of guarded commands in our framework (see Fig. 1), it is interesting to note that *any* operation is equivalent to a disjunction of its guarded commands, formed with converse guards; this result is true for *any* guard:

Proposition 4.27.

$$\forall\, G \bullet U = G \longrightarrow U \vee \neg G \longrightarrow U$$

Proof We prove this using refinement:

$$\dfrac{\dfrac{\overline{z \in G \longrightarrow U \vee \neg G \longrightarrow U}\ (1) \qquad \dfrac{\overline{z \in G \longrightarrow U}\ (2)}{z \mathrel{\dot\in} U} \qquad \dfrac{\overline{z \in \neg G \longrightarrow U}\ (2)}{z \mathrel{\dot\in} U}}{z \mathrel{\dot\in} U}\ (2)}{G \longrightarrow U \vee \neg G \longrightarrow U \sqsupseteq U}\ (1)$$

For the other direction, consider the following derivation which requires the *law of excluded middle*:

$$\dfrac{\dfrac{\overline{z.G \vee \neg\, z.G}\ (LEM) \qquad \dfrac{\dfrac{\overline{z.G}\ (2) \quad \overline{z \mathrel{\dot\in} U}\ (1)}{z \in G \longrightarrow U}}{z \in G \longrightarrow U \vee \neg G \longrightarrow U} \qquad \dfrac{\dfrac{\overline{\neg\, z.G}\ (2) \quad \overline{z \mathrel{\dot\in} U}\ (1)}{z \in \neg G \longrightarrow U}}{z \in G \longrightarrow U \vee \neg G \longrightarrow U}}{z \in G \longrightarrow U \vee \neg G \longrightarrow U}\ (2)}{U \sqsupseteq G \longrightarrow U \vee \neg G \longrightarrow U}\ (1)$$

□

4.11.3. Guarded Conditional

We define a *guarded conditional* operator, in terms of disjunction of two schemas guarded by converse guards:

Definition 4.13.

$$\texttt{gif}\ D\ \texttt{then}\ U_0^{\mathbb{P}\,T_0}\ \texttt{else}\ U_1^{\mathbb{P}\,T_1} =_{df} D \longrightarrow U_0 \vee \neg D \longrightarrow U_1$$

The following introduction and elimination rules are immediately derivable for the guarded conditional:

Proposition 4.28.

$$\dfrac{t.D \quad t \mathrel{\dot\in} U_0}{t \in \texttt{gif}\ D\ \texttt{then}\ U_0\ \texttt{else}\ U_1}\ (\texttt{gif}_0^+) \qquad \dfrac{\neg t.D \quad t \mathrel{\dot\in} U_1}{t \in \texttt{gif}\ D\ \texttt{then}\ U_0\ \texttt{else}\ U_1}\ (\texttt{gif}_1^+)$$

$$\dfrac{t \in \texttt{gif}\ D\ \texttt{then}\ U_0\ \texttt{else}\ U_1 \quad t \in D \longrightarrow U_0 \vdash P \quad t \in \neg D \longrightarrow U_1 \vdash P}{P}\ (\texttt{gif}^-)$$

□

Using our usual strategy involving elimination rules, we now demonstrate that the above theory is *equivalent* to the conditional theory we established in section 4.3. We begin by showing that every guarded conditional is a valid conditional in the "`if`" theory.

Proposition 4.29. The following rules are derivable:

$$\dfrac{t \in \texttt{gif}\ D\ \texttt{then}\ U_0\ \texttt{else}\ U_1 \quad t.D}{t \mathrel{\dot\in} U_0}\ (i) \qquad \dfrac{t \in \texttt{gif}\ D\ \texttt{then}\ U_0\ \texttt{else}\ U_1 \quad \neg t.D}{t \mathrel{\dot\in} U_1}\ (ii)$$

Proof For (i), consider the following derivation:

$$\dfrac{t \in \texttt{gif}\ D\ \texttt{then}\ U_0\ \texttt{else}\ U_1 \qquad \dfrac{\overline{t \in D \longrightarrow U_0}\ (1)}{t \mathrel{\dot\in} U_0} \qquad \dfrac{\dfrac{\dfrac{\overline{t \in \neg D \longrightarrow U_1}\ (1)}{\neg t.D} \qquad t.D}{\mathit{false}}}{t \mathrel{\dot\in} U_0}}{t \mathrel{\dot\in} U_0}\ (1)$$

For (*ii*), consider the following derivation:

$$\dfrac{t \in \mathtt{gif}\ D\ \mathtt{then}\ U_0\ \mathtt{else}\ U_1 \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{t \in D \longrightarrow U_0}\,(1)}{t.D} \qquad \neg t.D}{\mathit{false}}}{t \mathrel{\dot\in} U_1} \qquad \dfrac{\dfrac{}{t \in \neg D \longrightarrow U_1}\,(1)}{t \mathrel{\dot\in} U_1}}{t \mathrel{\dot\in} U_1}\,(1)$$

□

Then by propositions 4.29(i) and (ii), and the rule ($\mathtt{if}^{+}$), the following theorem is immediate:

Theorem 4.30.

$$\dfrac{t \in \mathtt{gif}\ D\ \mathtt{then}\ U_0\ \mathtt{else}\ U_1}{t \in \mathtt{if}\ D\ \mathtt{then}\ U_0\ \mathtt{else}\ U_1}$$

□

Turning now to showing that every conditional in the "if" theory is a valid guarded conditional.

Proposition 4.31.

$$\dfrac{t \in \mathtt{if}\ D\ \mathtt{then}\ U_0\ \mathtt{else}\ U_1 \quad t \in D \longrightarrow U_0 \vdash P \quad t \in \neg D \longrightarrow U_1 \vdash P}{P}$$

Proof Consider the following derivation, which requires the *law of excluded middle*:

$$\dfrac{\dfrac{}{t.D \vee \neg\, t.D}\,(LEM) \qquad \begin{matrix}\dfrac{\dfrac{}{t.D}\,(1) \qquad \dfrac{t \in \mathtt{if}\ D\ \mathtt{then}\ U_0\ \mathtt{else}\ U_1 \quad \dfrac{}{t.D}\,(1)}{t \mathrel{\dot\in} U_0}}{t \in D \longrightarrow U_0} \\ \vdots \\ P\end{matrix} \qquad \begin{matrix}\delta \\ \vdots \\ P\end{matrix}}{P}\,(1)$$

Where δ stands for the following branch:

$$\begin{matrix}\dfrac{\dfrac{}{\neg t.D}\,(1) \qquad \dfrac{t \in \mathtt{if}\ D\ \mathtt{then}\ U_0\ \mathtt{else}\ U_1 \quad \dfrac{}{\neg t.D}\,(1)}{t \mathrel{\dot\in} U_1}}{t \in \neg D \longrightarrow U_1} \\ \vdots \\ P\end{matrix}$$

□

Then by proposition 4.31, in addition to the rules ($\mathtt{gif}_{\mathit{0}}^{+}$) and ($\mathtt{gif}_{\mathit{1}}^{+}$), the following theorem is immediately derivable:

Theorem 4.32.

$$\dfrac{t \in \mathtt{if}\ D\ \mathtt{then}\ U_0\ \mathtt{else}\ U_1}{t \in \mathtt{gif}\ D\ \mathtt{then}\ U_0\ \mathtt{else}\ U_1}$$

□

Together, theorems 4.30 and 4.32 demonstrate that the concepts of *conditional* and *guarded conditional* control structures are equivalent.

4.11.4. Guarded Case Statement

We generalise the guarded conditional to guarded case statement. This is defined as *parallel composition* of commands whose guards are drawn from a given set of values:

Definition 4.14. Let $T = \{\cdots c_i \cdots\}$.

$$\texttt{gcases}\ E^T\ \texttt{in}\ c_0 : U_0^{\mathbb{P}\,T_0}\ \cdots\ c_n : U_n^{\mathbb{P}\,T_n}\ \texttt{endgcases} =_{df} \bigvee_{i=0}^{n} E = c_i \longrightarrow U_i$$

The following introduction and elimination rules are derivable for guarded cases:

Proposition 4.33. Let $i \in n+1$.

$$\frac{t.(E = c_i) \quad t \mathrel{\dot\in} U_i}{t \in \texttt{gcases}\ E\ \texttt{in}\ c_0 : U_0\ \cdots\ c_n : U_n\ \texttt{endgcases}}\ (\texttt{gcases}_i^+)$$

$$\frac{t \in \texttt{gcases}\ E\ \texttt{in}\ c_0 : U_0\ \cdots\ c_n : U_n\ \texttt{endgcases} \quad \cdots \quad t \in E = c_i \longrightarrow U_i \vdash P \quad \cdots}{P}\ (\texttt{gcases}^-)$$

□

In light of theorems 4.30 and 4.32, in conjunction with the fact that the “cases” (section 4.4) and “gcases” theories respectively generalise the theories of “if” (section 4.3) and “gif” (section 4.11.3), it is evident that the former theories are also equivalent; the rest of the section is devoted to proving this result. We begin by showing that every guarded case statement is also a valid case statement in the “cases” theory.

Proposition 4.34. Let $i \in n+1$, then the following rule is derivable:

$$\frac{t \in \texttt{gcases}\ E\ \texttt{in}\ c_0 : U_0\ \cdots\ c_n : U_n\ \texttt{endgcases} \quad t.(E = c_i)}{t \mathrel{\dot\in} U_i}$$

Proof Let $k \in n+1$, where $k \neq i$.

$$\frac{t \in \texttt{gcases}\ E\ \texttt{in}\ c_0 : U_0\ \cdots\ c_n : U_n\ \texttt{endgcases} \qquad \dfrac{\dfrac{}{t \in E = c_i \longrightarrow U_i}\,(1)}{t \mathrel{\dot\in} U_i} \qquad \begin{matrix}\delta \\ \vdots \\ t \mathrel{\dot\in} U_i\end{matrix} \quad \cdots}{t \mathrel{\dot\in} U_i}\ (1)$$

Where δ stands for the following branch:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{t \in E = c_k \longrightarrow U_k}\,(1)}{t.(E = c_k)} \qquad t.(E = c_i)}{c_k = c_i}}{\mathit{false}}}{t \mathrel{\dot\in} U_i}$$

□

This (in conjunction with the rule ($\texttt{cases}^+$)) leads directly to the following theorem:

Theorem 4.35.

$$\frac{t \in \texttt{gcases}\ E\ \texttt{in}\ c_0 : U_0\ \cdots\ c_n : U_n\ \texttt{endgcases}}{t \in \texttt{cases}\ E\ \texttt{in}\ c_0 : U_0\ \cdots\ c_n : U_n\ \texttt{endcases}}$$

□

Turning now to showing that every case statement is also a valid guarded case statement.

Proposition 4.36. Let $i \in n+1$, then the following rule is derivable:

$$\frac{t \in \texttt{cases}\ E\ \texttt{in}\ c_0 : U_0\ \cdots\ c_n : U_n\ \texttt{endcases} \quad \cdots \quad t \in E = c_i \longrightarrow U_i \vdash P \quad \cdots}{P}$$

Proof

$$\dfrac{\dfrac{t \in \texttt{cases}\ E\ \texttt{in}\ c_0 : U_0\ \cdots\ c_n : U_n\ \texttt{endcases}}{\bigvee_{i=0}^{n} t.(E = c_i)} \quad \cdots \quad \begin{matrix}\delta\\ \vdots\\ P\end{matrix} \quad \cdots}{P}\ (1)$$

Where δ stands for the following branch:

$$\dfrac{\overline{t.(E = c_i)}\ (1) \quad \dfrac{t \in \texttt{cases}\ E\ \texttt{in}\ c_0 : U_0\ \cdots\ c_n : U_n\ \texttt{endcases} \quad \overline{t.(E = c_i)}\ (1)}{t \mathrel{\dot\in} U_i}}{t \in E = c_i \longrightarrow U_i} \\ \vdots \\ P$$

□

Then by proposition 4.36, in addition to the rules ($\texttt{gcases}_i^+$), we get the following theorem immediately:

Theorem 4.37.

$$\dfrac{t \in \texttt{cases}\ E\ \texttt{in}\ c_0 : U_0\ \cdots\ c_n : U_n\ \texttt{endcases}}{t \in \texttt{gcases}\ E\ \texttt{in}\ c_0 : U_0\ \cdots\ c_n : U_n\ \texttt{endgcases}}$$

□

Theorems 4.35 and 4.37 together establish that the theories of *cases* and *guarded cases* are equivalent. This concludes the analysis.

4.12. While Loop

4.12.1. Logic and Semantics

Definition 4.15.

$$\texttt{while}\ D\ \texttt{do}\ U =_{df} \mu\, X \bullet D \longrightarrow U \mathbin{⨟} X \vee \neg D \longrightarrow \texttt{skip}$$

The following introduction and elimination rules are sound for the while loop:

Proposition 4.38.

$$\dfrac{\neg z.D \quad z \mathrel{\dot\in} \texttt{skip}}{z \in \texttt{while}\ D\ \texttt{do}\ U}\ (\texttt{while}_0^+)$$

Proof Let W be $\mu\, X \bullet D \longrightarrow U \mathbin{⨟} X \vee \neg D \longrightarrow \texttt{skip}$.

$$\dfrac{\dfrac{\dfrac{\neg z.D \quad z \mathrel{\dot\in} \texttt{skip}}{z \mathrel{\dot\in} \neg D \longrightarrow \texttt{skip}}}{z \in D \longrightarrow U \mathbin{⨟} W \vee \neg D \longrightarrow \texttt{skip}}}{z \in \texttt{while}\ D\ \texttt{do}\ U}\ (\mu^+)$$

□

Proposition 4.39.

$$\dfrac{z.D \quad z_0 \star y' \mathrel{\dot\in} U \quad z_0 =_{T_L} y \quad y \star z_1' \mathrel{\dot\in} \texttt{while}\ D\ \texttt{do}\ U \quad y =_{T_R} z_1}{z_0 \star z_1' \in \texttt{while}\ D\ \texttt{do}\ U}\ (\texttt{while}_1^+)$$

Proof Let W be $\mu\, X \bullet D \longrightarrow U \mathbin{⨟} X \vee \neg D \longrightarrow \texttt{skip}$.

$$\dfrac{\dfrac{\dfrac{z_0.z_1'.D \qquad \dfrac{z_0 \star y' \mathrel{\dot\in} U \quad z_0 =_{T_L} y \quad y \star z_1' \mathrel{\dot\in} \mathtt{while}\ D\ \mathtt{do}\ U \quad y =_{T_R} z_1}{z_0 \star z_1' \in U \fatsemi W}}{z_0 \star z_1' \mathrel{\dot\in} D \longrightarrow U \fatsemi W}}{z_0 \star z_1' \in D \longrightarrow U \fatsemi W \vee \neg D \longrightarrow \mathtt{skip}}}{z_0 \star z_1' \in \mathtt{while}\ D\ \mathtt{do}\ U}\ (\mu^+)$$

□

Proposition 4.40.

$$\dfrac{\begin{array}{lll} z \in \mathtt{while}\ D\ \mathtt{do}\ U & & \\ \neg z.D, z \mathrel{\dot\in} \mathtt{skip} & \vdash & P \\ z.D, z_0 \star y' \mathrel{\dot\in} U, z_0 =_{T_L} y, y \star z_1' \mathrel{\dot\in} \mathtt{while}\ D\ \mathtt{do}\ U, y =_{T_R} z_1 & \vdash & P \end{array}}{P}\ (\mathtt{while}^-)$$

Proof Let W be $\mu X \bullet D \longrightarrow U \fatsemi X \vee \neg D \longrightarrow \mathtt{skip}$.

$$\dfrac{\dfrac{z \in \mathtt{while}\ D\ \mathtt{do}\ U}{z \in D \longrightarrow U \fatsemi W \vee \neg D \longrightarrow \mathtt{skip}}\ (\mu^-) \qquad \begin{matrix}\delta_0\\ \vdots \\ P\end{matrix} \qquad \begin{matrix}\delta_2\\ \vdots \\ P\end{matrix}}{P}\ (1)$$

where δ_0 is:

$$\dfrac{\dfrac{\dfrac{}{z \in D \longrightarrow U \fatsemi W}\ (1a)}{z \mathrel{\dot\in} U \fatsemi W} \qquad \begin{matrix}\delta_1\\ \vdots \\ P\end{matrix}}{P}\ (2)$$

where δ_1 is:

$$\begin{matrix}\dfrac{\dfrac{\dfrac{}{z \mathrel{\dot\in} D \longrightarrow U \fatsemi W}\ (1a)}{z.D} \qquad \dfrac{}{z_0 \star y' \mathrel{\dot\in} U}\ (2) \qquad \dfrac{}{z_0 =_{T_L} y}\ (2) \qquad \dfrac{}{y \star z_1' \mathrel{\dot\in} W}\ (2) \qquad \dfrac{}{y =_{T_R} z_1}\ (2)}{\vdots} \\ P\end{matrix}$$

where δ_2 is:

$$\begin{matrix}\dfrac{\dfrac{\dfrac{}{z \mathrel{\dot\in} \neg D \longrightarrow \mathtt{skip}}\ (1b)}{\neg z.D} \qquad \dfrac{\dfrac{}{z \mathrel{\dot\in} \neg D \longrightarrow \mathtt{skip}}\ (1b)}{z \mathrel{\dot\in} \mathtt{skip}}}{\vdots} \\ P\end{matrix}$$

□

The following additional rules are derivable:

Lemma 4.41.

$$\dfrac{z \in \mathtt{while}\ D\ \mathtt{do}\ U \quad \neg z.D}{z \in \mathtt{skip}}$$

Proof

$$\dfrac{z \in \mathtt{while}\ D\ \mathtt{do}\ U \qquad \dfrac{}{z \in \mathtt{skip}}\ (0) \qquad \dfrac{\dfrac{\neg z.D \quad \dfrac{}{z.D}\ (0)}{\mathit{false}}}{z \in \mathtt{skip}}}{z \in \mathtt{skip}}\ (0)$$

□

Lemma 4.42.

$$\frac{z \in \mathtt{while}\ D\ \mathtt{do}\ U \quad z.D}{z \in U \mathbin{;} \mathtt{while}\ D\ \mathtt{do}\ U}$$

Proof

$$\frac{z \in \mathtt{while}\ D\ \mathtt{do}\ U \quad \dfrac{\dfrac{z.D \quad \overline{\neg z.D}\ (0)}{\mathit{false}}}{z \in U \mathbin{;} \mathtt{while}\ D\ \mathtt{do}\ U} \quad \begin{array}{c}\delta_0\\ \vdots\\ z \in U \mathbin{;} \mathtt{while}\ D\ \mathtt{do}\ U\end{array}}{z \in U \mathbin{;} \mathtt{while}\ D\ \mathtt{do}\ U}\ (0)$$

where δ_0 is:

$$\frac{\overline{z_0 \star y' \mathrel{\dot\in} U}\ (0) \quad \overline{z_0 =_{T_L} y}\ (0) \quad \overline{y \star z_1' \mathrel{\dot\in} \mathtt{while}\ D\ \mathtt{do}\ U}\ (0) \quad \overline{y =_{T_R} z_1}\ (0)}{z \in U \mathbin{;} \mathtt{while}\ D\ \mathtt{do}\ U}$$

□

4.12.2. Inequational Refinement Logic

Proposition 4.43.

$$\frac{\begin{array}{ll}\neg z.D & \vdash\ \mathtt{skip} \sqsupseteq U_1[0]\\ z.D, \mathtt{while}\ D\ \mathtt{do}\ U_0[f(n)] \sqsupseteq U_1[f(n)] & \vdash\ U_0 \mathbin{;} \mathtt{while}\ D\ \mathtt{do}\ U_0[f(n)] \sqsupseteq U_1[n]\end{array}}{\mathtt{while}\ D\ \mathtt{do}\ U_0 \sqsupseteq U_1}$$

Proof

$$\frac{\dfrac{\dfrac{\begin{array}{c}\delta_0\\ \vdots\\ z \in U_1[0]\end{array}}{\mathtt{while}\ D\ \mathtt{do}\ U_0[0] \sqsupseteq U_1[0]}\ (2) \quad \dfrac{\begin{array}{c}\delta_2\\ \vdots\\ z \in U_1[n]\end{array}}{\mathtt{while}\ D\ \mathtt{do}\ U_0[n] \sqsupseteq U_1[n]}\ (3)}{\mathtt{while}\ D\ \mathtt{do}\ U_0[n] \sqsupseteq U_1[n]}\ (1)}{\mathtt{while}\ D\ \mathtt{do}\ U_0 \sqsupseteq U_1}\ (0)$$

where δ_0 is:

$$\frac{\dfrac{\dfrac{\dfrac{\overline{z \in \mathtt{while}\ D\ \mathtt{do}\ U_0[0]}\ (2)}{z \mathrel{\dot\in} \mathtt{while}\ D\ \mathtt{do}\ U_0} \quad \dfrac{\begin{array}{c}\delta_1\\ \vdots\\ z.\mathsf{y} = 0\end{array}}{\neg z.D}}{z \mathrel{\dot\in} \mathtt{skip}}\ 4.41 \quad \begin{array}{c}\delta_1\\ \vdots\\ \dfrac{z.\mathsf{y} = 0}{\neg z.D}\\ \vdots\\ \mathtt{skip} \sqsupseteq U_1\end{array}}{z \mathrel{\dot\in} U_1} \quad \begin{array}{c}\delta_1\\ \vdots\\ z.\mathsf{y} = 0\end{array}}{z \in U_1[0]}$$

where δ_1 is:

$$\frac{\overline{z \in \mathtt{while}\ D\ \mathtt{do}\ U_0[0]}\ (2)}{z.\mathsf{y} = 0}$$

where δ_2 is:

$$\frac{\begin{array}{c}\delta_4\\ \vdots\\ z \in U_0 \mathbin{;} \mathtt{while}\ D\ \mathtt{do}\ U_0[f(n)]\end{array} \quad \begin{array}{c}\delta_3\\ \vdots\\ U_0 \mathbin{;} \mathtt{while}\ D\ \mathtt{do}\ U_0[f(n)] \sqsupseteq U_1[n]\end{array}}{z \in U_1[n]}$$

where δ_3 is:

$$\cfrac{\cfrac{\cfrac{\cfrac{}{z \in \mathtt{while}\ D\ \mathtt{do}\ U_0[n]}\ (3)}{z.\mathtt{y} = n} \qquad \cfrac{}{n > 0}\ (1)}{z.D} \qquad \cfrac{}{\mathtt{while}\ D\ \mathtt{do}\ U_0[f(n)] \sqsupseteq U_1[f(n)]}\ (1)}{\begin{array}{c}\vdots \\ U_0 \fatsemi \mathtt{while}\ D\ \mathtt{do}\ U_0[f(n)] \sqsupseteq U_1[n]\end{array}}$$

where δ_4 is:

$$\cfrac{\cfrac{\cfrac{}{z \in \mathtt{while}\ D\ \mathtt{do}\ U_0[n]}\ (3)}{z\ \dot{\in}\ \mathtt{while}\ D\ \mathtt{do}\ U_0} \qquad \begin{array}{c}\delta_5, \delta_6 \\ \vdots \\ z \in U_0 \fatsemi \mathtt{while}\ D\ \mathtt{do}\ U_0[f(n)]\end{array}}{z \in U_0 \fatsemi \mathtt{while}\ D\ \mathtt{do}\ U_0[f(n)]}\ (4)$$

where δ_5 is:

$$\cfrac{\cfrac{\cfrac{}{\neg z.D}\ (4) \qquad \cfrac{\cfrac{\cfrac{}{z \in \mathtt{while}\ D\ \mathtt{do}\ U_0[n]}\ (3)}{z.\mathtt{y} = n} \qquad \cfrac{}{n > 0}\ (1)}{z.D}}{\mathit{false}}}{z \in U_0 \fatsemi \mathtt{while}\ D\ \mathtt{do}\ U_0[f(n)]}$$

where δ_6 is:

$$\cfrac{\cfrac{}{z_0 \star w'\ \dot{\in}\ U_0}\ (4) \qquad \cfrac{}{z_0 =_{T_L} w}\ (4) \qquad \begin{array}{c}\delta_7 \\ \vdots \\ w \star z_1'\ \dot{\in}\ \mathtt{while}\ D\ \mathtt{do}\ U_0[f(n)]\end{array} \qquad \cfrac{}{w =_{T_R} z_1}\ (4)}{z \in U_0 \fatsemi \mathtt{while}\ D\ \mathtt{do}\ U_0[f(n)]}$$

where δ_7 is:

$$\cfrac{\cfrac{}{w \star z_1'\ \dot{\in}\ \mathtt{while}\ D\ \mathtt{do}\ U_0}\ (4) \qquad \cfrac{\cfrac{\cfrac{}{z_0 \star w'\ \dot{\in}\ U_0}\ (4)}{f(z_0.\mathtt{y}) = w.\mathtt{y}}\ (\clubsuit) \qquad \begin{array}{c}\delta_8 \\ \vdots \\ z_0.\mathtt{y} = n\end{array}}{w.\mathtt{y} = f(n)}}{w \star z_1'\ \dot{\in}\ \mathtt{while}\ D\ \mathtt{do}\ U_0[f(n)]}$$

where δ_8 is:

$$\cfrac{\cfrac{}{z \in \mathtt{while}\ D\ \mathtt{do}\ U_0[n]}\ (3)}{z_0.\mathtt{y} = n}$$

□

Rule used to perform step (1)

$$\frac{P(0) \qquad n > 0, m < n, P(m) \vdash P(n)}{P(n)}\ (1)$$

$$\frac{z_0 \star z_1' \in U}{f(z_0.\mathtt{y}) = z_1.\mathtt{y}}\ (\clubsuit)$$

4.12.3. General Refinement Logic

We generalise on the previous section in two aspects: first, the variant now depends on a particular state, rather than on a single observation; secondly, the state observations are not necessarily numeric. This concept is easily attained by defining a function f which associates every state of the system with a particular numeric value. Namely:

$$f \in T \to \mathbb{N}$$

Then the following rule is derivable:

Proposition 4.44.

$$\dfrac{\mathtt{skip} \sqsupseteq U_1 \uparrow \neg D \qquad \begin{array}{c}(\mathtt{while}\ D\ \mathtt{do}\ U_0) \uparrow (D \wedge f(\theta T) = m) \sqsupseteq U_1 \uparrow (D \wedge f(\theta T) = m) \vdash \\ U_0 \fatsemi (\mathtt{while}\ D\ \mathtt{do}\ U_0) \uparrow (D \wedge f(\theta T) = m) \sqsupseteq U_1 \uparrow (D \wedge f(\theta T) = n)\end{array}}{\mathtt{while}\ D\ \mathtt{do}\ U_0 \sqsupseteq U_1}$$

Proof Consider the following derivation which employs *course of values* induction:

$$\dfrac{\begin{array}{c}\delta_0\\ \vdots \\ (\mathtt{while}\ D\ \mathtt{do}\ U_0) \uparrow \neg D \sqsupseteq U_1 \uparrow \neg D\end{array} \qquad \dfrac{\begin{array}{c}\delta_1\\ \vdots \\ (\mathtt{while}\ D\ \mathtt{do}\ U_0) \uparrow (D \wedge f(\theta T) = x) \sqsupseteq U_1 \uparrow (D \wedge f(\theta T) = x)\end{array}}{(\mathtt{while}\ D\ \mathtt{do}\ U_0) \uparrow D \sqsupseteq U_1 \uparrow D}}{\mathtt{while}\ D\ \mathtt{do}\ U_0 \sqsupseteq U_1}$$

Where δ_0 stands for the following branch:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{z \in (\mathtt{while}\ D\ \mathtt{do}\ U_0) \uparrow \neg D}\,(1) \quad \dfrac{}{\neg z.D}\,(2)}{z \in \mathtt{while}\ D\ \mathtt{do}\ U_0} \quad \begin{array}{c}\eta\\ \vdots \\ z \mathrel{\dot\in} U_1\end{array} \quad \dfrac{\dfrac{\dfrac{}{z.D}\,(3) \quad \dfrac{}{\neg z.D}\,(2)}{\mathit{false}}}{z \mathrel{\dot\in} U_1}}{\dfrac{z \mathrel{\dot\in} U_1}{z \in U_1 \uparrow \neg D}\,(2)}\,(3)}{(\mathtt{while}\ D\ \mathtt{do}\ U_0) \uparrow \neg D \sqsupseteq U_1 \uparrow \neg D}\,(1)$$

Where η is:

$$\dfrac{\dfrac{\dfrac{}{z \in \mathtt{skip}}\,(3) \quad \mathtt{skip} \sqsupseteq U_1 \uparrow \neg D}{z \in U_1 \uparrow \neg D} \quad \dfrac{}{\neg z.D}\,(3)}{z \mathrel{\dot\in} U_1}$$

Let φ_0 and φ_n respectively be:

$$(\mathtt{while}\ D\ \mathtt{do}\ U_0) \uparrow (D \wedge f(\theta T) = 0) \sqsupseteq U_1 \uparrow (D \wedge f(\theta T) = 0)$$

and

$$(\mathtt{while}\ D\ \mathtt{do}\ U_0) \uparrow (D \wedge f(\theta T) = n) \sqsupseteq U_1 \uparrow (D \wedge f(\theta T) = n)$$

then δ_1 stands for the following branch:

$$\dfrac{\begin{array}{c}\beta_0\\ \vdots \\ \varphi_0\end{array} \qquad \begin{array}{c}\beta_1\\ \vdots \\ \varphi_n\end{array}}{(\mathtt{while}\ D\ \mathtt{do}\ U_0 \uparrow (D \wedge f(\theta T) = x) \sqsupseteq U_1 \uparrow (D \wedge f(\theta T) = x)}\,(4)$$

Where β_0 is:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{z \in (\mathtt{while}\ D\ \mathtt{do}\ U_0) \uparrow (D \wedge f(\theta T) = 0)}\,(5) \quad \dfrac{}{z.(D \wedge f(\theta T) = 0)}\,(6)}{z \in \mathtt{while}\ D\ \mathtt{do}\ U_0} \quad \begin{array}{c}\alpha_0\\ \vdots \\ z \mathrel{\dot\in} U_1\end{array} \quad \begin{array}{c}\alpha_1\\ \vdots \\ z \mathrel{\dot\in} U_1\end{array}}{z \mathrel{\dot\in} U_1}\,(7)}{z \in U_1 \uparrow (D \wedge f(\theta T) = 0)}\,(6)}{(\mathtt{while}\ D\ \mathtt{do}\ U_0) \uparrow (D \wedge f(\theta T) = 0) \sqsupseteq U_1 \uparrow (D \wedge f(\theta T) = 0)}\,(5)$$

and α_0, α_1 are respectively:

$$\dfrac{\dfrac{\dfrac{}{z \in \mathtt{skip}}\,(7) \quad \mathtt{skip} \sqsupseteq U_1 \uparrow \neg D}{z \in U_1 \uparrow \neg D} \quad \dfrac{}{\neg z.D}\,(7)}{z \mathrel{\dot\in} U_1} \qquad \dfrac{\dfrac{\dfrac{}{z.D}\,(7) \quad \dfrac{\dfrac{\dfrac{\dfrac{}{z.(D \wedge f(\theta T) = 0)}\,(6)}{z.(f(\theta T) = 0)}}{f(z) = 0}}{\neg z.D}\,(\clubsuit)}{\mathit{false}}}{z \mathrel{\dot\in} U_1}$$

β_1 stands for the following branch:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{z \in (\mathtt{while}\ D\ \mathtt{do}\ U_0)\uparrow(D \wedge f(\theta T) = n)}\,(8) \quad \dfrac{}{z.(D \wedge f(\theta T) = n)}\,(9)}{z \in \mathtt{while}\ D\ \mathtt{do}\ U_0} \quad \begin{matrix}\gamma_0\\ \vdots \\ z \mathrel{\dot{\in}} U_1\end{matrix} \quad \begin{matrix}\gamma_1\\ \vdots\\ z \mathrel{\dot{\in}} U_1\end{matrix}}{z \mathrel{\dot{\in}} U_1}\,(10)}{z \in U_1 \uparrow (D\wedge f(\theta T)=n)}\,(9)}{(\mathtt{while}\ D\ \mathtt{do}\ U_0)\uparrow(D\wedge f(\theta T)=n) \sqsupseteq U_1\uparrow(D\wedge f(\theta T)=n)}\,(8)$$

Where γ_0 is:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{z.(D \wedge f(\theta T) = n)}\,(9)}{z.D} \quad \dfrac{}{\neg z.D}\,(10)}{\mathit{false}}}{z \mathrel{\dot{\in}} U_1}$$

Let ψ be:

$$z \in U_0 \mathbin{;} (\mathtt{while}\ D\ \mathtt{do}\ U_0)\uparrow(D \wedge f(\theta T) = m)$$

then γ_1 branch is:

$$\dfrac{\dfrac{\begin{matrix}\dfrac{}{(\mathtt{while}\ D\ \mathtt{do}\ U_0)\uparrow(D\wedge f(\theta T)=m) \sqsupseteq U_1\uparrow(D\wedge f(\theta T)=m)}\,(4)\\ \vdots \\ U_0 \mathbin{;} (\mathtt{while}\ D\ \mathtt{do}\ U_0)\uparrow(D\wedge f(\theta T)=m) \sqsupseteq U_1\uparrow(D\wedge f(\theta T)=n)\end{matrix} \quad \begin{matrix}\gamma_2\\ \vdots\\ \psi\end{matrix}}{z \in U_1\uparrow(D\wedge f(\theta T)=n)} \quad \dfrac{}{z.(D\wedge f(\theta T)=n)}\,(9)}{z \mathrel{\dot{\in}} U_1}$$

Where γ_2 stands for the following branch:

$$\dfrac{\dfrac{}{z_0 \star y' \mathrel{\dot{\in}} U_0}\,(10) \quad \dfrac{}{z_0 =_{T_L} y}\,(10) \quad \dfrac{\dfrac{}{y \star z_1' \mathrel{\dot{\in}} \mathtt{while}\ D\ \mathtt{do}\ U}\,(10)}{y \star z_1' \mathrel{\dot{\in}} (\mathtt{while}\ D\ \mathtt{do}\ U_0)\uparrow(D\wedge f(\theta T)=m)} \quad \dfrac{}{y =_{T_R} z_1}\,(10)}{z \in U_0 \mathbin{;} (\mathtt{while}\ D\ \mathtt{do}\ U_0)\uparrow(D\wedge f(\theta T)=m)}$$

□

5. Conclusions and Further Work

As we mentioned in the introduction, this expository paper concentrates entirely on the theoretical basis of νZ. We have showed how an extremely simple logic can be extended towards an expressive specification logic and a program (development) logic. One of the benefits of this approach is its flexibility: one is not constrained by any particular specification or programming language infrastructure. The ability to provide elegant rules for total correctness development of procedures is also a strength: these rules resemble those which proved so useful in program development within constructive theories (see, for example, [26]) but are here combined with the ability to synthesize imperative programs.

Much infrastructural and pragmatic work remains to be done, both at the level of specification and program development. At the pragmatic level in particular, much work is being undertaken by Kajtazi and this will be reported in his PhD thesis.

6. Acknowledgements

The authors are grateful to Steve Reeves and Lindsay Groves for numerous discussions and critical appraisal of this work. We would like to acknowledge the support of the EPSRC RefineNet Network Grant (Ref: GR/S69979/02) in the development of this work, including comments from participants at the January 2004 meeting in Sheffield where an earlier version of this material was presented.

References

[1] J. R. Abrial. *The B-Book*. Cambridge University Press, 1996.

[2] J. R. Abrial. Extending B without Changing it (for Developing Distributed Systems). In H. Habrias, editor, *Proceedings of the 1st Conference on the B Method, Nantes, France, 24-26 November, 1996*, pages 169–190. Institut de Recherche en Informatique de Nantes (IRIN), 1996.

[3] D. Azada and P. Muenchaisri, editors. *APSEC 2003: 10th Asia-Pacific Software Engineering Conference, Chiangmai, Thailand, December 10-12, 2003, Proceedings*. IEEE Computer Society Press, December 2003.

[4] R. J. R. Back and R. Kurki-Suonio. Decentralization of Process Nets with Centralized Control. In *Proceedings of the 2nd Annual ACM Symposium on Principles of Distributed Computing, Montreal, Quebec, Canada, August, 1983*, pages 131–142. ACM Press, 1983.

[5] R. J. R. Back and K. Sere. Deriving an Occam Implementation of Action Systems. *Technical Report A-99. Department of Computer Science, Åbo Akademi University, Turku – Finland*, January 1990.

[6] R. J. R. Back and J. von Wright. *Refinement Calculus: A Systematic Introduction*. Springer, 1998.

[7] J. P. Bowen, S. Dunne, A. Galloway, and S. King, editors. *ZB 2000: Formal Specification and Development in Z and B, First International Conference of B and Z Users, York, UK, August 29 - September 2, 2000, Proceedings*, volume 1878 of *Lecture Notes in Computer Science*. Springer, 2000.

[8] M. J. Butler. An Approach to the Design of Distributed Systems with B and AMN. In J. P. Bowen, M. G. Hinchey, and D. Till, editors, *ZUM '97: The Z Formal Specification Notation, 10th International Conference of Z Users Reading, UK, April, 1997*, volume 1212 of *Lecture Notes in Computer Science*, pages 223–241. Springer-Verlag, 1997.

[9] M. J. Butler and M. Waldén. Distributed System Development in B. *Technical Report TUCS-53. Turku Centre for Computer Science, Åbo Akademi University, Turku – Finland*, October 1996.

[10] D.Bert, J. P. Bowen, S. King, and M. Waldén, editors. *ZB 2003: Formal Specification and Development in Z and B, Third International Conference of B and Z Users, Turku, Finland, June 4-6, 2003, Proceedings*, volume 2651 of *Lecture Notes in Computer Science*. Springer, 2003.

[11] W. P. de Roever and K. Engelhardt. *Data Refinement: Model-Oriented Proof Methods and Their Comparison*. Prentice Hall International, 1998.

[12] J. Derrick and E. A. Boiten. *Refinement in Z and Object-Z: Foundations and Advanced Applications*. Formal Approaches to Computing and Information Technology – FACIT. Springer, May 2001.

[13] J. Derrick and E. A. Boiten, editors. *REFINE 2005 International Workshop*, Electronic Notes in Theoretical Computer Science. BCS-FACS, April 2005. To appear.

[14] M. Deutsch. *An Analysis of Total Correctness Refinement Models for Partial Relation Semantics*. PhD thesis, University of Essex, 2005.

[15] M. Deutsch and M. C. Henson. An Analysis of Backward Simulation Data-Refinement for Partial Relation Semantics. In *APSEC 2003 [3]*, pages 38–48, 2003.

[16] M. Deutsch and M. C. Henson. An Analysis of Forward Simulation Data Refinement. In *ZB 2003 [10]*, pages 148–167, 2003.

[17] M. Deutsch and M. C. Henson. An analysis of total correctness refinement models for partial relation semantics II. *Logic Journal of the IGPL*, 11(3):319–352, 2003.

[18] M. Deutsch and M. C. Henson. An Analysis of Operation-Refinement in an Abortive Paradigm. In *REFINE 2005 [13]*, 2005.

[19] M. Deutsch, M. C. Henson, and S. Reeves. Results on Formal Stepwise Design in Z. In P. Strooper and P. Muenchaisri, editors, *APSEC 2002: 9th Asia-Pacific Software Engineering Conference, Gold Coast, Queensland, Australia, December 4-6, 2002, Proceedings*, pages 33–42. IEEE Computer Society Press, December 2002.

[20] M. Deutsch, M. C. Henson, and S. Reeves. An analysis of total correctness refinement models for partial relation semantics I. *Logic Journal of the IGPL*, 11(3):287–317, 2003.

[21] M. Deutsch, M. C. Henson, and S. Reeves. Modular reasoning in Z: scrutinising monotonicity and refinement. *University of Essex, technical report CSM-407 (under consideration of FACJ)*, December 2003.

[22] E. W. Dijkstra. *A Discipline of Programming*. Prentice Hall, 1976.

[23] M. C. Henson and S. Reeves. Investigating Z. *Logic and Computation*, 10(1):43–73, 2000.

[24] C.A.R Hoare and J. He. *Unifying Theories of Programming*. Prentice Hall International, 1998.

[25] M. B. Josephs. Specifying Reactive Systems in Z. *Technical Monograph PRG-TR-19-91. Oxford University Computing Laboratory*, 1991.

[26] P. Martin-Löf. Constructive mathematics and computer programming. In *Logic, Methodology and Philosophy of Science VI*, pages 153–175. North Holland, 1982.

[27] R. Miarka, E. A. Boiten, and J. Derrick. Guards, Preconditions, and Refinement in Z. In *ZB 2000 [7]*, pages 286–303, 2000.

[28] C. C. Morgan. *Programming from Specifications*. Prentice Hall International, 2nd edition, 1994.

[29] S. Schneider. *The B-Method – An Introduction*. Correctness of Computing. Palgrave, 2001.

[30] B. Strulo. How Firing Conditions Help Inheritance. In J. P. Bowen and M. G. Hinchey, editors, *ZUM '95: The Z Formal Specification Notation, 9th International Conference of Z Users Limerick, Ireland, September 7-9, 1995*, volume 967 of *Lecture Notes in Computer Science*, pages 264–275. Springer-Verlag, 1995.

[31] I. Toyn, editor. *Z Notation: Final Committee Draft, CD 13568.2*. Z Standards Panel, 1999.

[32] M. Waldén and K. Sere. Refining Action Systems within B-Tool. In M. Naftalin, T. Denvir, and M. Bertran, editors, *FME '96: Industrial Benefit and Advances in Formal Methods, 3rd International Symposium of Formal Methods Europe, Oxford, UK, March 18-22, 1996, Proceedings*, volume 1051 of *Lecture Notes in Computer Science*, pages 84–103. Springer, 1996.

[33] J. Woodcock and S. Brien. $\mathcal{W}$: A logic for $\mathcal{Z}$. In *Proceedings of ZUM '91, 6th Conf. on Z*. Springer Verlag, 1992.

[34] J. C. P. Woodcock and J. Davies. *Using Z: Specification, Refinement and Proof*. Prentice Hall, 1996.