SYSTEMATIC SOFTWARE DEVELOPMENT USING VDM

Second Edition

Teaching Notes

CLIFF B JONES

June 18, 1995

Contents

0

Introduction

This report provides information which should be of use in teaching courses which are based on the second edition of 'Systematic Software Development using VDM', (Prentice-Hall International) [Jon90]. The main chapters follow those of the book and contain both comments on the material and answers to many of the exercises.

The author would be grateful for feedback both on errors and proposals for extensions to these Teacher's Notes.

0.1 Brief history of VDM

VDM is a formal method for the description and development of computer systems. Its formal descriptions (or 'specifications') use mathematical notation to provide a precise statement of the intended function of a system. Such descriptions are built in terms of *models* of an underlying state with a collection of operations which are specified by pre- and post-conditions. VDM designs are guided by a number of *proof obligations* whose discharge establishes the correctness of design by either data reification or operation decomposition. Thus it can be seen that VDM addresses the stages of development from specification through to code.

VDM (Vienna Development Method) owes its existence to the IBM Laboratory in Vienna. The origins of that laboratory go back to a group which Heinz Zemanek brought from the *Technische Hochschule* (now the 'Technical University of Vienna'). The group initially worked on hardware projects. A compiler for ALGOL 60 followed. The recognition that language definition was a crucial issue for the future safe application of computers was emphasized by IBM's creation of the PL/I language. The Vienna group built on ideas of Elgot, Landin and McCarthy to create an *operational semantics* approach capable of defining the whole of PL/I including its TASKING features which involved parallelism. These massive reports were known internally as the 'Universal Language Document 3' and appeared in three more or less complete versions. The meta-language used was dubbed by outsiders the 'Vienna Definition Language' or VDL (see [**?**]). These descriptions were used as the basis for research into compiler design in 1968/70 [JL71].

The attempts to use the VDL definitions in design were in one sense successful; but they also showed clearly how the operational semantics approach could complicate formal reasoning in an unnecessary way. The Scott/Strachey/Landin work on *denotational semantics* was at the time taking shape in Oxford, Hans Bekič had long been pressing the Vienna group to adopt a more mathematical approach, and Cliff Jones had shown a 'functional semantics' for ALGOL 60 in a Hursley Technical Report [ACJ72]. The challenge, starting in late 1972, to design a compiler which translated the evolving ECMA/ANSI standard PL/I language into the order code of a completely novel machine presented the ideal opportunity to try out the denotational semantics approach. The project was fraught with difficulties and did not result in a finished compiler because of IBM's decision to abandon the machine architecture. But it did create VDM.

The formal description of PL/I in a denotational style is contained in a Technical Report [**?**] which was authored by Hans Bekič, Dines Bjørner, Wolfgang Henhapl, Cliff Jones and Peter

Lucas. The specification notation used became known as 'Meta-IV' (both this awful pun and the name 'VDM' are due to Dines Bjørner).

The diversion of the IBM group to handle more practical problems led to its effective dissolution. Among others to leave, Wolfgang Henhapl became a Professor in Darmstadt, Peter Lucas moved to IBM Research in the US, and Dines Bjørner took a visiting chair at Copenhagen and then a permanent one at the Technical University of Denmark. Of the key people Hans Bekič remained pursuing – in his spare time – important research on parallelism until his untimely death in 1982 (see [BJ84]).

Like other dispersions of scientists, this one did not kill the ideas but led to a larger community. The first step was to publish what had been done: Dines Bjørner and Cliff Jones edited Springer's LNCS 61 [BJ78] to this end. Dines Bjørner pursued the language description and compiler development work with Danish colleagues. This led to descriptions of both Ada [BO80b] and CHILL and the first validated European compiler for the Ada language. Cliff Jones picked up the work he had been doing on formal development methods for non-compiler problems. Several books have been published by Prentice Hall on VDM [Jon80b, BJ82, Jon86d, Jon90]. There are also numerous papers tackling problems such as parallelism (e.g. [Jon83a]). Peter Lucas has applied formal methods to application problems and Wolfgang Henhapl has worked on a support system (PSG) for VDM specifications.

There is now a BSI group preparing the standardisation of VDM which is chaired by Derek Andrews[1] and has the reference BSI IST/5/50. There is is also an EEC sponsored study group on VDM: 'VDM-Europe' is chaired by Søren Prehn[2]. It has already organised two conference the proceedings of which are published as [Jon87a, BJM88]; a further conference is scheduled for Kiel in April 1990. Public courses on VDM are offered by IST, Logica, Praxis, NCC etc. Course material on VDM (PM687) is available from the (UK) Open University.

0.2 Some background references

There are many good textbooks on classical logic – a useful one is [Ham82]; a textbook which describes natural deduction proofs very clearly is [NS85]. The work on LPF was described in [BCJ84] which also refers to other approaches to the same problem; Jen Cheng's thesis is [Che86] and a recent survey paper is [CJ90].

The research of the Vienna group was first described in research reports and papers. The first book – which contains references to the source papers – was [BJ78]. The program development aspects of VDM were described in [Jon80b] which was used in a number of industrial courses. This was developed in [Jon86d] where there is an emphasis on proof using natural deduction; this and the specific use of LPF have a large influence on the presentation. Although parallelism is not covered, the author's work in this area had also prompted some changes of notation. The application of VDM to programming language semantics is covered in [BJ82] which largely supercedes the earlier LNCS volume. Recent research on data reification is described in [Nip86, Nip87].

There are other books ranging, from monographs to textbooks, on formal methods. The reader who wishes to try VDM on some standard examples could extract them from these references. This would be particularly useful for the operation decomposition method described in Chapter 10. The method in [Jon90] differs from the referenced books because of the use of post-conditions of pairs of states. Some references are [Dij76, Gri81, Rey81, Heh, Bac86, Den86, SC87, Inc88].

There are very many VDM 'case studies' in the literature; an extensive bibliography is [Ras90].

[1] Mr. D.J. Andrews, Computer Studies Unit, University of Leicester, University Road, Leicester, LE1 7RH, U.K.

[2] Mr. S. Prehn, DDC, CRI A/S, Vesterbrogade, 1A, DK-1620, Copenhagen, Denmark

Unfortunately details of the notation in this material varies. Twelve studies have now been collected and updated to use BSI-VDM in [JS90].

0.3 Tool support

Various support tools are now available. 'Specbox'[3] is a parser and type checker for VDM which runs on PCs or workstations. A 'VDM Tool' has been built by IST[4] on their 'Genesis' system. The 'IPSE 2.5' project created a Theorem Proving Assistant known as *mural*[5]

Other theorem provers which could be tailored to VDM include [GMW79, Pau87, Gor88].

Acknowledgements

I am very grateful to the many teachers who have provided comments on their experiences with both my earlier and the current book. Particular thanks go to Bo Stig Hansen who sent me his own 'Student Notes' and to Peter Luckham who checked many of the answers given.

[3]Available from Adelard, 28, Rhonda Grove, London, E3 5AP

[4]Imperial Software Technology, 3, Glisson Road, Cambridge, CB1 2HA, U.K.

[5]Available via PEVE Unit, Department of Computer Science, Manchester University, M13 9PL, U.K.

1

Logic of Propositions

1.1 Comments

In Section 1.3, all of the inference rules are given without additional hypotheses. This matches the boxes and lines style of proof. Of course, one could write:

$$\boxed{\vee\text{-}I}\ \frac{,\ \vdash\ E_1}{,\ \vdash\ E_1 \vee E_2}$$

$$\boxed{\wedge\text{-}I}\ \frac{,_1\ \vdash\ E_1;\ ,_2\ \vdash\ E_2}{,_1, ,_2\ \vdash\ E_1 \wedge E_2}$$

and so on.

Also in Section 1.3, it was difficult to decide how much to labour the point about not substituting for arbitrary expressions. The less able students never try to violate the restriction and it is only the smart ones who spot the counter-examples which follow from the substitution into negative contexts. An example of an *invalid* argument is to notice that:

$$\boxed{\text{danger}}\ \frac{E_1 \wedge \neg E_1 \wedge E_2}{E_1 \wedge \neg E_1}$$

is valid ($E_1 \wedge \neg E_1 \wedge E_2\ \vdash\ E_1 \wedge \neg E_1$ by $\wedge$-E; the reverse by $\wedge$-E then *contr*). Then to use it *invalidly* to show

from	$E_1 \wedge \neg E_1 \wedge E_2\ \Rightarrow\ E_2$	
1	$E_1 \wedge \neg E_1\ \Rightarrow\ E_2$	error!!
3	$\neg(\neg E_1 \vee \neg E_2)$	$\neg \vee$-I(1,2)
infer	$E_1 \vee \neg E_1 \vee E_2$	

Error

which is invalid when $E_1 = u, E_2 =$ **false**. The problem comes from the illegal substitution in a 'negative' position.

1.2 Answers

Answer 1.1.1 from page 5

$E \wedge \mathbf{true}$	E
$E \wedge \mathbf{false}$	$\mathbf{false}$
$\mathbf{false} \Rightarrow E$	$\mathbf{true}$
$E \Rightarrow \mathbf{true}$	$\mathbf{true}$
$\mathbf{true} \Rightarrow E$	E
$E \Rightarrow \mathbf{false}$	$\neg E$
$E \vee \mathbf{false}$	E
$E \vee \mathbf{true}$	$\mathbf{true}$

Answer 1.1.2 from page 5

$E_1 \wedge E_2$	$E_2 \wedge E_1$
$E_1 \wedge (E_2 \wedge E_3)$	$(E_1 \wedge E_2) \wedge E_3$
$E_1 \wedge (E_2 \vee E_3)$	$E_1 \wedge E_2 \vee E_1 \wedge E_3$
$\neg (E_1 \vee E_2)$	$\neg E_1 \wedge \neg E_2$
$\neg \neg E$	E
$E_1 \Rightarrow E_2$	$\neg E_2 \Rightarrow \neg E_1$
$E_1 \Leftrightarrow E_2$	$(E_1 \Rightarrow E_2) \wedge (E_2 \Rightarrow E_1)$
$E_1 \vee E_2$	$E_2 \vee E_1$
$E_1 \vee (E_2 \vee E_3)$	$(E_1 \vee E_2) \vee E_3$
$E_1 \vee (E_2 \wedge E_3)$	$(E_1 \vee E_2) \wedge (E_1 \vee E_3)$

The third case needs no parenthesis because of the priority of the operators.

Answer 1.1.3 from page 6

$E_1 \wedge E_2$	**if** E_1 **then** E_2 **else false**
$E_1 \vee E_2$	**if** E_1 **then true else** E_2
$\neg E$	**if** E **then false else true**
$E_1 \Rightarrow E_2$	**if** E_1 **then** E_2 **else true**
$E_1 \Leftrightarrow E_2$	**if** E_1 **then** E_2 **else** (**if** E_2 **then false else true**)

Answer 1.1.4 from page 8

$E_1 \vee E_2 \vdash E_1$	no
$E_1, E_2 \vdash E_1$	yes
$E_1 \wedge E_2 \vdash E_1 \vee E_2$	yes
$E_1 \vee E_2 \vdash E_1 \wedge E_2$	no
$E_2 \vdash E_1 \Rightarrow E_2$	yes
$\neg E_1 \vdash E_1 \Rightarrow E_2$	yes
$E_1 \Rightarrow E_2, E_1 \vdash E_2$	yes
$\neg E_1 \vdash \neg (E_1 \wedge E_2)$	yes
$\neg E_1 \vdash \neg (E_1 \vee E_2)$	no
$E_1 \wedge (E_2 \Leftrightarrow E_3) \vdash E_1 \wedge E_2 \Leftrightarrow E_1 \wedge E_3$	yes
$E_1 \wedge E_2 \Leftrightarrow E_1 \wedge E_3 \vdash E_1 \wedge (E_2 \Leftrightarrow E_3)$	no

Answer 1.1.5 from page 8 Writing $\oplus$ for 'exclusive or':

E_1	E_2	$E_1 \oplus E_2$
true	**true**	**false**
true	**false**	**true**
false	**true**	**true**
false	**false**	**false**

$E_1 \oplus E_2 \vdash \neg (E_1 \Leftrightarrow E_2)$
$\neg (E_1 \oplus E_2) \vdash E_1 \Leftrightarrow E_2$
$E_1 \oplus E_2 \vdash E_1 \vee E_2$
$E_1 \wedge E_2 \vdash \neg (E_1 \oplus E_2)$
$(E_1 \wedge E_2) \oplus (E_1 \wedge E_3) \vdash E_1 \wedge (E_2 \oplus E_3)$
$(E_1 \vee E_2) \oplus (E_1 \vee E_3) \vdash E_1 \vee (E_2 \oplus E_3)$

from $E_1 \vee (E_2 \vee E_3)$
1 $(E_2 \vee E_3) \vee E_1$ $\vee$-*comm*(h)
2 $E_2 \vee (E_3 \vee E_1)$ $\vee$-*ass*(1)
3 $(E_3 \vee E_1) \vee E_2$ $\vee$-*comm*(2)
4 $E_3 \vee (E_1 \vee E_2)$ $\vee$-*ass*(3)
infer $(E_1 \vee E_2) \vee E_3$ $\vee$-*comm*(4)

Answer 1.3.1 from page 19

from $E_1 \wedge E_2$
1 $\neg (\neg E_1 \vee \neg E_2)$ $\wedge$-*defn*(h)
2 $\neg \neg E_i$ $\neg \vee$-E(1)
infer E_i $\neg \neg$-E(2)

for $1 \leq i \leq 2$

Answer 1.3.2 from page 21

from $\neg E_i$
1 $\neg E_1 \vee \neg E_2$ $\vee$-I(h)
2 $\neg \neg (\neg E_1 \vee \neg E_2)$ $\neg \neg$-I(1)
infer $\neg (E_1 \wedge E_2)$ $\wedge$-*defn*(2)

for $1 \leq$ i ≤ 2

Answer 1.3.3 from page 21

from $(E_1 \vee E_2) \wedge (E_1 \vee E_3)$		
1	$E_1 \vee E_2$	$\wedge$-E(h)
2	$E_1 \vee E_3$	$\wedge$-E(h)
3	**from** E_1	
	infer $E_1 \vee E_2 \wedge E_3$	$\vee$-I(h3)
4	**from** E_2	
4.1	**from** E_3	
4.1.1	$E_2 \wedge E_3$	$\wedge$-I(h4,h4.1)
	infer $E_1 \vee E_2 \wedge E_3$	$\vee$-I(4.1.1)
	infer $E_1 \vee E_2 \wedge E_3$	$\vee$-E(2,3,4.1)
infer $E_1 \vee E_2 \wedge E_3$		$\vee$-E(1,3,4)

Answer 1.3.4 from page 23

from $E_1 \wedge (E_2 \vee E_3)$		
1	E_1	$\wedge$-E(h)
2	$E_2 \vee E_3$	$\wedge$-E(h)
3	**from** E_2	
3.1	$E_1 \wedge E_2$	$\wedge$-I(1,h3)
	infer $E_1 \wedge E_2 \vee E_1 \wedge E_3$	$\vee$-I(3.1)
4	**from** E_3	
4.1	$E_1 \wedge E_3$	$\wedge$-I(1,h4)
	infer $E_1 \wedge E_2 \vee E_1 \wedge E_3$	$\vee$-I(4.1)
infer $E_1 \wedge E_2 \vee E_1 \wedge E_3$		$\vee$-E(2,3,4)

Answer 1.3.5 from page 23

from $E_1 \wedge E_2 \vee E_1 \wedge E_3$		
1	**from** $E_1 \wedge E_2$	
	infer $E_1 \wedge (E_2 \vee E_3)$	$\wedge$-$subs$($\vee$-I)(h1)
2	**from** $E_1 \wedge E_3$	
	infer $E_1 \wedge (E_2 \vee E_3)$	$\wedge$-$subs$($\vee$-I)(h2)
infer $E_1 \wedge (E_2 \vee E_3)$		$\vee$-E(h,1,2)

Answer 1.3.6 from page 23

from $\neg(E_1 \vee E_2)$
1 $\quad \neg E_1$ $\quad \neg\vee$-E(h)
2 $\quad \neg E_2$ $\quad \neg\vee$-E(h)
infer $\neg E_1 \wedge \neg E_2$ $\quad \wedge$-I(1,2)

from $\neg E_1 \wedge \neg E_2$
1 $\quad \neg E_1$ $\quad \wedge$-E(h)
2 $\quad \neg E_2$ $\quad \wedge$-E(h)
infer $\neg(E_1 \vee E_2)$ $\quad \neg\vee$-I(1,2)

from $\neg(E_1 \wedge E_2)$
1 $\quad \neg\neg(\neg E_1 \vee \neg E_2)$ $\quad \wedge$-*defn*(h)
infer $\neg E_1 \vee \neg E_2$ $\quad \neg\neg$-E(1)

from $\neg E_1 \vee \neg E_2$
1 $\quad \neg\neg(\neg E_1 \vee \neg E_2)$ $\quad \neg\neg$-I(h)
infer $\neg(E_1 \wedge E_2)$ $\quad \wedge$-*defn*(1)

Answer 1.3.7 from page 24

from $\neg E_1$
1 $\quad \neg E_1 \vee E_2$ $\quad \vee$-I(h)
infer $E_1 \Rightarrow E_2$ $\quad \Rightarrow$-*defn*(1)

from E_2
1 $\quad \neg E_1 \vee E_2$ $\quad \vee$-I(h)
infer $E_1 \Rightarrow E_2$ $\quad \Rightarrow$-*defn*(1)

Answer 1.3.8 from page 26

from $E_1 \Rightarrow E_2$
1 $\quad \neg E_1 \vee E_2$ $\quad \Rightarrow$-*defn*(h)
2 $\quad$ **from** $\neg E_1$
$\quad$ **infer** $\neg E_2 \Rightarrow \neg E_1$ $\quad \Rightarrow vac$-I(h2)
3 $\quad$ **from** E_2
3.1 $\quad\quad \neg\neg E_2$ $\quad \neg\neg$-I(h3)
$\quad$ **infer** $\neg E_2 \Rightarrow \neg E_1$ $\quad \Rightarrow vac$-I(3.1)
infer $\neg E_2 \Rightarrow \neg E_1$ $\quad \vee$-E(1,2,3)

Answer 1.3.9 from page 26

from	$E_1 \vee E_2 \;\Rightarrow\; E_3$	
1	$\neg (E_1 \vee E_2) \vee E_3$	$\Rightarrow$-*defn*(h)
2	$\neg E_1 \wedge \neg E_2 \vee E_3$	$\vee$-*subs*($\vee$-*deM*(1))
3	$(\neg E_1 \vee E_3) \wedge (\neg E_2 \vee E_3)$	$\wedge\vee$-*dist*(2)
4	**from** $\neg E_1 \vee E_3$	
	infer $E_1 \;\Rightarrow\; E_3$	$\Rightarrow$-*defn*(h4)
5	**from** $\neg E_2 \vee E_3$	
	infer $E_2 \;\Rightarrow\; E_3$	$\Rightarrow$-*defn*(h5)
infer	$(E_1 \;\Rightarrow\; E_3) \vee (E_1 \;\Rightarrow\; E_3)$	$\wedge$-*subs*(3,4,5)

Answer 1.3.10 from page 27

from	$E_1 \wedge E_2$	
1	E_1	$\wedge$-*E*(h)
2	E_2	$\wedge$-*E*(h)
3	$E_1 \;\Rightarrow\; E_2$	$\Rightarrow$ *vac-I*(2)
4	$E_2 \;\Rightarrow\; E_1$	$\Rightarrow$ *vac-I*(1)
5	$(E_1 \;\Rightarrow\; E_2) \wedge (E_2 \;\Rightarrow\; E_1)$	$\wedge$-*I*(3,4)
infer	$E_1 \;\Leftrightarrow\; E_2$	$\Leftrightarrow$-*defn*(5)

Answer 1.3.11 from page 27 a

Other results for the introduction of $\Leftrightarrow$ and its negation can be proved in a similar way.

from	$E_1 \Leftrightarrow E_2$	
1	$(E_1 \Rightarrow E_2) \wedge (E_2 \Rightarrow E_1)$	$\Leftrightarrow$-*defn*(h)
2	$E_1 \Rightarrow E_2$	$\wedge$-E(1)
3	$E_2 \Rightarrow E_1$	$\wedge$-E(1)
4	$\neg E_1 \vee E_2$	$\Rightarrow$-*defn*(2)
5	**from** $\neg E_1$	
5.1	$\quad \neg E_2$	$\Rightarrow$ *vac*-E(3,h5)
5.2	$\quad \neg E_1 \wedge \neg E_2$	$\wedge$-I(h5,5.1)
	infer $E_1 \wedge E_2 \vee \neg E_1 \wedge \neg E_2$	$\vee$-I(5.2)
6	**from** E_2	
6.1	$\quad E_1$	$\Rightarrow$-E(3,h6)
6.2	$\quad E_1 \wedge E_2$	$\wedge$-I(h6,6.1)
	infer $E_1 \wedge E_2 \vee \neg E_1 \wedge \neg E_2$	$\vee$-I(6.2)
infer	$E_1 \wedge E_2 \vee \neg E_1 \wedge \neg E_2$	$\vee$-E(4,5,6)

from	$E_1 \wedge E_2 \vee \neg E_1 \wedge \neg E_2$	
1	**from** $E_1 \wedge E_2$	
	infer $E_1 \Leftrightarrow E_2$	$\Leftrightarrow$-I(h1)
2	**from** $\neg E_1 \wedge \neg E_2$	
	infer $E_1 \Leftrightarrow E_2$	$\Leftrightarrow$-I(h2)
infer	$E_1 \Leftrightarrow E_2$	$\vee$-E(h,1,2)

Answer 1.3.11 from page 27 c

from $E_1 \wedge (E_2 \;\Leftrightarrow\; E_3)$		
1	E_1	$\wedge$-E(h)
2	$E_2 \;\Leftrightarrow\; E_3$	$\wedge$-E(h)
3	$E_2 \wedge E_3 \vee \neg E_2 \wedge \neg E_3$	$\Leftrightarrow$-E(2)
4	**from** $E_2 \wedge E_3$	
4.1	E_2	$\wedge$-E(h4)
4.2	E_3	$\wedge$-E(h4)
4.3	$E_1 \wedge E_2 \wedge E_1 \wedge E_3$	$\wedge$-I(1,4.1,1,4.2)
	infer $(E_1 \wedge E_2) \;\Leftrightarrow\; (E_1 \wedge E_3)$	$\Leftrightarrow$-I(4.3)
5	**from** $\neg E_2 \wedge \neg E_3$	
5.1	$\neg E_2$	$\wedge$-E(h5)
5.2	$\neg E_3$	$\wedge$-E(h5)
5.3	$\neg E_1 \vee \neg E_2$	$\vee$-I(5.1)
5.4	$\neg E_1 \vee \neg E_3$	$\vee$-I(5.2)
5.5	$\neg (E_1 \wedge E_2)$	deM(5.3)
5.6	$\neg (E_1 \wedge E_3)$	deM(5.4)
5.7	$\neg (E_1 \wedge E_2) \wedge \neg (E_1 \wedge E_3)$	$\wedge$-I(5.5,5.6)
	infer $(E_1 \wedge E_2) \;\Leftrightarrow\; (E_1 \wedge E_3)$	$\Leftrightarrow$-I(5.7)
infer $(E_1 \wedge E_2) \;\Leftrightarrow\; (E_1 \wedge E_3)$		$\vee$-E(3,4,5)

Answer 1.3.11 from page 27 d

The converse of $\wedge\Leftrightarrow$-*dist* is false! Consider $E_1 = f, E_2 = E_3 = t$.

from $E_1 \vee E_2 \;\Leftrightarrow\; E_1 \vee E_3$		
1	$(E_1 \vee E_2) \wedge (E_1 \vee E_3) \vee \neg (E_1 \vee E_2) \wedge \neg (E_1 \vee E_3)$	$\Leftrightarrow$-E(h)
2	**from** $(E_1 \vee E_2) \wedge (E_1 \vee E_3)$	
2.1	$E_1 \vee E_2 \wedge E_3$	$\wedge\vee$-*dist*(h2)
	infer $E_1 \vee (E_2 \;\Leftrightarrow\; E_3)$	$\vee$-*subs*($\Leftrightarrow$-I)(2.1)
3	**from** $\neg (E_1 \vee E_2) \wedge \neg (E_1 \vee E_3)$	
3.1	$\neg E_1 \wedge \neg E_2 \wedge \neg E_1 \wedge \neg E_3$	$\wedge$-*subs*(deM)(h3)
3.2	$\neg E_2 \wedge \neg E_3$	$\wedge$-E($\wedge$-E(3.1))
3.3	$E_2 \;\Leftrightarrow\; E_3$	$\Leftrightarrow$-I(3.2)
	infer $E_1 \vee (E_2 \;\Leftrightarrow\; E_3)$	$\vee$-I(3.3)
infer $E_1 \vee (E_2 \;\Leftrightarrow\; E_3)$		$\vee$-E(1,2,3)

Answer 1.3.11 from page 27 e

2

Reasoning about Predicates

2.1 Comments

In Section 2.2, BSH has pointed out that the results about quantifying over empty sets follow naturally from the expansions $\exists x \in X \cdot p(x)$ as $\exists x \cdot x \in X \wedge p(x)$; $\forall x \in X \cdot p(x)$ as $\forall x \cdot x \in X \Rightarrow p(x)$. Although this is a useful point, I am reluctant to use the unbounded form of the quantifiers at this stage of the book.

2.2 Answers

Answer 2.1.1 from page 32

$$is\text{-}hexable(i) \quad \triangleq \quad 0 \leq i \leq 15$$

Answer 2.1.2 from page 33

$$is\text{-}leapyr(i) \quad \triangleq \quad 4 \textbf{ divides } i \wedge \neg (100 \textbf{ divides } i) \vee 400 \textbf{ divides } i$$

Answer 2.1.3 from page 33

$$is\text{-}common\text{-}multiple(i, j, m) \quad \triangleq \quad i \textbf{ divides } m \wedge j \textbf{ divides } m$$

Answer 2.1.4 from page 33

$$post\text{-}sqrt(i, r) \quad \triangleq \quad r * r = i$$

Answer 2.1.5 from page 33

$$post\text{-}idiv(i, j, q, r) \quad \triangleq \quad j * q + r = i \wedge r < j$$

Answer 2.2.1 from page 38

$\exists i \in \mathbb{N} \cdot i = i$	true
$\forall i \in \mathbb{N} \cdot i = i$	true
$\exists i \in \mathbb{N} \cdot i \neq i$	false
$\exists i, j \in \mathbb{N}_1 \cdot i \textbf{ mod } j \geq j$	false
$\forall i \in \mathbb{Z} \cdot \exists j \in \mathbb{Z} \cdot i + j = 0$	true
$\exists j \in \mathbb{Z} \cdot \forall i \in \mathbb{Z} \cdot i + j = 0$	false
$\forall i, j \in \mathbb{N} \cdot i \neq j$	false
$\forall i \in \mathbb{N} \cdot \exists j \in \mathbb{N} \cdot j = i \Leftrightarrow 1$	false(0)
$\forall i \in \mathbb{N} \cdot \exists j \in \mathbb{N} \cdot i < j < 2 * i \wedge is\text{-}odd(j)$	false(1)
$\forall i \in \mathbb{N}_1 \cdot \neg\, is\text{-}prime(4 * i)$	true
$\forall i \in \mathbb{N} \cdot \exists j \in \mathbb{N} \cdot j \leq 3 \wedge is\text{-}leapyr(i + j)$	false(1897)
$\exists!\, i \in \mathbb{N} \cdot i = i$	false
$\exists!\, i \in \mathbb{Z} \cdot i * i = i$	false

Answer 2.2.2 from page 39

$$\forall i \in \mathbb{N} \cdot \exists j \in \mathbb{N} \cdot j > i$$
$$\neg\, \exists i \in \mathbb{N} \cdot \forall j \in \mathbb{N} \cdot j \leq i$$

Answer 2.2.3 from page 39

$$\textit{is-ge}(i, j) \quad \triangleq \quad \exists k \in \mathbb{N} \cdot i = j + k$$

Answer 2.2.4 from page 39

$$\textit{sign}(i) \quad \triangleq \quad \textbf{if } i < 0 \textbf{ then } \Leftrightarrow 1 \textbf{ else if } i = 0 \textbf{ then } 0 \textbf{ else } +1$$

$$\begin{array}{l} \forall i, j \in \mathbb{Z} \cdot \\ \qquad \textit{sign}(i) * i \geq 0 \wedge \\ \qquad (\textit{sign}(i) = 0 \;\Leftrightarrow\; i = 0) \wedge \\ \qquad \textit{sign}(i * j) = \textit{sign}(i) * \textit{sign}(j) \end{array}$$

Answer 2.2.5 from page 39

$$\begin{array}{l} i \textbf{ mod } j = r \;\Leftrightarrow \\ \quad \textit{abs}(r) < \textit{abs}(j) \wedge \exists m \in \mathbb{Z} \cdot m * j + r = i \wedge \textit{sign}(r) = \textit{sign}(i) \end{array}$$

from	$s \in X, \neg\, p(s/x)$	
1	$\exists x \in X \cdot \neg\, p(x)$	$\exists$-I(h)
2	$\neg\,\neg\, \exists x \in X \cdot \neg\, p(x)$	$\neg\,\neg$-I(1)
infer	$\neg\, \forall x \in X \cdot p(x)$	$\forall$-*defn*(2)

Answer 2.3.1 from page 44 a

from	$\neg\, \forall x \in X \cdot p(x);\; y \in X, \neg\, p(y/x) \vdash E$	
1	$\neg\,\neg\, \exists x \in X \cdot \neg\, p(x)$	$\forall$-*defn*(h0)
2	$\exists x \in X \cdot \neg\, p(x)$	$\neg\,\neg$-E(1)
infer	E	$\exists$-E(2,h)

Answer 2.3.1 from page 44 b

from $\exists x \in X \cdot E_1(x) \vee E_2(x)$
1 $\quad y \in X, y \notin (fE_1(x) \cup fE_2(x)))$ $\quad$ var-I
2 $\quad$ **from** $E_1(y/x) \vee E_2(y/x)$
2.1 $\quad\quad$ **from** $E_1(y/x)$
2.1.1 $\quad\quad\quad \exists x \in X \cdot E_1(x)$ $\quad \exists$-I(1,h2.1)
$\quad\quad$ **infer** $(\exists x \in X \cdot E_1(x)) \vee (\exists x \in X \cdot E_2(x))$ $\quad \vee$-I(2.1.1)
2.2 $\quad\quad$ **from** $E_2(y/x)$
2.2.1 $\quad\quad\quad \exists x \in X \cdot E_2(x)$ $\quad \exists$-I(1,h2.2)
$\quad\quad$ **infer** $(\exists x \in X \cdot E_1(x)) \vee (\exists x \in X \cdot E_2(x))$ $\quad \vee$-I(2.2.1)
$\quad$ **infer** $(\exists x \in X \cdot E_1(x)) \vee (\exists x \in X \cdot E_2(x))$ $\quad \vee$-E(h2,2.1,2.2)
infer $(\exists x \in X \cdot E_1(x)) \vee (\exists x \in X \cdot E_2(x))$ $\quad \exists$-E(h,1.2)

from $(\exists x \in X \cdot E_1(x)) \vee (\exists x \in X \cdot E_2(x))$
1 $\quad$ **from** $\exists x \in X \cdot E_1(x)$
1.1 $\quad\quad y \in X, y \notin (fE_1(x) \cup fE_2(x)$ $\quad$ var-I
1.2 $\quad\quad$ **from** $E_1(y/x)$
1.2.1 $\quad\quad\quad E_1(y/x) \vee E_2(y/x)$ $\quad \vee$-I(h1.2)
$\quad\quad$ **infer** $\exists x \in X \cdot E_1(x) \vee E_2(x)$ $\quad \exists$-I(1.1,1.2.1)
$\quad$ **infer** $\exists x \in X \cdot E_1(x) \vee E_2(x)$ $\quad \exists$-E(h1,1.1,1.2)
2 $\quad$ **from** $\exists x \in X \cdot E_2(x)$
$\quad\quad$ similar
$\quad$ **infer** $\exists x \in X \cdot E_1(x) \vee E_2(x)$
infer $\exists x \in X \cdot E_1(x) \vee E_2(x)$ $\quad \vee$-E(h,1,2)

Answer 2.3.2 from page 44 a

from $\exists x \in X \cdot p(x) \wedge E_2(x)$
1 $\quad$ **from** $y \in X, E_1(y/x) \wedge E_2(y/z)$
1.1 $\quad\quad E_1(y/z)$ $\quad \wedge$-E(h1)
1.2 $\quad\quad E_2(y/z)$ $\quad \wedge$-E(h1)
1.3 $\quad\quad \exists x \in X \cdot E_1(x)$ $\quad \exists$-I(h1,1.1)
1.4 $\quad\quad \exists x \in X \cdot E_2(x)$ $\quad \exists$-I(h1,1.2)
$\quad$ **infer** $(\exists x \in X \cdot E_1(x)) \wedge (\exists x \in X \cdot E_2(x))$ $\quad \wedge$-I(1.3,1.4)
infer $(\exists x \in X \cdot E_1(x)) \wedge (\exists x \in X \cdot E_2(x))$ $\quad \exists$-E(h,1)

Answer 2.3.2 from page 44 b

from $(\forall x \in X \cdot E_1(x)) \vee (\forall x \in X \cdot E_2(x))$		
1	**from** $\forall x \in X \cdot E_1(x)$	
1.1	**from** $x \in X$	
1.1.1	$E_1(x)$	$\forall$-E(1,h1.1)
	infer $E_1(x) \vee E_2(x)$	$\vee$-I(1.1.1)
	infer $\forall x \in X \cdot E_1(x) \vee E_2(x)$	$\forall$-I(1.1
2	**from** $\forall x \in X \cdot E_2(x)$	
	similar	
	infer $\forall x \in X \cdot E_1(x) \vee E_2(x)$	
infer $\forall x \in X \cdot E_1(x) \vee E_2(x)$		$\vee$-E(h,1,2)

Answer 2.3.2 from page 44 c

from $(\forall x \in X \cdot E_1(x)) \wedge (\forall x \in X \cdot E_2(x))$		
1	$\forall x \in X \cdot E_1(x)$	$\wedge$-E(h)
2	$\forall x \in X \cdot E_2(x)$	$\wedge$-E(h)
3	**from** $x \in X$	
3.1	$E_1(x)$	$\forall$-E(1)
3.2	$E_2(x)$	$\forall$-E(2)
	infer $E_1(x) \wedge E_2(x)$	$\wedge$-I(3.1,3.2)
infer $\forall x \in X \cdot E_1(x) \wedge E_2(x)$		$\forall$-I(3)

from $\forall x \in X \cdot E_1(x) \wedge E_2(x)$		
1	**from** $x \in X$	
1.1	$E_1(x) \wedge E_2(x)$	$\forall$-E(h,h1)
	infer $E_1(x)$	$\wedge$-E(1.1)
2	$\forall x \in X \cdot E_1(x)$	$\forall$-I(1)
3	**from** $x \in X$	
3.1	$E_1(x) \wedge E_2(x)$	$\forall$-E(h,h3)
	infer $E_2(x)$	$\wedge$-E(3.1)
4	$\forall x \in X \cdot E_2(x)$	$\forall$-I(3)
infer $(\forall x \in X \cdot E_1(x)) \wedge (\forall x \in X \cdot E_2(x))$		$\wedge$-I(2,4)

Answer 2.3.2 from page 44 d

3

Functions and Operations

3.1 Comments

In Section 3.2, one of the major changes since the first edition is the avoidance here of rules which explicitly manipulate the definitions of functions. The technique used now of generating inference rules is far less cumbersome in the proofs.

In Section 3.2, the discussion of under-determined functions could be extended. In particular, one could argue that determinism of the final implementation should be covered in the proof obligation. Since, however, the notation presented does not really permit the creation of other than deterministic functions, the issue is left.

In Section 3.4 the VDM convention of using keywords for emphasizing the structure of a specification is followed in this book. There are obviously many alternatives. In the specification language called 'Z' (see [Hay87, WL88, Spi88]), keywords are avoided. In work by Manna and Waldinger, the following style is employed:

$$\begin{aligned} f(a) \leftarrow\ & \text{find } r \text{ such that } \mathit{post}\text{-}f(a,r) \\ & \quad \text{where } \mathit{pre}\text{-}f(a) \end{aligned}$$

Basically, one is defining a set of pairs (the argument, result pairs of the function) and any syntax is acceptable if it clearly shows this (and, preferably, the distinction between pre- and post-conditions) without forcing the definition of an implementation.

In Section 3.4 the reasons for wanting the non-deterministic interpretation for operation specifications – even in the absence of concurrency – are explained in [HJ89].

In Section 3.3 The research area of proofs about partial functions is still very active. See [CJ90] for a review and references.

3.2 Answers

Answer 3.1.1 from page 50

$\mathit{mins}\ (s\colon \mathbb{Z}\text{-}\mathbf{set})\ r\colon \mathbb{Z}$
pre $s \neq \{\,\}$
post $r \in s \wedge \forall i \in s \cdot r \leq i$

Answer 3.1.2 from page 50

$\mathit{sub}\ (i\colon \mathbb{Z}, j\colon \mathbb{Z})\ r\colon \mathbb{Z}$
post $i = j + r$

Answer 3.1.3 from page 50

$\mathit{subp}\ (i\colon \mathbb{N}_1, j\colon \mathbb{N}_1)\ r\colon \mathbb{N}_1$
pre $i > j$
post $i = j + r$

Answer 3.1.4 from page 50

$$\begin{array}{l} abs\ (i{:}\,\mathbb{Z})\ r{:}\,\mathbb{N} \\ \textbf{post}\ \ r = i \lor r = \Leftrightarrow i \end{array}$$

Answer 3.1.5 from page 50

$$m\ \textbf{is mult of}\ i \quad \triangleq \quad \exists n \in \mathbb{N} \cdot n * i = m$$

$$\textit{is-common-mult}(i, j, m) \quad \triangleq \quad m\ \textbf{is mult of}\ i \land m\ \textbf{is mult of}\ j$$

$$\begin{array}{l} scm\ (i{:}\,\mathbb{N}_1, j{:}\,\mathbb{N}_1)\ m{:}\,\mathbb{N} \\ \textbf{post}\ \ \textit{is-common-mult}(i, j, m) \land \\ \qquad \neg\, \exists k \in \mathbb{N}_1 \cdot k < m \land \textit{is-common-mult}(i, j, k) \end{array}$$

Answer 3.1.6 from page 50

$$\begin{array}{l} mod\ (i{:}\,\mathbb{N}, j{:}\,\mathbb{N}_1)\ r{:}\,\mathbb{N} \\ \textbf{post}\ \ r < j \land \exists m \in \mathbb{N} \cdot m * j + r = i \end{array}$$

Answer 3.1.7 from page 50

- $sub(4, 1) \neq 8$
- '*subp*' correct over positive integers ($\mathbb{N}$), not for negative results.
- '*abs*' correct, since '*max*' is defined over $\mathbb{Z}$.
- $scm(4, 6) \neq 24$

Answer 3.2.1 from page 57 To show that:

$$\forall y \in \mathbb{Z} \cdot double(y) \in \mathbb{Z} \land \textit{post-double}(y, double(y))$$

prove:

$$y \in \mathbb{Z} \vdash double(y) \in \mathbb{Z} \land \textit{post-double}(y, double(y))$$

and then use $\forall$-I. Use the rule:

$$\boxed{\mathrm{R}double}\ \frac{x \in \mathbb{Z}}{double(x) = x + x}$$

from	$y \in \mathbb{Z}$	
1	$double(y) = y + y$	R*double*(h)
2	$y + y \in \mathbb{Z}$	$\mathbb{Z}$, h
3	$double(y) \in \mathbb{Z}$	$=$-*subs*(1,2)
4	$double(y) = 2 * y$	$\mathbb{Z}$,$=$-*subs*(1)
5	$\textit{post-double}(y, double(y))$	*post-double*(h,3,4)
infer	$double(y) \in \mathbb{Z} \land \textit{post-double}(y, double(y))$	$\land$-I(3,5)

Answer 3.2.1 from page 57 (sequent form)

from $f \in \mathbb{R}$		
1	$conv(f) = (f + 40) * 5/9 \Leftrightarrow 40$	R$conv$(h)
2	$(f + 40) * 5/9 \Leftrightarrow 40 \in \mathbb{R}$	$\mathbb{R}$, h
3	$conv(f) \in \mathbb{R}$	$=$-$subs$(1,2)
4	$((f + 40) * 5/9 \Leftrightarrow 40) * 9/5 + 32 = f$	$\mathbb{R}$, h
5	$post\text{-}conv(f, conv(f))$	$post\text{-}conv$(h,3,1,4)
infer	$conv(f) \in \mathbb{R} \wedge post\text{-}conv(f, conv(f))$	$\wedge$-I(3,5)

Answer 3.2.2 from page 57

from $i \in \mathbb{N}$		
1	**from** $pre\text{-}choose(i)$	
1.1	$i = 3 \vee i = 8$	$pre\text{-}choose$(h1)
1.2	$11 \Leftrightarrow i \in \mathbb{N}$	$\mathbb{N}$, 1.1
1.3	$choose(i) = 11 \Leftrightarrow i$	R$choose$(h)
1.4	$choose(i) \in \mathbb{N}$	$eqsubs$(1.3,1.2)
1.5	$(i = 3 \;\Rightarrow\; (11 \Leftrightarrow i) = 8) \wedge (i = 8 \;\Rightarrow\; (11 \Leftrightarrow i) = 3)$	$\mathbb{N}$, 1
1.6	$post\text{-}choose(i, choose(i))$	$post\text{-}choose$(h,1.4,1.5)
	infer $choose(i) \in \mathbb{N} \wedge post\text{-}choose(i, choose(i))$	$\wedge$-I(1.3,1.6)
2	$\delta(pre\text{-}choose(i))$	$pre\text{-}choose$(h)
infer	$pre\text{-}choose(i) \;\Rightarrow\; choose(i) \in \mathbb{N} \wedge post\text{-}choose(i, choose(i))$	$\Rightarrow$-I(2,1)

Answer 3.2.3 from page 57

from $i \in \mathbb{Z}$		
1	$i < 0 \vee i \geq 0$	$\mathbb{Z}$, h
2	**from** $i < 0$	
2.1	$abs(i) = \Leftrightarrow i$	Rabs(h, h1)
2.2	$abs(i) \in \mathbb{Z}$	$eqsubs$(Int,2.1)
2.3	$0 \leq \Leftrightarrow i \wedge (\Leftrightarrow i = i \vee \Leftrightarrow i = \Leftrightarrow i)$	$\mathbb{Z}$, h2, h
2.4	$post\text{-}abs(i, abs(i))$	$post\text{-}abs$(h,2.2,2.3)
	infer $abs(i) \in \mathbb{Z} \wedge post\text{-}abs(i, abs(i))$	$\wedge$-I(2.2,2.4)
3	**from** $i \geq 0$	
	similar	
	infer $abs(i) \in \mathbb{Z} \wedge post\text{-}abs(i, abs(i))$	
infer	$abs(i) \in \mathbb{Z} \wedge post\text{-}abs(i, abs(i))$	$\vee$-E(1,2,3)

Answer 3.2.4 from page 59

Answer 3.2.5 from page 59

$$sign(i) \;\triangleq\; \textbf{if } i < 0 \textbf{ then } \Leftrightarrow 1 \textbf{ else if } i = 0 \textbf{ then } 0 \textbf{ else } +1$$

from	$i \in \mathbb{Z}$	
1	$0, \Leftrightarrow 1, 1 \in \mathbb{Z}$	$\mathbb{Z}$
2	$sign(i) \in \mathbb{Z}$	R$sign$(h,1)
3	$i < 0 \vee i = 0 \vee i > 0$	$\mathbb{Z}$, h
4	**from** $i < 0$	
4.1	$sign(i) = \Leftrightarrow 1$	R$sign$, h4
	infer $post\text{-}sign(i, sign(i))$	$post\text{-}sign$(h,2,4.1)
5	**from** $i = 0$	
	similar	
	infer $post\text{-}sign(i, sign(i))$	
6	**from** $i > 0$	
	similar	
	infer $post\text{-}sign(i, sign(i))$	
	$post\text{-}sign(i, sign(i))$	$\vee$-E(3,4,5,6)
infer	$sign(i) \in \mathbb{Z} \wedge post\text{-}sign(i, sign(i))$	$\wedge$-I(2,7)

Answer 3.2.5 from page 59

Notice that the commoning up of the range check is only shown for interest; it would be quite correct to handle it under each of the three cases.

from	$i, j \in \mathbb{Z}$	
1	$i \geq 0 \vee i < 0$	$\mathbb{Z}$, h
2	**from** $i \geq 0$	
2.1	$multp(i, j) \in \mathbb{Z}$	$multp$(h)
2.2	$multp(\Leftrightarrow i, \Leftrightarrow j) \in \mathbb{Z}$	$multp$(h)
2.3	$mult(i, j) \in \mathbb{Z}$	R$mult$(h,2.1,2.2)
2.4	$multp(i, j) = i * j$	$multp$(h)
2.5	$post\text{-}mult(i, j, multp(i, j))$	$post\text{-}mult$(h,2.1,2.4)
	infer $mult(i, j) \in \mathbb{Z} \wedge post\text{-}mult(i, j, mult(i, j))$	$\wedge$-I(2.3,2.5)
3	**from** $i < 0$	
	similar	
	infer	
infer	$mult(i, j) \in \mathbb{Z} \wedge post\text{-}mult(i, j, mult(i, j))$	$\vee$-E(1,2,3)

Answer 3.2.6 from page 60

from $i, j \in \mathbb{N}$		
1	$add(0, j) = j$	R3.12(h)
2	$add(0, j) \in \mathbb{N}$	*eqsubs*(1,h)
3	$j = 0 + j$	$\mathbb{N}$, h
4	$post\text{-}add(0, j, add(0, j))$	*post-add*(h,1,3)
5	$add(0, j) \in \mathbb{N} \wedge post\text{-}add(0, j, add(0, j))$	$\wedge$-I(2,4)
6	**from** $n \in \mathbb{N}_1, add(n \Leftrightarrow 1, j) \in \mathbb{N}, post\text{-}add(n \Leftrightarrow 1, j, add(n \Leftrightarrow 1, j))$	
6.1	$add(n \Leftrightarrow 1, j) = n \Leftrightarrow 1 + j$	*post-add*(h6)
6.2	$add(n, j) = n + j$	R3.13(h6,h,6.1)
6.3	$add(n, j) \in \mathbb{N}$	$\mathbb{N}$, 6.2
6.4	$post\text{-}add(n, j, add(n, j))$	*post-add*(h6,h,6.2)
	infer $add(n, j) \in \mathbb{N} \wedge post\text{-}add(n, j, add(n, j))$	$\wedge$-I(6.3,6.4)
infer $add(i, j) \in \mathbb{N} \wedge post\text{-}add(i, j, add(i, j))$		$\mathbb{N}$-*ind*(5,6)

Answer 3.2.7 from page 66

Answer 3.2.9 from page 66

$$multp(i, j) \quad \triangleq \quad \textbf{if } i = 0 \; \textbf{then } 0 \; \textbf{else } multp(i \Leftrightarrow 1, j) + j$$

Notice that we have to handle the pre-condition in the induction of this proof.

Answer 3.4.1 from page 84

SUBJI
ext wr $m : \mathbb{Z}$,
rd $n : \mathbb{Z}$
post $m + n = \overset{\Leftrightarrow}{m}$

Answer 3.4.2 from page 84

SUMC
ext wr $m : \mathbb{N}$,
wr $n : \mathbb{N}$
pre $m > 0$
post $m + n = \overset{\Leftrightarrow}{m} + \overset{\Leftrightarrow}{n} \wedge m < \overset{\Leftrightarrow}{m}$

Answer 3.4.3 from page 84

The phrase 'restriction' is taken to allow ignoring negative numbers.

IDIV
ext wr $m : \mathbb{Z}$,
rd $n : \mathbb{Z}$,
wr $q : \mathbb{Z}$
pre $m \geq 0 \wedge n > 0$
post $q * m + n = \overset{\Leftrightarrow}{m} \wedge 0 \leq m < n$

IDIV $(n: \mathbb{Z})\ q: \mathbb{Z}$
ext wr $m : \mathbb{Z}$
pre $m \geq 0 \wedge n > 0$
post $q * m + n = \overset{\Leftrightarrow}{m} \wedge 0 \leq m < n$

Answer 3.4.4 from page 84

Use the same overall specification of FACT.

$$
\begin{array}{l}
\mathit{INIT} \\
\textbf{ext wr } fn \ : \ \mathbb{N}, \\
\qquad \textbf{wr } t \ \ : \ \mathbb{N} \\
\textbf{post } fn = 1 \wedge t = 0
\end{array}
$$

$$
\begin{array}{l}
\mathit{LOOP} \\
\textbf{ext rd } n \ \ : \ \mathbb{N}, \\
\qquad \textbf{wr } fn \ : \ \mathbb{N}, \\
\qquad \textbf{wr } t \ \ : \ \mathbb{N} \\
\textbf{pre } fn = t! \wedge t \leq n \\
\textbf{post } fn = n!
\end{array}
$$

$$
\begin{array}{l}
\mathit{BODY} \\
\textbf{ext wr } fn \ : \ \mathbb{N}, \\
\qquad \textbf{wr } t \ \ : \ \mathbb{N} \\
\textbf{pre } fn = t! \\
\textbf{post } fn = t! \wedge t > \overleftarrow{t}
\end{array}
$$

4

Set Notation

4.1 Comments

In Section 4.1, the more general form of set comprehension $\{f(i) \mid p(i)\}$ needs to be used with care. It is, however, interesting to note that f does *not* have to be total!

In Section 4.2, the presentation of *inter alia* set theory appears to lack (BSH is one of the people to have pointed this out) a 'no junk' rule which closes off the data type. For example, one could add:

$$\boxed{\textit{set-gen}}\ \frac{s \in X\text{-}\mathbf{set}}{s = \{\,\} \vee \exists e \in X, s' \in X\text{-}\mathbf{set} \cdot s = e \odot s'}$$

I have chosen to rely on the induction rules to achieve this sort of 'closing off' of types.

Other interesting properties which could be presented as sequents and set as exercises include:

$s \cap s = s$
$\cup, \cap$ distribute in both directions, left and right
$s \Leftrightarrow \{\,\} = s$
$s_1 \Leftrightarrow s_2 \subseteq s_1$
$s \Leftrightarrow s = \{\,\}$
$s_1 \Leftrightarrow (s_2 \cap s_3) = (s_1 \Leftrightarrow s_2) \cup (s_1 \Leftrightarrow s_3)$
$s_1 \Leftrightarrow (s_2 \cup s_3) = (s_1 \Leftrightarrow s_2) \cap (s_1 \Leftrightarrow s_3)$
$(s_1 \Leftrightarrow s_2) \cap s_3 = (s_1 \cap s_3) \Leftrightarrow s_2$
$s_1 \cup (s_1 \Leftrightarrow s_2) = s_1$
$s_1 \subseteq s_2 \;\;\Rightarrow\;\; s_1 \cup s_2 = s_1$
$s_1 \subseteq s_2 \;\;\Rightarrow\;\; s_1 \cap s_2 = s_2$
$s_1 \cup (s_2 \Leftrightarrow s_1) = s_1 \cup s_2$
$s_1 \cap (s_1 \Leftrightarrow s_2) = s_1 \Leftrightarrow s_2$
$s_1 \cap (s_2 \Leftrightarrow s_1) = \{\,\}$
$(s_1 \cap s_2) \subseteq s_1$
$s_1 \subseteq (s_1 \cup s_2)$

4.2 Answers

Answer 4.1.1 from page 92

$$\{a, c\} \cap \{c, d, a\} = \{a, c\}$$
$$\{a, c\} \Leftrightarrow \{c, d, a\} = \{\,\}$$
$$\mathbf{card}\{x^2 \mid x \in \{\Leftrightarrow 1, \ldots, +1\}\} = 2$$
$$5 \in \{3, \ldots, 7\} \;\Leftrightarrow\; \mathbf{true}$$
$$\{7, \ldots, 3\} = \{\,\}$$
$$\{i \in \mathbb{N} \mid i^2 \in \{4, 9\}\} = \{2, 3\}$$
$$\{i \in \mathbb{Z} \mid i^2 = i\} = \{0, 1\}$$
$$\bigcup\{\{a, b\}, \{\,\}, \{b, c\}, \{d\}\} = \{a, b, c, d\}$$
$$\bigcup\{\,\} = \{\,\}$$

Answer 4.1.2 from page 92

$$\{i \in \mathbb{Z} \mid 100 < i < 200 \wedge 9 \;\mathbf{divides}\; i\}$$
$$\{i \in \mathbb{Z} \mid 100 < i < 200 \wedge \mathit{is\text{-}prime}(i)\}$$
$$\mathbb{N}_1 \subset \mathbb{N} \subset \mathbb{Z}$$

Answer 4.1.3 from page 92

$$e \cup e = e$$
$$e \cap \{\,\} = \{\,\}$$
$$(e_1 \subseteq e_2) \;\Leftrightarrow\; (e_1 \Leftrightarrow e_2 = \{\,\})$$
$$e \cap e = e$$
$$e \cup \{\,\} = e$$
$$e_1 \subseteq e_2 \wedge e_2 \subseteq e_3 \;\Rightarrow\; e_1 \subseteq e_3$$
$$\{\,\} \subseteq e$$
$$\mathbf{card}\,(e_1 \cup e_2) = \mathbf{card}\; e_1 + \mathbf{card}\; e_2 \Leftrightarrow \mathbf{card}\,(e_1 \cap e_2)$$
$$(e_1 \Leftrightarrow e_2) \cap e_3 = (e_1 \cap e_3) \Leftrightarrow e_2$$
$$e_1 \Leftrightarrow (e_1 \Leftrightarrow e_2) = e_1 \cap e_2$$
$$\bigcup\{\bigcup es\} = \bigcup es$$

Answer 4.1.4 from page 92

$$e_1 \cap e_2 = e_2 \cap e_1$$
$$e_1 \cap (e_2 \cap e_3) = (e_1 \cap e_2) \cap e_3$$
$$e_1 \cap (e_2 \cup e_3) = (e_1 \cap e_2) \cup (e_1 \cap e_3)$$

Answer 4.1.5 from page 92

$$\mathit{is\text{-}disj} : X\textbf{-set} \times X\textbf{-set} \to \mathbb{B}$$
$$\mathit{is\text{-}disj}(s_1, s_2) \;\triangleq\; s_1 \cap s_2 = \{\,\}$$

Answer 4.1.6 from page 93

$$\bigcap \_ :\; X\textbf{-set} \times X\textbf{-set} \to X\textbf{-set}$$
$$\bigcap ss = \{e \in \bigcup ss \mid \forall s \in ss \cdot e \in s\}$$

Here, ss should be non-empty. But this restriction can be avoided by fixing some universe:

$$\bigcap ss = \{e \in X \mid \forall s \in ss \cdot e \in s\}$$

then:

$$\bigcap\{\,\} = X$$

Answer 4.1.7 from page 93

$$
\begin{aligned}
& s_1 \ominus s_2 = (s_1 \cup s_2) \Leftrightarrow (s_1 \cap s_2) \\
& s_1 \ominus s_2 = \{\,\} \;\;\Rightarrow\;\; s_1 = s_2 \\
& s_1 \ominus s_1 = \{\,\} \\
& s_1 \Leftrightarrow s_2 \subseteq s_1 \ominus s_2 \\
& s_1 \ominus s_2 = s_2 \ominus s_1 \\
& s_1 \ominus s_2 = (s_1 \Leftrightarrow s_2) \cup (s_2 \Leftrightarrow s_1) \\
& s_1 \ominus (s_1 \ominus s_2) \\
& \quad = s_1 \ominus ((s_1 \Leftrightarrow s_2) \cup (s_2 \Leftrightarrow s_1)) \\
& \quad = (s_1 \cup (s_2 \Leftrightarrow s_1)) \Leftrightarrow (s_1 \cap s_2) \cup (s_2 \Leftrightarrow s_1) \\
& \quad = (s_1 \cup (s_2 \Leftrightarrow s_1)) \Leftrightarrow (s_1 \cap (s_1 \Leftrightarrow s_2) \cup s_1 \cap (s_2 \Leftrightarrow s_1) \\
& \quad = (s_1 \cup s_2) \Leftrightarrow ((s_1 \Leftrightarrow s_2) \cup \{\,\}) \\
& \quad = (s_1 \cup s_2) \Leftrightarrow (s_1 \Leftrightarrow s_2) \\
& \quad = s_2
\end{aligned}
$$

Answer 4.2.1 from page 97

$$\boxed{\cap\text{-}b}\;\frac{s \in X\text{-}\mathbf{set}}{\{\,\} \cap s = \{\,\}}$$

$$\boxed{\cap\text{-}i}\;\frac{e \in X;\; s_1, s_2 \in X\text{-}\mathbf{set};\; e \in s_2}{(e \odot s_1) \cap s_2 = e \odot (s_1 \cap s_2)}$$

$$\boxed{\cap\text{-}i}\;\frac{e \in X;\; s_1, s_2 \in X\text{-}\mathbf{set};\; e \notin s_2}{(e \odot s_1) \cap s_2 = s_1 \cap s_2}$$

from $s \in X\text{-}\mathbf{set}$		
1	$\{\,\} \cap \{\,\} = \{\,\}$	$\cap\text{-}b$
2	**from** $e \in X, s_1 \in X\text{-}\mathbf{set}, s_1 \cap \{\,\} = \{\,\}$	
2.1	$e \notin \{\,\}$	$\in$
2.2	$(e \odot s_1) \cap \{\,\}$	
	$= s_1 \cap \{\,\}$	$\cap\text{-}i$
	infer $= \{\,\}$	ih2
infer $s \cap \{\,\} = \{\,\}$		*Set-ind*(1,2)

Answer 4.2.1 from page 97 a

from $s_1, s_2, s_3 \in X$-**set**
1 $(\{\,\} \cap s_2) \cap s_3 = \{\,\} \cap (s_2 \cap s_3)$ $\quad$ $\cap$-b(twice)
2 **from** $e \in X, s \in X$-**set**, $(s \cap s_2) \cap s_3 = s \cap (s_2 \cap s_3)$
2.1 $(e \in s_2 \vee e \notin s_2) \wedge (e \in s_3 \vee e \notin s_3)$ $\quad$ *Set*
2.2 **from** $e \in s_2, e \in s_3$
2.2.1 $((e \odot s) \cap s_2) \cap s_3$
$= (e \odot (s \cap s_2)) \cap s_3)$ $\quad$ $\cap$-i
2.2.2 $= e \odot ((s \cap s_2) \cap s_3)$ $\quad$ $\cap$-i
2.2.3 $= e \odot (s \cap (s_2 \cap s_3))$ $\quad$ ih2
infer $= (e \odot s) \cap (s_2 \cap s_3)$ $\quad$ $\cap$-i
2.3/2.5 other 3 cases similar
infer $((e \odot s) \cap s_2) \cap s_3 = (e \odot s) \cap (s_2 \cap s_3)$ $\vee$-E(2.1, 2.2/5)
infer $(s_1 \cap s_2) \cap s_3 = s_1 \cap (s_2 \cap s_3)$ $\quad$ *Set-ind*(1,2)

Answer 4.2.1 from page 97 b

For commutativity one needs two lemmas then the main proof is straightforward as are the remaining proofs in this exercise.
Answer 4.2.2 from page 97

$$\boxed{\bigcup\text{-}b}\ \frac{}{\bigcup\{\,\} = \{\,\}}$$

$$\boxed{\bigcup\text{-}i}\ \frac{s \in X\text{-}\mathbf{set},\ ss \in (X\text{-}\mathbf{set})\text{-}\mathbf{set}}{\bigcup(s \odot ss) = s \cup \bigcup ss}$$

from $ss_1, ss_2 \in (X$-**set**$)$-**set**
1 $\bigcup(\{\,\} \cup ss_2)$
$= \bigcup ss_2$ $\quad$ $\cup$-b
2 $= \{\,\} \cup \bigcup ss_2$ $\quad$ $\cup$-b
3 $= \bigcup\{\,\} \cup \bigcup ss_2$ $\quad$ $\bigcup$-b
4 **from** $s \in X$-**set**, $ss \in (X$-**set**$)$-**set**, $\bigcup(ss \cup ss_2) = \bigcup ss \cup \bigcup ss_2$
4.1 $\bigcup((s \odot ss) \cup ss_2)$
$= \bigcup(s \odot (ss \cup ss_2))$ $\quad$ $\cup$-i
4.2 $= s \cup \bigcup(ss \cup ss_2)$ $\quad$ $\bigcup$-i
4.3 $= s \cup \bigcup ss \cup \bigcup ss_2$ $\quad$ ih4
infer $= \bigcup(s \odot ss) \cup \bigcup ss_2$ $\quad$ $\bigcup$-i
infer $\bigcup(ss_1 \cup ss_2) = \bigcup ss_1 \cup \bigcup ss_2$ $\quad$ *Set-ind*(3,4)

Answer 4.2.2 from page 97 b

Answer 4.2.3 from page 98

$$\boxed{\text{d-b}}\ \frac{s \in X\text{-}\mathbf{set}}{\{\,\} \Leftrightarrow s = \{\,\}}$$

$$\boxed{\text{d-i}}\ \frac{e \in X;\ s_1, s_2 \in X\text{-}\mathbf{set};\ e \notin s_2}{(e \odot s_1) \Leftrightarrow s_2 = e \odot (s_1 \Leftrightarrow s_2)}$$

$$\boxed{\text{d-i}}\ \frac{e \in X;\ s_1, s_2 \in X\text{-}\mathbf{set};\ e \in s_2}{(e \odot s_1) \Leftrightarrow s_2 = s_1 \Leftrightarrow s_2}$$

	from $s_1, s_2 \in X$-**set**	
1	$\{\,\} \cup s_2$	
	$= s_2$	$\cup$-b
2	$\in X$-**set**	h
3	**from** $s \in X$-**set**, $e \in X$, $(s \cup s_2) \in X$-**set**	
3.1	$(e \odot s) \cup s_2$	
	$= e \odot (s \cup s_2)$	$\cup$-i
3.2	**infer** $(e \odot s) \cup s_2 \in X$-**set**	ih3, $\odot$-sig
	infer $(s_1 \cup s_2) \in X$-**set**	*Set-ind*(2,3)

Type inference: *Answer 4.2.5 from page 98*

Answer 4.4.1 from page 105

BULKADD $(ws\colon Word\text{-}\mathbf{set})\ dups\colon Word\text{-}\mathbf{set}$
ext wr $dict : Word\text{-}\mathbf{set}$
post $dict = \overleftarrow{dict} \cup ws \wedge dups = ws \cap \overleftarrow{dict}$

Answer 4.4.2 from page 105

The initial state is $nms_0 = \{\,\}$,

ENTER $(nm\colon Name)$
ext wr $nms : Name\text{-}\mathbf{set}$
pre $nm \notin nms$
post $nms = \overleftarrow{nms} \cup \{nm\}$

EXIT $(nm\colon Name)$
ext wr $nms : Name\text{-}\mathbf{set}$
pre $nm \in nms$
post $nms = \overleftarrow{nms} \Leftrightarrow \{nm\}$

ISPRESENT $(nm\colon Name)\ res\colon \mathbb{B}$
ext rd $nms : Name\text{-}\mathbf{set}$
post $res \Leftrightarrow (nm \in nms)$

Answer 4.4.3 from page 105

In the initial state $ns_0 = ys_0 = \{\,\}$.
Notice that there is an 'invariant' that the ns and ys sets are always disjoint.

ENROL $(nm\colon Name)$
ext wr $ns : Name\text{-}\mathbf{set}$,
rd $ys : Name\text{-}\mathbf{set}$
pre $nm \notin (ns \cup ys)$
post $ns = \overleftarrow{ns} \cup \{nm\}$

PASS $(nm\colon Name)$
ext wr $ns : Name\text{-}\mathbf{set}$,
wr $ys : Name\text{-}\mathbf{set}$

pre $nm \in ns \wedge nm \notin ys$
post $ns = \overleftarrow{ns} - \{nm\} \wedge ys = \overleftarrow{ys} \cup \{nm\}$

$RESULT$ () res: $Name$-**set**
ext rd ys : $Name$-**set**
post $res = ys$

Answer 4.4.4 from page 105

As an invariant, the four sets ($singfem$, $singmale$, $marfem$, $marmale$) should remain pairwise disjoint.

$MARMALE$ () rs: $Name$-**set**
ext rd $marmale$: $Name$-**set**
post $rs = marmale$

$NEWFEM$ (f: $Name$)
ext wr $singfem$: $Name$-**set**
  rd $marfem$: $Name$-**set**
  rd $singmale$: $Name$-**set**
  rd $marmale$: $Name$-**set**
pre $f \notin (singfem \cup marfem \cup singmale \cup marmale)$
post $singfem = \overleftarrow{singfem} \cup \{f\}$

$MARRIAGE$ (m: $Name$, f: $Name$)
ext wr $singfem$: $Name$-**set**
  wr $marfem$: $Name$-**set**
  wr $singmale$: $Name$-**set**
  wr $marmale$: $Name$-**set**
pre $m \in singmale \wedge f \in singfem$
post $singfem = \overleftarrow{singfem} - \{f\} \wedge$
  $marfem = \overleftarrow{marfem} \cup \{f\} \wedge$
  $singmale = \overleftarrow{singmale} - \{m\} \wedge$
  $marmale = \overleftarrow{marmale} \cup \{m\}$

Answer 4.4.6 from page 108

The student should in addition query the intention of $GROUP$ (e.g. must es all be in one equivalence group?) and, at least for $EQUATE$, should worry about satisfiability.

$EQUATES$ (es: $\mathbb{N}$-**set**)
ext wr p : $Partition$
post $p = \{s \in \overleftarrow{p} \mid is\text{-}disj(s, es)\} \cup \{\bigcup\{s \in \overleftarrow{p} \mid \neg\, is\text{-}disj(s, es)\}\}$

$GROUPS$ (es: $\mathbb{N}$-**set**) r: $\mathbb{N}$-**set**
ext rd p : $Partition$
pre $es \subseteq \bigcup p$
post $\exists ps \cdot ps \subseteq p \wedge (\forall s \in ps \cdot \neg\, is\text{-}disj(es, s)) \wedge r = \bigcup ps$

5

Composite Objects and Invariants

5.1 Comments

With regard to Section 5.2, some care is needed in generating induction rules for difficult invariants (e.g. suppose the invariant eliminates the base element): see [Lun89]. much work has, of course, been done on describing types since VDM was first developed. In particular, the ideas of 'dependent products' offer ways of avoiding some invariants. It might be reasonable to add such extensions to VDM, but this has not been done in this book.

In Section 5.3, the decision to characterize the set of initial states by a predicate should be contrasted to the position in [Jon80b] where an *INIT* operation was (nearly) always included with a data type. Even there, the *INIT* operations never fitted well with proving that invariants were preserved; with the new approach to satisfiability proofs and the module notation for data types, recognising the special role of initial states is much cleaner.

5.2 Answers

Answer 5.1.1 from page 117

$$\textit{mk-Date}: \mathbb{N} \times \{\textsc{Jan}, \ldots, \textsc{Dec}\} \times \{1, \ldots, 31\} \to \textit{Date}$$

$$\begin{array}{l} \textit{year}: \textit{Date} \to \mathbb{N} \\ \textit{month}: \textit{Date} \to \{\textsc{Jan}, \ldots, \textsc{Dec}\} \\ \textit{day}: \textit{Date} \to \{1, \ldots, 31\} \end{array}$$

$$\textit{mk-Date}(1944, \textsc{Jun}, 1)$$

$$\begin{array}{l} \textit{earlier} : \textit{Date} \times \textit{Date} \to \mathbb{B} \\ \textit{earlier}(dt_1, dt_2)) \quad \triangleq \\ \qquad \textit{year}(dt_1) < \textit{year}(dt_2) \vee \\ \qquad \textit{year}(dt_1) = \textit{year}(dt_2) \wedge \\ \qquad\quad (\textit{month}(dt_1) < \textit{month}(dt_2) \vee \\ \qquad\quad \textit{month}(dt_1) = \textit{month}(dt_2) \wedge \textit{day}(dt_1) < \textit{day}(dt_2)) \end{array}$$

$$\begin{array}{l} \textit{earlier}(dt_1, dt_2) \quad \triangleq \\ \qquad \textbf{let } \textit{mk-Date}(y_1, m_1, d_1) = dt_1 \textbf{ in} \\ \qquad \textbf{let } \textit{mk-Date}(y_2, m_2, d_2) = dt_2 \textbf{ in} \\ \qquad y_1 < y_2 \vee \ldots \end{array}$$

$$\begin{array}{l} \textit{earlier}(\textit{mk-Date}(y_1, m_1, d_1), \textit{mk-Date}(y_2, m_2, d_2)) \quad \triangleq \\ \qquad y_1 < y_2 \vee \ldots \end{array}$$

$$
\begin{array}{l}
Date \;::\; \ldots \\
\textbf{inv}\ (mk\text{-}Date(y,m,d)) \triangleq \\
\quad (m \in \{\text{APR}, \text{JUN}, \text{SEP}, \text{NOV}\} \;\Rightarrow\; d \le 30) \wedge \\
\quad (m = \text{FEB} \wedge is\text{-}leapyr(y) \;\Rightarrow\; d \le 29) \wedge \\
\quad (m = \text{FEB} \wedge \neg\, is\text{-}leapyr(y) \;\Rightarrow\; d \le 28)
\end{array}
$$

$$\mu(b, \{year \mapsto 1986\}) = mk\text{-}Date(1986, \text{JUN}, 1)$$

Answer 5.1.2 from page 118

No invariant is required because there are no irregular times to be ruled out: on this occasion the reality to be modelled is 'smooth'.

$$
\begin{array}{lll}
Time \;::\; hr & : & \{0, \ldots, 23\} \\
 mn & : & \{0, \ldots, 59\} \\
 sc & : & \{0, \ldots, 59\}
\end{array}
$$

$$\mu(\_, mn \mapsto \_) \colon Time \times \{0, \ldots, 59\} \to Time$$

Answer 5.1.3 from page 118

$$
\begin{array}{l}
Light = Colour\text{-}\textbf{set} \\
\textbf{inv}\ (s) \triangleq \textbf{card}\ s = 1 \vee s = \{\text{RED}, \text{AMBER}\}
\end{array}
$$

Answer 5.1.4 from page 118

$$
\begin{array}{l}
Roomno \;::\; fl \;:\; \{1, \ldots, 25\} \\
 rm \;:\; \{0, \ldots, 63\} \\
\textbf{inv}\ (mk\text{-}Roomno(fl, rm)) \triangleq \\
\quad fl \ne 13 \wedge (fl = 1 \;\Rightarrow\; rm = 0) \wedge (fl \ge 21 \;\Rightarrow\; is\text{-}even(rm))
\end{array}
$$

Answer 5.1.5 from page 118

$$
\begin{array}{l}
mk\text{-}Llistel(5, \textbf{nil}) \\
mk\text{-}Llistel(1, \\
\quad mk\text{-}Llistel(2, \\
\qquad mk\text{-}Llistel(4, \textbf{nil}))) \\
\textbf{nil}
\end{array}
$$

$$\mu(first, tl \mapsto mk\text{-}Llistel(0, \textbf{nil}))$$

$$
\begin{array}{l}
ljoin : Llist \times Llist \to Llist \\
ljoin(l_1, l_2) \quad \triangleq \\
\quad \textbf{cases}\ l_1\ \textbf{of} \\
\quad \textbf{nil} \qquad\qquad\qquad \to l_2, \\
\quad mk\text{-}Llistel(h_1, t_1) \to \mu(l_1, tl \mapsto ljoin(t_1, l_2)) \\
\quad \textbf{end}
\end{array}
$$

Answer 5.1.6 from page 119

$$Pllist = \{\textbf{nil}\} \cup \mathbb{Z} \cup Pllistel$$

$$
\begin{array}{l}
Pllistel \;::\; car \;:\; Pllist \\
 cdr \;:\; Pllist
\end{array}
$$

$gather : Pllist \to \mathbb{Z}\text{-}\mathbf{set}$
$gather(l) \quad \triangleq$
  if $l = \mathbf{nil}$
  then $\{\}$
  else if $l \in \mathbb{Z}$
    then $\{l\}$
    else (**let** $mk\text{-}Pllistel(car, cdr) = l$ **in**
      $gather(car) \cup gather(cdr))$

$sumll : Pllist \to \mathbb{Z}$
$sumll(l) \quad \triangleq$
  if $l = \mathbf{nil}$
  then 0
  else if $l \in \mathbb{Z}$
    then l
    else (**let** $mk\text{-}Pllistel(car, cdr) = l$ **in**
      $sumll(car) + sumll(cdr))$

from	$l_1, l_2 \in Llist$	
1	$lsum(ljoin(\mathbf{nil}, l_2)) = lsum(l_2)$	*ljoin*
2	$lsum(\mathbf{nil}) = 0$	*lsum*
3	$lsum(ljoin(\mathbf{nil}, l_2)) = lsum(\mathbf{nil}) + lsum(l_2)$	1,2
4	**from** $hd \in \mathbb{Z}, tl \in Llist,$ $lsum(ljoin(tl, l_2)) = lsum(tl) + lsum(l_2)$	
4.1	$lsum(ljoin(mk\text{-}Llistel(hd, tl), l_2)) = lsum(mk\text{-}Llistel(hd, ljoin(tl, l_2)))$	*ljoin*
4.2	$= hd + lsum(ljoin(tl, l_2))$	*lsum*
4.3	$= hd + lsum(tl) + lsum(l_2)$	ih4
4.4	$lsum(mk\text{-}Llistel(hd, tl)) + lsum(l_2) = hd + lsum(tl) + lsum(l_2)$	*lsum*
	infer $lsum(ljoin(mk\text{-}Llistel(hd, tl), l_2)) = lsum(mk\text{-}Llistel(hd, tl)) + lsum(l_2)$	4.3,4.4
infer	$lsum(ljoin(l_1, l_2)) = lsum(l_1) + lsum(l_2)$	$Llist\text{-}ind(3,4)$

Answer 5.2.1 from page 120

$$\boxed{\textit{Pllist-ind}}\ \frac{p(\mathbf{nil});\quad i \in \mathbb{Z} \vdash p(i);\quad car, cdr \in \textit{Pllist}, p(car), p(cdr) \vdash\ p(\textit{mk-Pllistel}(car, cdr))}{ll \in \textit{Pllist}\ \vdash\ p(ll)}$$

$\textit{flatten} : \textit{Pllist} \to \textit{Llist}$
$\textit{flatten}(l)\ \ \triangleq$
 if $l = \mathbf{nil}$
 then nil
 else if $l \in \mathbb{Z}$
  then $\textit{mk-Llistel}(l, \mathbf{nil})$
  else (**let** $\textit{mk-Pllist}(car, cdr) = l$ **in**
   $\textit{ljoin}(\textit{flatten}(car), \textit{flatten}(cdr)))$

from $ll \in \textit{Plist}$		
1	$\textit{sumll}(\mathbf{nil}) = 0$	*sumll*
2	$\textit{lsum}(\textit{flatten}(\mathbf{nil})) = 0$	*lsum*, *flatten*
3	$\textit{sumll}(\mathbf{nil}) = \textit{lsum}(\textit{flatten}(\mathbf{nil}))$	1,2
4	**from** $i \in \mathbb{Z}$	
4.1	$\textit{sumll}(i) = i$	*sumll*,h4
4.2	$\textit{lsum}(\textit{flatten}(i))$	*flatten*,h4
	$= \textit{lsum}(\textit{mk-Llistel}(i, \mathbf{nil}))$	
4.3	$= i$	*lsum*
	infer $\textit{sumll}(i) = \textit{lsum}(\textit{flatten}(i))$	4.1,4.3
5	**from** $car, cdr \in \textit{Pllist}$,	
	sumll(car) = lsum(flatten(car)),	
	sumll(cdr) = lsum(flatten(cdr))	
5.1	$\textit{sumll}(\textit{mk-Lisplist}(car, cdr))$	
	$= \textit{sumll}(car) + \textit{sumll}(cdr)$	
5.2	$= \textit{lsum}(\textit{flatten}(car)) + \textit{lsum}(\textit{flatten}(cdr))$	ih5,ih5
5.3	$\textit{lsum}(\textit{flatten}(\textit{mk-Lisplist}(car, cdr)))$	
	$= \textit{lsum}(\textit{ljoin}(\textit{flatten}(car), \textit{flatten}(cdr)))$	
5.4	$= \textit{lsum}(\textit{flatten}(car)) +$ $\textit{lsum}(\textit{flatten}(cdr))$	Exercise 5.2.1
	infer $\textit{sumll}(\textit{mk-Pllist}(car, cdr))$ $= \textit{lsum}(\textit{flatten}(\textit{mk-Pllist}(car, cdr)))$	5.2,5.4
infer $\textit{sumll}(ll) = \textit{lsum}(\textit{flatten}(ll))$		*Lisplist-ind*(3,4,5)

Answer 5.2.2 from page 120

Answer 5.2.3 from page 123

$$
\begin{array}{l}
ins : \mathbb{N} \times Setrep \to Setrep \\
ins(n, sr) \ \ \triangleq \\
\quad \textbf{cases } sr \textbf{ of} \\
\quad \textbf{nil} \to mk\text{-}Node(\textbf{nil}, n, \textbf{nil}), \\
\quad mk\text{-}Node(lt, mv, rt) \to \textbf{if } n = mv \\
\qquad\qquad\qquad\qquad\quad \textbf{then } sr \\
\qquad\qquad\qquad\qquad\quad \textbf{else if } n < mv \\
\qquad\qquad\qquad\qquad\qquad \textbf{then } mk\text{-}Node(ins(n, lt), mv, rt) \\
\qquad\qquad\qquad\qquad\qquad \textbf{else } mk\text{-}Node(lt, mv, ins(n, rt)) \\
\quad \textbf{end}
\end{array}
$$

from	$n \in \mathbb{N}, sr \in Setrep$	
1	$ins(n, \textbf{nil}) = mk\text{-}Node(\textbf{nil}, n, \textbf{nil})$	
2	$inv\text{-}Node(ins(n, \textbf{nil}))$	$inv\text{-}Node$, $retrns$,1
3	$ins(n, \textbf{nil}) \in Setrep$	2,$Setrep$
4	$retrns(ins(n, \textbf{nil})) = \{n\}$	$retrns$
5	$\quad = retrns(\textbf{nil}) \cup \{n\}$	$retrns$
6	**from** $mn \in \mathbb{N}, lt, rt \in Setrep,$ $inv\text{-}Node(mk\text{-}Node(lt, mv, rt)),$ $ins(n, lt) \in Setrep,$ $ins(n, rt) \in Setrep,$ $retrns(ins(n, lt)) = retrns(lt) \cup \{n\},$ $retrns(ins(n, rt)) = retrns(rt) \cup \{n\}$	
6.1	$n = mv \vee n < mv \vee n > mv$	$\mathbb{N}$,h,h4
6.2	**from** $n = mv$	
6.2.1	$ins(n, mk\text{-}Node(lt, mv, rt))$ $= mk\text{-}Node(lt, mv, rt)$	ins,h4.2
6.2.2	$inv\text{-}Node(ins(n, mk\text{-}Node(lt, mv, rt)))$	6.2.1,h6
	infer $ins(n, mk\text{-}Node(lt, mv, rt)) \in Setrep \wedge$ $retrns(ins(n, mk\text{-}Node(lt, mv, rt)))$ $= retrns(mk\text{-}Node(lt, mv, rt)) \cup \{n\}$	6.2.2,$Setrep$
6.3	**from** $n < mv$	
6.3.1	$ins(n, mk\text{-}Node(lt, mv, rt))$ $= mk\text{-}Node(ins(n, lt), mv, rt)$	ins
	infer $ins(n, mk\text{-}Node(lt, mv, rt)) \in Setrep \wedge$ $retrns(ins(n, mk\text{-}Node(lt, mv, rt))) =$ $retrns(mk\text{-}Node(lt, mv, rt)) \cup \{n\}$	
6.4	**from** $n > m$ *similar* **infer** ...	
	infer ...	$\vee$-E(6.1,6.2,6.3,6.4)
infer	$ins(n, sr) \in Setrep \wedge retrns(ins(n, sr)) = retrns(sr) \cup \{n\}$	$Setrep\text{-}ind(5,6)$

Answer 5.2.3 from page 123

Answer 5.3.1 from page 130

$\forall x \in \mathbb{Z} \cdot \exists r \in \mathbb{Z} \cdot r = 2 * x$
$\forall i \in \mathbb{N} \cdot i = 3 \vee i = 8 \;\Rightarrow\; \exists j \in \mathbb{N} \cdot (i = 3 \;\Rightarrow\; j = 8) \wedge (i = 8 \;\Rightarrow\; j = 3)$
$\forall i, j \in \mathbb{Z} \cdot \exists r \in \mathbb{Z} \cdot r = i * j$

from		
1	**from** $x \in \mathbb{Z}$	
1.1	$\quad 2 * x \in \mathbb{Z}$	$\mathbb{Z}$,h1
1.2	$\quad 2 * x = 2 * x$	1.1
	infer $\exists r \in \mathbb{Z} \cdot r = 2 * x$	$\exists$-I(1.1,1.2)
infer $\forall x \in \mathbb{Z} \cdot \exists r \in \mathbb{Z} \cdot r = 2 * x$		$\forall$-I(1)

Answer 5.3.2 from page 130

6

Map Notation

6.1 Comments

Other interesting properties include:

$$\{\,\} \mathbin{\lhd\!\!\!\!-}\, m = m$$
$$m_1 \dagger m_2 = (\mathbf{dom}\, m_2 \mathbin{\lhd\!\!\!\!-}\, m_1) \cup m_2$$

There would clearly be a case for developing more notation for maps including the use of composition etc. Although not done in the chapter, the development can yield interesting project work.

6.2 Answers

Answer 6.1.1 from page 140

x
$\{a, b, c\}$
$\{x\}$
undefined
$\{a \mapsto x, b \mapsto x, c \mapsto x, d \mapsto x\}$
$\{a \mapsto x, b \mapsto y, c \mapsto x, d \mapsto x\}$
undefined
$\{a \mapsto x\}$
$\{b \mapsto x\}$

Answer 6.1.2 from page 141

$m \dagger \{\,\} = m$
$\{\,\} \dagger m = m$
$m_1 \dagger (m_2 \dagger m_3) = (m_1 \dagger m_2) \dagger m_3$
$\mathbf{dom}\,(m_1 \dagger m_2) = \mathbf{dom}\, m_1 \cup \mathbf{dom}\, m_2$
$\mathbf{rng}\,(m_1 \dagger m_2) = \mathbf{rng}\,(\mathbf{dom}\, m_2 \lhd m_1) \cup \mathbf{rng}\, m_2$
$\mathbf{dom}\,\{x \mapsto f(x) \mid p(x)\} = \{x \mid p(x)\}$
$\mathbf{rng}\,(m_1 \dagger m_2) \subseteq (\mathbf{rng}\, m_1 \cup \mathbf{rng}\, m_2)$

Answer 6.1.3 from page 141

$$\{\{1 \mapsto \{\textit{mk-Roomno}(1, 0)\}\}\} \cup$$
$$\{\{i \mapsto \{\textit{mk-Roomno}(i, j) \mid 0 \le j \le 63\} \mid 2 \le i \le 20 \wedge i \ne 13\}\} \cup$$
$$\{\{i \mapsto \{\textit{mk-Roomno}(i, j) \mid 0 \le j \le 62 \wedge \textit{is-even}(j)\} \mid 21 \le i \le 25\}\}$$

Answer 6.1.4 from page 141

$\textit{ELS}\ ()\ r{:}\, \mathbb{N}\textbf{-set}$
$\textbf{ext rd}\ p\ :\ \textit{Partrep}$

$\mathbf{post}\ r = \mathbf{dom}\, p$

$ADD\ (e{:}\, \mathbb{N})$
$\mathbf{ext}\ \mathbf{wr}\ p\ :\ Partrep$
$\mathbf{pre}\ e \notin \mathbf{dom}\, p$
$\mathbf{post}\ \exists id \in Pid \cdot id \notin \mathbf{rng}\, p \wedge p = \overset{\Leftrightarrow}{p} \cup \{e \mapsto id\}$

$EQUATE\ (e_1{:}\, \mathbb{N}, e_2{:}\, \mathbb{N})$
$\mathbf{ext}\ \mathbf{wr}\ p\ :\ Partrep$
$\mathbf{post}\ \mathbf{let}\ pid_1, pid_2 = \overset{\Leftrightarrow}{p}(e_1), \overset{\Leftrightarrow}{p}(e_2)\ \mathbf{in}$
$\qquad \exists pid \in \{pid_1, pid_2\} \cdot p = \overset{\Leftrightarrow}{p} \dagger \{e \mapsto pid \mid \overset{\Leftrightarrow}{p}(e) \in \{pid_1, pid_2\}\}$

$GROUP\ (e{:}\, \mathbb{N})\ r{:}\, \mathbb{N}\text{-}\mathbf{set}$
$\mathbf{ext}\ \mathbf{rd}\ p\ :\ Partrep$
$\mathbf{post}\ r = \{e_i \in \mathbf{dom}\, p \mid p(e_i) = p(e)\}$

$EQUATE\ (es{:}\, \mathbb{N}\text{-}\mathbf{set})$
$\mathbf{ext}\ \mathbf{wr}\ p\ :\ Partrep$
$\mathbf{post}\ \mathbf{let}\ pids = \{\overset{\Leftrightarrow}{p}(e) \mid e \in es\}\ \mathbf{in}$
$\qquad \exists pid \in pids \cdot p = \overset{\Leftrightarrow}{p} \dagger \{e \mapsto pid \mid \overset{\Leftrightarrow}{p}(e) \in pids\}$

Answer 6.1.5 from page 143

$\mathbf{rng}\, m = \{second(p) \mid p \in m\}$
$m_1 \dagger m_2 = m_2 \cup \{p \in m_1 \mid \neg \exists p_2 \in m_2 \cdot first(p_2) = first(p)\}$
$m_1 \cup m_2 = m_1 \cup m_2$
$s \triangleleft m = \{p \in m \mid first(p) \in s\}$
$s \mathbin{\triangleleft\!\!\!-} m = \{p \in m \mid first(p) \notin s\}$

Answer 6.2.2 from page 146

$$\boxed{\triangleleft\text{-}b}\ \frac{s \in D\text{-}\mathbf{set}}{s \triangleleft \{\,\} = \{\,\}}$$

$$\boxed{\triangleleft\text{-}i}\ \frac{s \in D\text{-}\mathbf{set};\ m \in (D \overset{m}{\longleftrightarrow} R);\ d \in D;\ r \in R;\ d \notin s}{s \triangleleft (\{d \mapsto r\} \odot m) = s \triangleleft m}$$

$$\boxed{\triangleleft\text{-}i}\ \frac{s \in D\text{-}\mathbf{set};\ m \in (D \overset{m}{\longleftrightarrow} R);\ d \in D;\ r \in R;\ d \in s}{s \triangleleft (\{d \mapsto r\} \odot m) = \{d \mapsto r\} \odot (s \triangleleft m)}$$

$$\boxed{\triangleleft\!\!\!-\, b}\ \frac{s \in D\text{-}\mathbf{set}}{s \mathbin{\triangleleft\!\!\!-} \{\,\} = \{\,\}}$$

$$\boxed{\triangleleft\!\!\!-\, i}\ \frac{s \in D\text{-}\mathbf{set};\ m \in (D \overset{m}{\longleftrightarrow} R);\ d \in D;\ r \in R;\ d \notin s}{s \mathbin{\triangleleft\!\!\!-} (\{d \mapsto r\} \odot m) = \{d \mapsto r\} \odot (s \mathbin{\triangleleft\!\!\!-} m)}$$

$$\boxed{\triangleleft\!\!\!-\, i}\ \frac{s \in D\text{-}\mathbf{set};\ m \in (D \overset{m}{\longleftrightarrow} R);\ d \in D;\ r \in R;\ d \in s}{s \mathbin{\triangleleft\!\!\!-} (\{d \mapsto r\} \odot m) = s \mathbin{\triangleleft\!\!\!-} m}$$

$$\boxed{\cup\text{-}b}\ \frac{m \in (D \overset{m}{\longleftrightarrow} R)}{\{\,\} \cup m = m}$$

$$\boxed{\cup\text{-}i}\ \frac{m_1, m_2 \in (D \xleftrightarrow{m} R);\ d \in D;\ r \in R;\ \textit{is-disj}(\{d\} \cup \mathbf{dom}\, m_1, \mathbf{dom}\, m_2)}{(\{d \mapsto r\} \odot m_1) \cup m_2 = \{d \mapsto r\} \odot (m_1 \cup m_2)}$$

from	$m \in (D \xleftrightarrow{m} R)$	
1	$\{\,\} \lhd \{\,\} = \{\,\}$	$\lhd$-b
2	**from** $d \in D, r \in R, \{\,\} \lhd m = \{\,\}$	
2.1	$d \notin \{\,\}$	*Set*
2.2	$\{\,\} \lhd (\{d \mapsto r\} \odot m)$	
	$= \{\,\} \lhd m$	$\lhd$-i, 2.1
	infer $= \{\,\}$	ih2
infer	$\{\,\} \lhd m = \{\,\}$	*Map-ind*(1,2)

Answer 6.2.3 from page 147

from	$m \in D \xleftrightarrow{m} R$	
1	$\{\,\} \cup \{\,\} = \{\,\}$	$\cup$-b
2	**from** $d \in D, r \in R, m \in D \xleftrightarrow{m} R, m \cup \{\,\} = m$	
2.1	$\textit{is-disj}(\{d\} \cup \mathbf{dom}\, m, \mathbf{dom}\, \{\,\})$	**dom**-b, *Set*
2.2	$(\{d \mapsto r\} \odot m) \cup \{\,\}$	
	$= \{d \mapsto r\} \odot (m \cup \{\,\})$	2.1, $\cup$-i
2.3	**infer** $= \{d \mapsto r\} \odot m$	ih2
infer	$m \cup \{\,\} = m$	*Map-ind*(1,2)

Answer 6.2.3 from page 147

from	$m_1, m_2, m_3 \in D \xleftrightarrow{m} R$, $\textit{is-disj}(\mathbf{dom}\, m_1, \mathbf{dom}\, m_2)$, $\textit{is-disj}(\mathbf{dom}\, m_2, \mathbf{dom}\, m_3)$, $\textit{is-disj}(\mathbf{dom}\, m_1, \mathbf{dom}\, m_3)$	
1	$(\{\,\} \cup m_2) \cup m_3$	
	$= m_2 \cup m_3$	$\cup$-b
2	$= \{\,\} \cup (m_2 \cup m_3)$	$\cup$-b
3	**from** $d \in D, r \in R, m \in (D \xleftrightarrow{m} R), \textit{is-disj}(\mathbf{dom}\, m, \mathbf{dom}\, m_2)$, $\textit{is-disj}(\mathbf{dom}\, m, \mathbf{dom}\, m_3)$, $d \notin (\mathbf{dom}\, m \cup \mathbf{dom}\, m_2 \cup \mathbf{dom}\, m_3)$, $(m \cup m_2) \cup m_3 = m \cup (m_2 \cup m_3)$	
3.1	$((\{d \mapsto r\} \odot m) \cup m_2) \cup m_3$	
	$= (\{d \mapsto r\} \odot (m \cup m_2)) \cup m_3$	$\cup$-i(h3)
3.2	$= \{d \mapsto r\} \odot ((m \cup m_2) \cup m_3)$	$\cup$-i(h3)
3.3	$= \{d \mapsto r\} \odot (m \cup (m_2 \cup m_3))$	ih3
	infer $= (\{d \mapsto r\} \odot m) \cup (m_2 \cup m_3)$	$\cup$-i,h3
infer	$(m_1 \cup m_2) \cup m_3 = m_1 \cup (m_2 \cup m_3)$	*Map-ind*(2,3)

Answer 6.2.3 from page 147

from	$d \in D, r \in R, m_1, m_2 \in (D \xrightarrow{m} R)$,	
	$d \notin \mathbf{dom}\, m_1, \textit{is-disj}(\mathbf{dom}\, m_1, \mathbf{dom}\, m_2)$	
1	$\{d \mapsto r\} \odot (\{\,\} \cup m_2)$	
	$= \{d \mapsto r\} \odot m_2$	$\cup$-b
2	$= \{\,\} \cup (\{d \mapsto r\} \odot m_2)$	$\cup$-b
3	**from** $d_2 \in D, r_2 \in R, m \in (D \xrightarrow{m} R)$	
	$\textit{is-disj}(\mathbf{dom}\, m, \mathbf{dom}\, m_2)$,	
	$d \notin (\mathbf{dom}\, m \cup \mathbf{dom}\, m_2)$,	
	$\{d \mapsto r\} \odot (m \cup m_2) = m \cup (\{d \mapsto r\} \odot m_2)$,	
	$d \neq d_2$	
3.1	$\{d \mapsto r\} \odot ((\{d_2 \mapsto r_2\} \odot m) \cup m_2)$	
	$= \{d \mapsto r\} \odot (\{d_2 \mapsto r_2\} \odot (m \cup m_2))$	$\cup$-i
3.2	$= \{d_2 \mapsto r_2\} \odot (\{d \mapsto r\} \odot (m \cup m_2))$	$\odot$-comm
3.3	$= \{d_2 \mapsto r_2\} \odot (m \cup (\{d \mapsto r\} \odot m_2))$	ih3
	infer $= (\{d_2 \mapsto r_2\} \odot m) \cup (\{d \mapsto r\} \odot m_2))$	$\cup$-i
infer	$\{d \mapsto r\} \odot (m_1 \cup m_2) = m_1 \cup (\{d \mapsto r\} \odot m_2)$	*Map-ind*(2,3)

from	$m_1, m_2 \in D \xrightarrow{m} R, \textit{is-disj}(\mathbf{dom}\, m_1, \mathbf{dom}\, m_2)$	
1	$\{\,\} \cup m_2$	
	$= m_2$	$\cup$-b
2	$= m_2 \cup \{\,\}$	*Map*
3	**from** $d \in D, r \in R, m \in (D \xrightarrow{m} R)$,	
	$\textit{is-disj}(\mathbf{dom}\, m, \mathbf{dom}\, m_2)$,	
	$d \notin (\mathbf{dom}\, m \cup \mathbf{dom}\, m_2), m \cup m_2 = m_2 \cup m$	
3.1	$(\{d \mapsto r\} \odot m) \cup m_2$	
	$= \{d \mapsto r\} \odot (m \cup m_2)$	$\cup$-i
3.2	$= \{d \mapsto r\} \odot (m_2 \cup m)$	ih3
	infer $= m_2 \cup (\{d \mapsto r\} \odot m)$	lemma
infer	$m_1 \cup m_2 = m_2 \cup m_1$	*Map-ind*(2,3)

Answer 6.2.3 from page 147

from	$m_1, m_2 \in (D \xrightarrow{m} R), \textit{is-disj}(\mathbf{dom}\, m_1, \mathbf{dom}\, m_2)$	
1	$m_1 \dagger \{\,\}$	
	$= m_1$	$\dagger$-b
2	$= m_1 \cup \{\,\}$	$\cup$-b
3	**from** $d \in D, r \in R, m \in (D \xrightarrow{m} R), \textit{is-disj}(\mathbf{dom}\, m, \mathbf{dom}\, m_1)$,	
	$d \notin (\mathbf{dom}\, m \cup \mathbf{dom}\, m_1), m_1 \dagger m = m_1 \cup m$	
3.1	$m_1 \dagger (\{d \mapsto r\} \odot m)$	
	$= \{d \mapsto r\} \odot (m_1 \dagger m)$	$\dagger$-i
3.2	$= \{d \mapsto r\} \odot (m_1 \cup m)$	ih3
	infer $= m_1 \cup (\{d \mapsto r\} \odot m)$	lemma
infer	$m_1 \dagger m_2 = m_1 \cup m_2$	*Map-ind*(2,3)

Answer 6.2.3 from page 147

Answer 6.3.1 from page 150

$$
\begin{array}{l}
CLOSEAC\ (ac\colon Acno)\ r\colon Balance\\
\textbf{ext wr}\ am\ :\ Acno \xrightarrow{m} Acdata\\
\textbf{pre}\ ac \in \textbf{dom}\ am\\
\textbf{post}\ am = \{ac\} \mathbin{⩤} \overleftarrow{am} \wedge\\
\qquad\quad r = bal(\overleftarrow{am}(ac))
\end{array}
$$

Assume no check on customer number; the result shows a balance to be paid out to the customer.

$$
\begin{array}{l}
REMC\ (cu\colon Cno)\\
\textbf{ext wr}\ odm\ :\ Cno \xrightarrow{m} Overdraft\\
\qquad \textbf{rd}\ am\ \ :\ Acno \xrightarrow{m} Acdata\\
\textbf{pre}\ cu \in \textbf{dom}\ odm \wedge\\
\qquad \neg\, \exists ac \in \textbf{dom}\ am \cdot own(am(ac)) = cu\\
\textbf{post}\ odm = \{cu\} \mathbin{⩤} \overleftarrow{odm}
\end{array}
$$

Assume all accounts have been closed first.

$$
\begin{array}{l}
TRANSF\ (f\colon Acno, t\colon Acno, a\colon \mathbb{N})\\
\textbf{ext rd}\ odm\ :\ Cno \xrightarrow{m} Overdraft\\
\qquad \textbf{wr}\ am\ \ :\ Acno \xrightarrow{m} Acdata\\
\textbf{pre}\ f, t \in \textbf{dom}\ am \wedge f \neq t \wedge\\
\qquad bal(am(f)) - a \geq odm(own(am(f)))\\
\textbf{post}\ am = \overleftarrow{am} \dagger\\
\qquad \{f \mapsto \mu(\overleftarrow{am}(f), bal \mapsto bal(\overleftarrow{am}(f)) - a),\\
\qquad\ \ t \mapsto \mu(\overleftarrow{am}(t), bal \mapsto bal(\overleftarrow{am}(t)) + a)\}
\end{array}
$$

Assume operation only invoked if overdrafts OK.

$$
\begin{array}{l}
CHGOD\ (cu\colon Cno, od\colon Overdraft)\\
\textbf{ext wr}\ odm\ :\ Cno \xrightarrow{m} Overdraft\\
\qquad \textbf{rd}\ am\ \ :\ Acno \xrightarrow{m} Acdata\\
\textbf{pre}\ cu \in \textbf{dom}\ odm \wedge\\
\qquad \neg\, \exists ac \in \textbf{dom}\ am \cdot own(am(ac)) = cu \wedge bal(am(ac)) < od\\
\textbf{post}\ odm = \overleftarrow{odm} \dagger \{cu \mapsto od\}
\end{array}
$$

Assume cannot change overdraft so as to violate invariant (this shows the weakness in the reality of this model!)

$$Bank = Acno \xrightarrow{m} Acdata$$

$$
\begin{array}{l}
Acdata :: own\ :\ Cno\\
\qquad\qquad\ \ bal\ \ :\ Balance\\
\qquad\qquad\ \ od\ \ \ :\ Overdraft\\
\textbf{inv}\ (mk\text{-}Acdata(own, bal, od)) \triangleq bal \geq -od
\end{array}
$$

Answer 6.3.2 from page 151

$$
\begin{array}{l}
REM\ (e\colon X)\\
\textbf{ext wr}\ b\ :\ Bag\\
\textbf{pre}\ mpc(e, b) \geq 1
\end{array}
$$

post if $mpc(e, b) = 1$
  then $b = \{e\} \Leftrightarrow \overleftrightarrow{b}$
  else $b = \overleftrightarrow{b} \dagger \{e \mapsto mpc(e, \overleftrightarrow{b}) \Leftrightarrow 1\}$

$$\forall e \in X, \overleftrightarrow{b} \in Bag \cdot pre\text{-}REM(e, \overleftrightarrow{b}) \ \Rightarrow\ \exists b \in Bag \cdot post\text{-}REM(e, \overleftrightarrow{b}, b)$$

from $e \in X, \overleftrightarrow{b} \in Bag$		
1	**from** $pre\text{-}REM(e, \overleftrightarrow{b})$	
1.1	$mpc(e, \overleftrightarrow{b}) \geq 1$	pre-REM,h1
1.2	**from** $mpc(e, \overleftrightarrow{b}) = 1$	
1.2.1	$\{e\} \lhd \overleftrightarrow{b} \in Bag$	h
	infer $\exists b \in Bag \cdot post\text{-}REM(e, \overleftrightarrow{b}, b)$	1.2.1
1.3	**from** $mpc(e, \overleftrightarrow{b}) > 1$	
1.3.1	$(mpc(e, \overleftrightarrow{b}) \Leftrightarrow 1) \in \mathbb{N}_1$	
1.3.2	$(\overleftrightarrow{b} \dagger \{e \mapsto mpc(e, \overleftrightarrow{b}) \Leftrightarrow 1\}) \in Bag$	1.3.1
	infer $\exists b \in Bag \cdot post\text{-}REM(e, \overleftrightarrow{b}, b)$	
	infer $\exists b \in Bag \cdot post\text{-}REM(e, \overleftrightarrow{b}, b)$	$\vee$-E(1.1,1.2,1.3)
infer $pre\text{-}REM(e, \overleftrightarrow{b}) \ \Rightarrow\ \exists b \in Bag \cdot post\text{-}REM(e, \overleftrightarrow{b}, b)$		$\Rightarrow$-I

Sequent form of *Answer 6.3.2 from page 151*

Answer 6.3.4 from page 155

$$Studx = Studnm \xrightarrow{m} \{\mathrm{Y},\mathrm{N}\}$$

The initial state is:

$$m_0 = \{\,\}$$

No invariant is required on this state.

ENROL $(nm\colon Studnm)$
ext wr $m : Studx$
pre $nm \notin$ **dom** m
post $\overleftrightarrow{m} = m \cup \{nm \mapsto \mathrm{N}\}$

PASS $(nm\colon Studnm)$
ext wr $m : Studx$
pre $nm \in$ **dom** $m \wedge m(nm) = \mathrm{N}$
post $m = \overleftrightarrow{m} \dagger \{nm \mapsto \mathrm{Y}\}$

RESULT $()$ $res\colon Studnm$-**set**
ext rd $m : Studx$
post $res = \{nm \in$ **dom** $m \mid m(nm) = \mathrm{Y}\}$

Answer 6.3.5 from page 156

$$Studw = Name \xrightarrow{m} Roomno$$

The initial state is:

$$m_0 = \{\,\}$$

$ARRIVE\ (nm\colon Name, rn\colon Roomno)$
ext wr $m\ :\ Studw$
pre $nm \notin \mathbf{dom}\, m$
post $m = \overleftarrow{m} \cup \{nm \mapsto rm\}$

$MOVE\ (nm, Name, rn\colon Roomno)$
ext wr $m\ :\ Studw$
pre $nm \in \mathbf{dom}\, m$
post $m = \overleftarrow{m} \dagger \{nm \mapsto rn\}$

$WHO\ (rn\colon Roomno)\ res\colon Name\text{-}\mathbf{set}$
ext rd $m\ :\ Studw$
post $res = \{nm \in \mathbf{dom}\, m \mid m(nm) = rm\}$

Answer 6.3.6 from page 156

$ALLOC\ (nm\colon Name)\ rn\colon Roomno$
ext wr $occupancy\ :\ Roomno \xrightarrow{m} Name$
rd $rooms\ :\ Roomno\text{-}\mathbf{set}$
pre $\mathbf{dom}\, occupancy \subset rooms$
post $rn \in (rooms - \mathbf{dom}\, \overleftarrow{occupancy}) \wedge$
$occupancy = \overleftarrow{occupancy} \cup \{rn \mapsto nm\}$

$CHKOUT\ (rn\colon Roomno)$
ext wr $occupancy\ :\ Roomno \xrightarrow{m} Name$
pre $rn \in \mathbf{dom}\, occupancy$
post $occupancy = \{rn\} \mathbin{\lessdot\!\!-} \overleftarrow{occupancy}$

$SPARE\quad rns\colon Roomno\text{-}\mathbf{set}$
ext rd $occupancy\ :\ Roomno \xrightarrow{m} Name$
rd $rooms\ :\ Roomno\text{-}\mathbf{set}$
post $rns = rooms - \mathbf{dom}\, occupancy$

Answer 6.3.7 from page 156

$Bom = Pno \xrightarrow{m} Pno\text{-}\mathbf{set}$
$\mathbf{inv}\ (m) \triangleq is\text{-}wfr(\bigcup\{\{(p_1, p_2) \mid p_2 \in m(p_1)\} \mid p_1 \in \mathbf{dom}\, m\})$

$is\text{-}wfr\colon (X \times X)\text{-}\mathbf{set} \to \mathbb{B}$

$expla : Pno \times Bom \to Pno\text{-}\mathbf{set}$
$expla(p, m) \quad \triangleq \quad \{p\} \cup \bigcup\{expla(c, m) \mid c \in m(p)\}$
pre $p \in \mathbf{dom}\, m$

$explb : Pno \times Bom \to Pno\text{-}\mathbf{set}$
$explb(p, m) \quad \triangleq \quad$ **if** $m(p) = \{\,\}$
then $\{p\}$
else $\bigcup\{explb(c, m) \mid c \in (p)\}$

pre $p \in \mathbf{dom}\ m$

$WHEREUSED\ (p: Pno)\ r: Pno\text{-}\mathbf{set}$
ext rd $b : Bom$
post $r = \{a \in \mathbf{dom}\ b \mid p \in b(a)\}$

7

Sequence Notation

7.1 Comments

In Section 7.2, it is interesting to note that the observations made in Exercise 7.2.4 can also be used as evidence for a transformational style of development.

7.2 Answers

Answer 7.1.1 from page 164

true, **false**, **true** and **false** respectively.

Answer 7.1.2 from page 165

$[b]$
2
a
$[\,]$
$[a, b]$
$\{\{a\}, a, [a]\}$
$[a, a]$
$[a, [b]]$

Answer 7.1.3 from page 165

$s_a = [a, a]$
$s_b = [[b], \{1\}, b]$
$s_c = [a, a]$

Answer 7.1.4 from page 165

$$\begin{array}{l} \mathit{is\text{-}uniques}\ (s\colon X^*)\ r\colon \mathbb{B} \\ \textbf{post}\ r \Leftrightarrow \forall i, j \in \textbf{inds}\, s \cdot i = j \vee s(i) \neq s(j) \end{array}$$

or:

$$\mathit{is\text{-}uniques}(s) \quad \triangleq \quad \forall i, j \in \textbf{inds}\, s \cdot i = j \vee s(i) \neq s(j)$$

$$\begin{array}{l} \mathit{arbl}\ (b\colon X\text{-}\textbf{set})\ s\colon X^* \\ \textbf{post}\ \textbf{elems}\, s = b \wedge \mathit{is\text{-}uniques}(s) \end{array}$$

Answer 7.1.5 from page 165

$$\begin{array}{l} \mathit{alloccs}\ (s\colon X^*, v\colon X)\ r\colon \mathbb{N}_1\text{-}\textbf{set} \\ \textbf{post}\ r = \{i \in \textbf{inds}\, s \mid s(i) = v\} \end{array}$$

or:

$alloccs(s, v) \quad \triangleq \quad \{i \in \mathbf{inds}\, s \mid s(i) = v\}$

$firstocc\ (s\colon X^*, v\colon X)\ r\colon \mathbb{N}_1$
pre $v \in \mathbf{elems}\, s$
post $s(r) = v \wedge v \notin \mathbf{elems}\,(s(1, \ldots, r \Leftrightarrow 1))$

$locate\ (ss\colon X^*, s\colon X^*)\ r\colon \mathbb{N}$
post $\neg\, \exists i, j \in \mathbb{N} \cdot s(i, \ldots, j) = ss \wedge r = 0 \vee$
$\qquad \exists k \in \mathbb{N} \cdot s(r, \ldots, k) = ss \wedge \neg\, \exists i, j \in \mathbb{N} \cdot i < r \wedge s(i, \ldots, j) = ss$

Answer 7.1.6 from page 166

$\mathbf{inds}\, s \triangleq \mathbf{dom}\, s$
$\mathbf{elems}\, s \triangleq \mathbf{rng}\, s$

And sequence equality is just map equality.

$s_1 \frown s_2 = s_1 \cup \{i + \mathbf{len}\, s_1 \mapsto s_2(i) \mid i \in \mathbf{dom}\, s_2\}$
$\mathbf{hd}\, s = s(1)$
$\mathbf{tl}\, s = \{i \Leftrightarrow 1 \mapsto s(i) \mid i \in \{2, \ldots, \mathbf{card}\, \mathbf{dom}\, s\}\}$

$s(i, \ldots, j) \quad \triangleq \quad \{(k \Leftrightarrow i) + 1 \mapsto s(k) \mid k \in \{i, \ldots, j\}\}$

from	$s \in X^*$	
1	$[\,] \frown [\,] = [\,]$	$\frown$-b
2	**from** $e \in X, t \in X^*, t \frown [\,] = t$	
2.1	$\quad cons(e, t) \frown [\,]$	
	$\qquad = cons(e, t \frown [\,])$	$\frown$-i
	infer $\qquad = cons(e, t)$	ih2
infer	$s \frown [\,] = s$	$Seq\text{-}ind(1,2)$

Answer 7.2.1 from page 168

from	$s_1, s_2, s_3 \in X^*$	
1	$([\,] \frown s_2) \frown s_3$	
	$\quad = s_2 \frown s_3$	$\frown$-b
2	$\quad = [\,] \frown (s_2 \frown s_3)$	$\frown$-b
3	**from** $e \in X, s \in X^*,$ $(s \frown s_2) \frown s_3 = s \frown (s_2 \frown s_3)$	
3.1	$\quad (cons(e, s) \frown s_2) \frown s_3$	
	$\qquad = cons(e, (s \frown s_2)) \frown s_3$	$\frown$-i
3.2	$\qquad = cons(e, ((s \frown s_2) \frown s_3))$	$\frown$-i
3.3	$\qquad = cons(e, (s \frown (s_2 \frown s_3)))$	ih3
	infer $\qquad = cons(e, s) \frown (s_2 \frown s_3)$	$\frown$-i
infer	$(s_1 \frown s_2) \frown s_3 = s_1 \frown (s_2 \frown s_3)$	$Seq\text{-}ind(2,3)$

Answer 7.2.1 from page 168

Answer 7.2.2 from page 168

$\mathbf{len}\,[\,] = 0$
$\mathbf{len}\; cons(e, s) = 1 + \mathbf{len}\; s$
$(cons(e, s))(1) = e$
$(cons(e, s))(i + 1) = s(i)$
$\mathbf{hd}\,(cons(e, s)) = e$
$\mathbf{tl}\,(cons(e, s)) = s$

from	$s_1, s_2 \in X^*$	
1	$[\,] \frown s_2 = s_2$	$\frown$-b
2	$\mathbf{len}\,[\,] = 0$	$\mathbf{len}$-b
3	$\mathbf{len}\,([\,] \frown s_2) = \mathbf{len}\,[\,] + \mathbf{len}\; s_2$	$=$-$subs$(1,2)
4	**from** $e \in X, s \in X^*$,	
	$\quad \mathbf{len}\,(s \frown s_2) = \mathbf{len}\; s + \mathbf{len}\; s_2$	
4.1	$\quad \mathbf{len}\,(cons(e, s) \frown s_2)$	
	$\quad\quad = \mathbf{len}\,(cons(e, (s \frown s_2)))$	$\frown$-i
4.2	$\quad\quad = 1 + \mathbf{len}\,(s \frown s_2)$	$\mathbf{len}$-i
4.3	$\quad\quad = 1 + \mathbf{len}\; s + \mathbf{len}\; s_2$	ih4
	infer $\quad = \mathbf{len}\,(cons(e, s)) + \mathbf{len}\; s_2$	$\mathbf{len}$-i
infer	$\mathbf{len}\,(s_1 \frown s_2) = \mathbf{len}\; s_1 + \mathbf{len}\; s_2$	*Seq-ind*(3,4)

Answer 7.2.2 from page 168

from	$t \in X^*$	
1	$t = [\,] \lor t = cons(e, s)$	*Seq*
2	**from** $t = [\,]$	
	infer $t = [\,] \lor cons(\mathbf{hd}\; t, \mathbf{tl}\; t) = t$	$\lor$-I(h2)
3	**from** $t = cons(e, s)$	
3.1	$\quad \mathbf{hd}\,(cons(e, s)) = e$	**hd**
3.2	$\quad \mathbf{tl}\,(cons(e, s)) = s$	**tl**
3.3	$\quad cons(\mathbf{hd}\,(cons(e, s)), \mathbf{tl}\; cons(e, s))) = cons(e, s)$	3.1,3.2
3.4	$\quad cons(\mathbf{hd}\; t, \mathbf{tl}\; t) = t$	ih3
	infer $t = [\,] \lor cons(\mathbf{hd}\; t, \mathbf{tl}\; t) = t$	$\lor$-I(3.4)
infer	$t = [\,] \lor cons(\mathbf{hd}\; t, \mathbf{tl}\; t) = t$	$\lor$-E(1,2,3)

Answer 7.2.2 from page 168

Answer 7.2.3 from page 169

$rev\ (s: X^*)\ r: X^*$
post $\mathbf{len}\; r = \mathbf{len}\; s \land \forall i \in \mathbf{inds}\; s \cdot r(i) = s(\mathbf{len}\; s + 1 \Leftrightarrow i)$

$$
\begin{aligned}
& rev(rev(s)) = s \\
& \textbf{len}\ rev(rev(s)) = \textbf{len}\ rev(s) = \textbf{len}\ s \\
& \forall i \in \textbf{inds}\ s \cdot (rev(rev(s)))(i) \\
& \quad = (rev(s))(\textbf{len}\,(rev(s)) + 1 \Leftrightarrow i) \\
& \quad = (rev(s))(\textbf{len}\, s + 1 \Leftrightarrow i) \\
& \quad = s(\textbf{len}\, s + 1 \Leftrightarrow (\textbf{len}\, s + 1 \Leftrightarrow i) \\
& \quad = s(i)
\end{aligned}
$$

$$
\mathit{is\text{-}palindrome}(s) \quad \triangleq \quad \forall i \in \textbf{inds}\ s \cdot s(i) = s(\textbf{len}\, s + 1 \Leftrightarrow i)
$$

$$
\begin{aligned}
& \mathit{is\text{-}palindrome}(s) \\
& \quad \textbf{len}\ rev(s) = \textbf{len}\ s \\
& \quad \forall i \in \textbf{inds}\ s \cdot (rev(s))(i) \\
& \quad\ = s(\textbf{len}\, s + 1 \Leftrightarrow i) \\
& \quad\ = s(i)
\end{aligned}
$$

Answer 7.3.1 from page 173

$Qtp = Qitem^*$

$ENQ\ (it: Qitem)$
ext wr $q\ :\ Qtp$
post $\exists i \in \textbf{inds}\ q \cdot del(q, i) = \overset{\Leftrightarrow}{q} \wedge q(i) = it$

$DEQ\ ()\ it: Qitem$
ext wr $q\ :\ Qtp$
pre $q \neq [\,]$
post $\overset{\Leftrightarrow}{q} = [it] \frown q$

$ISEMPTY\ ()\ r: \mathbb{B}$
ext rd $q\ :\ Qtp$
post $r \ \Leftrightarrow\ (q = [\,])$

$Qtps = Qitem\text{-}\mathbf{set}$

$ENQ\ (it: Qitem)$
ext wr $q\ :\ Qtps$
post $q = \overset{\Leftrightarrow}{q} \cup \{it\}$

Ignoring the problem of duplicates:

$DEQ\ ()\ it: Qitem$
ext wr $q\ :\ Qtps$
pre $q \neq \{\,\}$
post $it \in \overset{\Leftrightarrow}{q} \wedge q = \overset{\Leftrightarrow}{q} \Leftrightarrow \{it\} \wedge \neg \exists it_2 \in \overset{\Leftrightarrow}{q} \cdot p(it_2) < p(it)$

$ISEMPTY\ ()\ r: \mathbb{B}$
ext rd $q\ :\ Qtps$
post $r \ \Leftrightarrow\ (q = \{\,\})$

Now preserve order within priority:

$Qtpm = Priority \overset{m}{\Leftrightarrow\!\!\rightarrow} Data^*$

$\mathbf{inv}\ (m) \triangleq \mathbf{inds}\, m = Priority$

$ENQ\ (it: Qitem)$
$\mathbf{ext\ wr}\ q\ :\ Qtpm$
$\mathbf{post}\ \ q = \overleftarrow{q} \dagger \{p(it) \mapsto \overleftarrow{q}(p(it)) \frown [d(it)]\}$

$DEQ\ ()\ it: Qitem$
$\mathbf{ext\ wr}\ q\ :\ Qtpm$
$\mathbf{pre\ elems}\, q \neq \{[\,]\}$
$\mathbf{post\ let}\ i = mins(\{p \in \mathbf{inds}\ \overleftarrow{q} \mid \overleftarrow{q}(p) \neq [\,]\})\ \mathbf{in}$
$\quad q = \overleftarrow{q} \dagger \{i \mapsto \mathbf{tl}\ \overleftarrow{q}(i)\} \wedge it = \mathbf{hd}\,(\overleftarrow{q}(i))$

$ISEMPTY\ ()\ r: \mathbb{B}$
$\mathbf{ext\ rd}\ q\ :\ Qtpm$
$\mathbf{post}\ \ r \ \Leftrightarrow\ (\mathbf{elems}\, q = \{[\,]\})$

Answer 7.3.2 from page 173

$Stack = X^*$

With initial object: $s_0 = [\,]$.

$PUSH\ (e: X)$
$\mathbf{ext\ wr}\ s\ :\ Stack$
$\mathbf{post}\ \ s = [e] \frown \overleftarrow{s}$

$ISEMPTY\ ()\ r: \mathbb{B}$
$\mathbf{ext\ rd}\ s\ :\ Stack$
$\mathbf{post}\ \ r \ \Leftrightarrow\ (s = [\,])$

$POP\ ()\ e: X$
$\mathbf{ext\ wr}\ s\ :\ Stack$
$\mathbf{pre}\ \ s \neq [\,]$
$\mathbf{post}\ \ \overleftarrow{s} = [e] \frown s$

$Stackf = X^*$
$\mathbf{inv}\ (s) \triangleq \mathbf{len}\, s \leq 256$

$PUSH\ (e: X)$
$\mathbf{ext\ wr}\ s\ :\ Stackf$
$\mathbf{pre\ len}\, s < 256$
$\mathbf{post}\ \ s = [e] \frown \overleftarrow{s}$

ISEMPTY, POP as above *mutatis mutandis.*

$ISFULL\ ()\ r: \mathbb{B}$
$\mathbf{ext\ rd}\ s\ :\ Stackf$
$\mathbf{post}\ \ r \ \Leftrightarrow\ (\mathbf{len}\, s = 256)$

$PUSH\ (e: X)$
$\mathbf{ext\ wr}\ s\ :\ Stackf$

$\textbf{post}\ (\textbf{len}\ \overleftrightarrow{s} < 256 \wedge s = [e] \frown \overleftrightarrow{s}) \vee$
$\quad (\textbf{len}\ \overleftrightarrow{s} = 256 \wedge s = [e] \frown \overleftrightarrow{s}(1, \ldots, 255))$

The rest as in b.

Initially:

$s_0 = [\,]$
$i_0 = 0$

With an invariant:

$i \leq \textbf{len}\, s$

$POP\ ()$
$\textbf{ext wr}\ s\ :\ X^*,$
$\quad \textbf{wr}\ i\ :\ \mathbb{N}$
$\textbf{pre}\ s \neq [\,] \wedge i = \textbf{len}\, s$
$\textbf{post}\ s = \overleftrightarrow{s}(1, \ldots, \textbf{len}\ \overleftrightarrow{s} \Leftrightarrow 1) \wedge i = \textbf{len}\, s$

$PUSH\ (e{:}\, X)$
$\textbf{ext wr}\ s\ :\ X^*,$
$\quad \textbf{wr}\ i\ :\ \mathbb{N}$
$\textbf{pre}\ i = \textbf{len}\, s$
$\textbf{post}\ s = \overleftrightarrow{s} \frown [e] \wedge i = \textbf{len}\, s$

$DOWN$
$\textbf{ext rd}\ s\ :\ X^*,$
$\quad \textbf{wr}\ i\ :\ \mathbb{N}$
$\textbf{pre}\ i > 1$
$\textbf{post}\ i = \overleftrightarrow{i} \Leftrightarrow 1$

$READ\ ()\ r{:}\, X$
$\textbf{ext rd}\ s\ :\ X^*,$
$\quad \textbf{rd}\ i\ :\ \mathbb{N}$
$\textbf{pre}\ s \neq [\,]$
$\textbf{post}\ r = s(i)$

$RESET$
$\textbf{ext rd}\ s\ :\ X^*,$
$\quad \textbf{wr}\ i\ :\ \mathbb{N}$
$\textbf{post}\ i = \textbf{len}\, s$

Answer 7.3.3 from page 176

$Mcode = Letter \overset{m}{\Leftrightarrow\!\!\rightarrow} Letter$
$\textbf{inv}\ (m) \triangleq \textbf{dom}\, m = Letter \wedge \forall l \in \textbf{dom}\, m \cdot m(m(l)) = l$

Notice that this property implies 'is-oneone'. Only one 'CODE' operation need be specified!

Answer 7.3.4 from page 176

$Filenms = Filenm\text{-}\textbf{set}$

$Filenm = Char^*$

$MATCH\ (l: Char^*)\ r: Filenms$
ext rd $fs\ :\ Filenms$
post $r = \{f \in fs \mid is\text{-}prefix(l, f)\}$

Answer 7.3.6 from page 176

For the board:

$$Snlad :: b \ : \ Board$$
$$\qquad ps \ : \ Placings$$

$$Placings = Playerid \xleftrightarrow{m} [Posn]$$

$$Board = Posn \xleftrightarrow{m} Posn$$

inv $(b) \triangleq$ **dom** $b = Posn$

$Posn$ not further defined

$add: Posn \times Dice \to Posn$
$maxpl: \to Posn$

$MOVE\ (p: Playerid, d: Dice)$
ext rd $b \ :\ Board$
wr $ps\ :\ Placings$
pre $p \in$ **dom** ps
post $ps = \overleftrightarrow{ps} \dagger \{p \mapsto b(add(\overleftrightarrow{ps}(p), d))\}$

$FINISH\ ()\ p: [Playerid]$
ext rd $ps\ :\ Placings$
post $p \neq \mathbf{nil} \wedge ps(p) = maxpl() \vee$
$p = \mathbf{nil} \wedge \neg \exists pl \in \mathbf{dom}\ ps \cdot ps(pl) = maxpl()$

8

Data Reification

8.1 Comments

8.2 Answers

Answer 8.1.1 from page 187

In most languages *Dictb* could be very compact but would require regeneration – or large scale rearrangement – when new words added. In Pascal *Dictc* would use the heap and new words could be added easily but many of the arrays with a space for 26 entries would be largely empty. The commoning of the prefixes of words is some counter-balance to the use of a (4 byte?) pointer in place of a letter.

$$Word = Letter^{+}$$

$$Dictd = Letter \overset{m}{\Leftrightarrow\!\!\rightarrow} Word\text{-}\mathbf{set}$$
$$\mathbf{inv}\ (m) \triangleq \forall l \in \mathbf{dom}\, m \cdot \forall w \in m(l) \cdot \mathbf{hd}\, w = l$$

$$retr : Dictd \rightarrow Dict$$
$$retr(m) \quad \triangleq \quad \bigcup \mathbf{rng}\, m$$

Answer 8.1.2 from page 187

$$\begin{array}{lll} World :: & male & : Name\text{-}\mathbf{set} \\ & female & : Name\text{-}\mathbf{set} \\ & married & : Name\text{-}\mathbf{set} \end{array}$$
$$\mathbf{inv}\ (mk\text{-}World(m, f, e)) \triangleq is\text{-}disj(m, f) \wedge e \subseteq m \cup f$$

$$\begin{array}{lll} Worldx :: & singfem & : Name\text{-}\mathbf{set} \\ & marfem & : Name\text{-}\mathbf{set} \\ & singmale & : Name\text{-}\mathbf{set} \\ & marmale & : Name\text{-}\mathbf{set} \end{array}$$
$$\mathbf{inv}\ (mk\text{-}Worldx(sf, ef, sm, em)) \triangleq is\text{-}pdisj([sf, ef, sm, em])$$

$$retr\text{-}World : Worldx \rightarrow World$$
$$retr\text{-}World(mk\text{-}Worldx(sf, ef, sm, em)) \quad \triangleq$$
$$\quad mk\text{-}World(sm \cup em, sf \cup ef, em \cup ef)$$

$$retr\text{-}Worldx : World \rightarrow Worldx$$
$$retr\text{-}Worldx(mk\text{-}World(m, f, e)) \quad \triangleq$$
$$\quad mk\text{-}Worldx(f \Leftrightarrow e, f \cap e, m \Leftrightarrow e, m \cap e)$$

Note that the invariants are respected in both directions.

Answer 8.1.3 from page 187

$$Llist = [Llistel]$$

$$\begin{aligned} Llistel \; :: \; & hd \; : \; \mathbb{N} \\ & tl \; : \; Llist \end{aligned}$$

$$retr\text{-}Llist : \mathbb{N}^* \to Llist$$

$$\begin{aligned} retr\text{-}Llist(s) \quad \triangleq \quad & \textbf{if } s = [\,] \\ & \textbf{then nil} \\ & \textbf{else } mk\text{-}Llistel(\textbf{hd}\, s, retr\text{-}Llist(\textbf{tl}\, s)) \end{aligned}$$

$$retr\text{-}LIST : Llist \to \mathbb{N}^*$$

$$\begin{aligned} retr\text{-}LIST(l) \quad \triangleq \quad & \textbf{cases } l \textbf{ of} \\ & \textbf{nil} \to [\,], \\ & mk\text{-}Llistel(hd, tl) \to [hd] \frown retr\text{-}LIST(tl) \\ & \textbf{end} \end{aligned}$$

Answer 8.1.4 from page 187

$$Setrep = [Node]$$

$$\begin{aligned} Node \; :: \; & lt \; : \; Setrep \\ & mv \; : \; \mathbb{N} \\ & rt \; : \; Setrep \end{aligned}$$

inv ...

$$retr\text{-}SET : Setrep \to \mathbb{N}\text{-}\textbf{set}$$

$$\begin{aligned} & retr\text{-}SET(sr) \quad \triangleq \\ & \quad \textbf{cases } sr \textbf{ of} \\ & \quad \textbf{nil} \to \{\,\}, \\ & \quad mk\text{-}Node(lt, mv, rt) \to retr\text{-}SET(lt) \cup \{mv\} \cup retr\text{-}SET(rt) \\ & \quad \textbf{end} \end{aligned}$$

Adequacy:

$$\forall s \in \mathbb{N}\text{-}\textbf{set} \cdot \exists sr \in Setrep \cdot retr\text{-}SET(sr) = s$$

from $s \in \mathbb{N}$-**set**		
1	**nil** $\in$ *Setrep*	*Setrep*
2	$retr\text{-}SET(\mathbf{nil}) = \{\,\}$	*retr-SET*
3	$\exists sr \in Setrep \cdot retr\text{-}SET(sr) = \{\,\}$	$\exists$-*I*(1,2)
4	**from** $s \in \mathbb{N}$-**set**, $i \in \mathbb{N}$,	
	$\quad \exists sr \in Setrep \cdot retr\text{-}SET(sr) = s$	
4.1	$\quad$ **from** $sr \in Setrep, retr\text{-}SET(sr) = s$	
4.1.1	$\qquad ins(i, sr) \in Setrep$	*ins*
4.1.2	$\qquad retr\text{-}SET(ins(i, sr)) = \{i\} \cup retr\text{-}SET(sr)$	*ins*
	$\quad$ **infer** $\exists sr \in Setrep \cdot retr\text{-}SET(sr) = s \cup \{i\}$	$\exists$-*I*(4.1.1,4.1.2,4.1)
	infer $\exists sr \in Setrep \cdot retr\text{-}SET(sr) = s \cup \{i\}$	$\exists$-*E*(h4,4.1)
infer $\exists sr \in Setrep \cdot retr\text{-}SET(sr) = s$		$set\text{-}ind(h, 3, 4)$

Sequent form of *Answer 8.1.4 from page 187*

The *ins* results come from the answer to Exercise 5.2.3 on page 123.

Answer 8.1.5 from page 187

$$Binnum = Bit^*$$

$$Bit = \{0, 1\}$$

$$retr\text{-}\mathbb{N} : Binnum \to \mathbb{N}$$

$$\begin{array}{lll} retr\text{-}\mathbb{N}(bl) & \triangleq & \mathbf{if}\ bl = [\,] \\ & & \mathbf{then}\ 0 \\ & & \mathbf{else}\ 2 \uparrow (\mathbf{len}\ bl \Leftrightarrow 1) * \mathbf{hd}\ bl + retr\text{-}\mathbb{N}(\mathbf{tl}\ bl) \end{array}$$

Notice things are easier with:

$$Binnumt = [Comp]$$

$$\begin{array}{lll} Comp :: & f & : Binnumt \\ & b & : Bit \end{array}$$

$$retr\text{-}\mathbb{N} : Binnumt \to \mathbb{N}$$

$$\begin{array}{lll} retr\text{-}\mathbb{N}(t) & \triangleq & \mathbf{cases}\ t\ \mathbf{of} \\ & & \mathbf{nil} \qquad\qquad\qquad \to 0, \\ & & mk\text{-}Binnumt(f, b) \to 2 * retr\text{-}\mathbb{N}(f) + b \\ & & \mathbf{end} \end{array}$$

$$\begin{array}{lll} Signmag :: & s & : \{+, \Leftrightarrow\} \\ & b & : Binnum \end{array}$$

$$retr\text{-}\mathbb{Z} : Signmag \to \mathbb{Z}$$

$$\begin{array}{lll} retr\text{-}\mathbb{Z}(mk\text{-}Signmag(s, b)) & \triangleq & \mathbf{cases}\ s\ \mathbf{of} \\ & & + \to retr\text{-}\mathbb{N}(b), \\ & & \Leftrightarrow \to \Leftrightarrow retr\text{-}\mathbb{N}(b) \\ & & \mathbf{end} \end{array}$$

$$Onescomp :: Binnum$$

$$
\begin{array}{l}
\mathit{retr}\text{-}\mathbb{Z} : \mathit{Onescomp} \to \mathbb{Z} \\
\mathit{retr}\text{-}\mathbb{Z}(bl) \quad \triangleq \quad \textbf{if } bl \in \{1\}^* \\
\qquad\qquad\qquad\quad \textbf{then } 0 \\
\qquad\qquad\qquad\quad \textbf{else if hd } bl = 0 \\
\qquad\qquad\qquad\qquad \textbf{then } \mathit{retr}\text{-}\mathbb{N}(bl) \\
\qquad\qquad\qquad\qquad \textbf{else } \Leftrightarrow \mathit{retr}\text{-}\mathbb{N}(\mathit{comp}(bl))
\end{array}
$$

Answer 8.1.6 from page 188

The object $\mathit{mk\text{-}state}(\{1,3\}\{\,\})$ cannot be represented and, therefore, 'Rep' is <u>not</u> adequate. But:

$$
\begin{array}{l}
\mathit{State} \; :: \; \ldots \\
\textbf{inv } (\mathit{mk\text{-}State}(as, bs)) \triangleq \mathit{is\text{-}disj}(as, bs) \land \exists n \in \mathbb{N} \cdot as \cup bs = \{1, \ldots, n\}
\end{array}
$$

can be represented; the retrieve function is.

$$
\begin{array}{l}
\mathit{retr\text{-}State} : \mathit{Arep} \to \mathit{State} \\
\mathit{retr\text{-}State}(bl) \quad \triangleq \quad \mathit{mk\text{-}State}(\{i \in \textbf{inds } bl \mid bl(i)\}, \{i \in \textbf{inds } bl \mid \neg\, bl(i)\})
\end{array}
$$

Answer 8.2.1 from page 194

$$
\begin{array}{l}
\mathit{ADDWORDa} \; (w\colon \mathit{Word}) \\
\textbf{ext wr } \mathit{dict} \; : \; \mathit{Dicta} \\
\textbf{pre } \neg\, \exists i \in \textbf{inds } \mathit{dict} \cdot \mathit{dict}(i) = w \\
\textbf{post } \exists i \in \textbf{inds } \mathit{dict} \cdot \overleftarrow{\mathit{dict}} = \mathit{del}(\mathit{dict}, i) \land \mathit{dict}(i) = w
\end{array}
$$

The domain rule is as in Theorem 8.8 on page 192.

The range rule follows from:

$$
\begin{array}{l}
\overleftarrow{ws}, ws \in \mathit{Dicta},\; w \in \mathit{Word} \;\vdash \\
\qquad w \notin \textbf{elems } \overleftarrow{ws} \land \\
\qquad (\exists i \in \textbf{inds } ws \cdot \overleftarrow{ws} = \mathit{del}(ws, i) \land ws(i) = w) \;\Rightarrow\; \textbf{elems } ws = \textbf{elems } \overleftarrow{ws} \cup \{w\}
\end{array}
$$

$$
\begin{array}{lll}
\textbf{from } \overleftarrow{ws}, ws \in \mathit{Word}^*,\; w \in \mathit{Word} & & \\
1 \quad \textbf{from } \exists i \in \textbf{inds } ws \cdot \overleftarrow{ws} = \mathit{del}(ws, i) \land ws(i) = w & & \\
\qquad \textbf{infer elems } ws = \textbf{elems } \overleftarrow{ws} \cup \{w\} & & \textbf{elems} \\
2 \quad \delta(\exists i \in \textbf{inds } ws \cdot \overleftarrow{ws} = \mathit{del}(ws, i) \land ws(i) = w) & & \mathit{del}, \text{h} \\
\textbf{infer } \exists i \in \textbf{inds } ws \cdot \overleftarrow{ws} = \mathit{del}(ws, i) \land ws(i) = w \;\Rightarrow & & \\
\qquad \textbf{elems } ws = \textbf{elems } \overleftarrow{ws} \cup \{w\} & & \Rightarrow\text{-}I(2,1)
\end{array}
$$

Result rule for *Answer 8.2.1 from page 194*

$$
\begin{array}{l}
\mathit{CHECKWORDb} \; (w\colon \mathit{Word}) \; b\colon \mathbb{B} \\
\textbf{ext rd } \mathit{dict} \; : \; \mathit{Dictb} \\
\textbf{post } b \;\Leftrightarrow\; w \in \mathit{dict}(\textbf{len } w)
\end{array}
$$

Notice that:

$$
\begin{array}{l}
\mathit{ADDWORDb} \; (w\colon \mathit{Word}) \\
\textbf{ext wr } \mathit{dict} \; : \; \mathit{Dictb} \\
\textbf{pre } w \notin \mathit{dict}(\textbf{len } w) \\
\textbf{post } \; \ldots
\end{array}
$$

could have an undefined pre-condition (and is therefore wrong); and:

$ADDWORDb\ (w: Word)$
ext wr $dict\ :\ Dictb$
pre $\mathbf{len}\, dict \geq \mathbf{len}\, w \wedge w \notin dict(\mathbf{len}\, w)$
post $dict(\mathbf{len}\, w) = \overleftarrow{dict}(\mathbf{len}\, w) \cup \{w\}$

assumes an invariant which has not been discussed; a correct answer is:

$ADDWORDb\ (w: Word)$
ext wr $dict\ :\ Dictb$
pre $\mathbf{len}\, dict < \mathbf{len}\, w \vee w \notin dict(\mathbf{len}\, w)$
post $(\mathbf{len}\, \overleftarrow{dict} \geq \mathbf{len}\, w \;\Rightarrow\; dict(\mathbf{len}\, w) = \overleftarrow{dict}(\mathbf{len}\, w) \cup \{w\}) \wedge$
$(\mathbf{len}\, \overleftarrow{dict} < \mathbf{len}\, w \;\Rightarrow$
$dict = \overleftarrow{dict} \frown [i \mapsto \{\,\} \mid 1 \leq i \leq \mathbf{len}\, w \Leftrightarrow (\mathbf{len}\, \overleftarrow{dict} + 1)] \frown [\{w\}])$

which clearly indicates that a map would have been a better choice than a sequence (because of the ability to leave 'holes')! The retrieve function is:

$retr\text{-}Dict : Dictb \to Dict$
$retr\text{-}Dict(sl) \;\;\triangleq\;\; \bigcup(\mathbf{elems}\, s)$

Answer 8.2.2 from page 195

$Nms = Name\text{-}\mathbf{set}$

$Nml = Name^*$
inv $(s) \triangleq is\text{-}uniques(s)$

$retr\text{-}Nms : Nml \to Nms$
$retr\text{-}Nms(s) \;\;\triangleq\;\; \mathbf{elems}\, s$

Adequacy:

$\forall nms \in Name\text{-}\mathbf{set} \cdot \exists nml \in Name^* \cdot retr\text{-}Nms(nml) = nms$

$ENTER1\ (nm: Name)$
ext wr $nml\ :\ Name^*$
pre $nm \notin \mathbf{elems}\, nml$
post $nml = [nm] \frown \overleftarrow{nml}$

from $\overleftarrow{nml}, nml \in Name^*,\ nm \in Name$		
1	**from** $nml = \overleftarrow{nml} \frown [nm]$	
1.1	$\mathbf{elems}\, nml = \mathbf{elems}\, \overleftarrow{nml} \cup \mathbf{elems}\, [nm]$	L7.6(h1)
	infer $= \mathbf{elems}\, \overleftarrow{nml} \cup \{nm\}$	**elems**
2	$\delta(nml = \overleftarrow{nml} \frown [nm])$	$\frown$, h
infer $nml = \overleftarrow{nml} \frown [nm] \;\Rightarrow\; \mathbf{elems}\, nml = \mathbf{elems}\, \overleftarrow{nml} \cup \{nm\}$		$\Rightarrow$-I(2,1)

Result rule for $ENTER1$

$EXIT1\ (nm{:}\ Name)$
$\textbf{ext wr}\ nml\ :\ Name^*$
$\textbf{pre}\ nm \in \textbf{elems}\ nml$
$\textbf{post}\ \exists i \in \textbf{inds}\ nml \cdot nml(i) = nm \wedge nml = del(\overleftarrow{nml}, i)$

$ISPRESENT1\ (nm{:}\ Name)\ res{:}\ \mathbb{B}$
$\textbf{ext rd}\ nml\ :\ Name^*$
$\textbf{post}\ res \Leftrightarrow (nm \in \textbf{elems}\ nml)$

etc.

Answer 8.2.3 from page 195

$Studx = Name \xleftrightarrow{m} \{\mathrm{Y},\mathrm{N}\}$

$Studx1 :: NS : Name\text{-}\textbf{set},$
$\quad YS : Name\text{-}\textbf{set}$
$\textbf{inv}\ (mk\text{-}Studx1(ns, ys)) \triangleq is\text{-}disj(ns, ys)$

$retr\text{-}Studx : Studx1 \to Studx$
$retr\text{-}Studx(mk\text{-}Studx1(ns, ys)) \triangleq \{nm \mapsto \mathrm{Y} \mid nm \in ys\} \cup \{nm \mapsto \mathrm{N} \mid nm \in ns\}$

Notice, totality of retrieve function relies on the invariant for $Studx1$.

Answer 8.2.4 from page 195

The abstraction is:

$Tp :: ta : X^*$
$\quad tb : X^*$

$tp_0 = mk\text{-}Tp([\,], [\,])$

$ALA\ (v{:}\ X)$
$\textbf{ext wr}\ ta\ :\ X^*,$
$\quad \textbf{rd}\ tb\ :\ X^*$
$\textbf{pre len}\ ta + \textbf{len}\ tb < max$
$\textbf{post}\ ta = \overleftarrow{ta} \frown [v]$

$ALB\ (v{:}\ X)$
$\textbf{ext rd}\ ta\ :\ X^*,$
$\quad \textbf{wr}\ tb\ :\ X^*$
$\textbf{pre len}\ ta + \textbf{len}\ tb \leq max$
$\textbf{post}\ tb = \overleftarrow{tb} \frown [v]$

The representation is:

$Tt :: T : X^*$
$\quad pa : \mathbb{N}$
$\quad pb : \mathbb{N}$
$\textbf{inv}\ (mk\text{-}Tt(t, pa, pb)) \triangleq pa < pb \leq \textbf{len}\ t + 1$

$tt_0 = mk\text{-}Tt([\ldots], 0, n)$

$ALA1\ (v: X)$
$\textbf{ext wr } t\ : X^*,$
$\quad \textbf{wr } pa\ : \mathbb{N},$
$\quad \textbf{rd } pb\ : \mathbb{N}$
$\textbf{pre } \ldots$
$\textbf{post } t = \overleftarrow{t} \dagger [pa \mapsto v] \wedge pa = \overleftarrow{pa} + 1$

$ALB1\ (v: X)$
$\textbf{ext wr } t\ : X^*,$
$\quad \textbf{rd } pa\ : \mathbb{N},$
$\quad \textbf{wr } pb\ : \mathbb{N}$
$\textbf{pre } \ldots$
$\textbf{post } t = \overleftarrow{t} \dagger [pb \mapsto v] \wedge pb = \overleftarrow{pb} - 1$

$$retr\text{-}Tp(mk\text{-}Tt(t, pa, pb)) \quad \triangleq \quad mk\text{-}Tp(t(1, \ldots, pa), t(pb, \ldots, \textbf{len}\, t))$$

Answer 8.2.5 from page 195

$Store = Addr \xrightarrow{m} Val$

$Addr :: p : Pageno$
$\qquad\quad o : Offset$

$Page = Offset \xrightarrow{m} Val$
$\textbf{inv } (m) \triangleq \textbf{dom}\, m = Offset$

$Vstore :: fs : Pageno \xrightarrow{m} Page$
$\qquad\qquad bs : Pageno \xrightarrow{m} Page$
$\textbf{inv } (mk\text{-}Vstore(fs, bs)) \triangleq is\text{-}disj(\textbf{dom}\, fs, \textbf{dom}\, bs)$

$retr\text{-}Store : Vstore \to Store$
$retr\text{-}Store(mk\text{-}Vstore(fs, bs)) \quad \triangleq$
$\quad \{mk\text{-}Addr(p, o) \mapsto (fs(p))(o) \mid p \in \textbf{dom}\, fs \wedge o \in Offset\} \cup$
$\quad \{mk\text{-}Addr(p, o) \mapsto (bs(p))(o) \mid p \in \textbf{dom}\, bs \wedge o \in Offset\}$

$RD\ (a: Addr)\ v: Val$
$\textbf{ext rd } s : Store$
$\textbf{pre } a \in \textbf{dom}\, s$
$\textbf{post } v = s(a)$

$RDVS\ (a: Addr)\ v: Val$
$\textbf{ext wr } fs\ : Pageno \xrightarrow{m} Page,$
$\quad \textbf{wr } bs\ : Pageno \xrightarrow{m} Page$
$\textbf{pre } p(a) \in (\textbf{dom}\, fs \cup \textbf{dom}\, bs)$
$\textbf{post } fs \cup bs = \overleftarrow{fs} \cup \overleftarrow{bs} \wedge p(a) \in \textbf{dom}\, fs \wedge v = fs(p(a))(o(a))$

Answer 8.2.6 from page 195

With what has been done above, little remains to be done. Exercise 8.1.4 defines *retr-SET* and proves adequacy of *Setrep*. The function *retrns* of Section 5.2 is the same as *retr-SET* and thus the further results carry over:

$isin: \mathbb{N} \times Setrep \to \mathbb{B}$

And the properties proven in Section 5.2:

$ins: \mathbb{N} \times Setrep \to Setrep$

And the properties proven in Exercise 5.2.3:

$del: \mathbb{N} \times Setrep \to Setrep$

And the properties proven in Exercise 5.2.4.

These properties include the results needed for the modelling proofs for:

$ISIN\ (n: \mathbb{N})\ r: \mathbb{B}$
ext rd $sr : Setrep$
post $isin(n, sr) \;\Rightarrow\; n \in retr\text{-}SET(sr)$

$INS\ (n: \mathbb{N})$
ext wr $sr : Setrep$
pre $n \notin retr\text{-}SET(sr)$
post $sr = ins(n, \overleftarrow{sr})$

$DEL\ (n: \mathbb{N})$
ext wr $sr : Setrep$
pre $n \in retr\text{-}SET(sr)$
post $sr = del(n, \overleftarrow{sr})$

9

More on Data Types

9.1 Comments

In connection with operation quotation, it would be necessary to show that the quoting contexts were such that they would accept any implementation of the specification. This monotonicity (with respect to the satisfaction ordering) holds for normal programming language constructs but does not hold for arbitrary formulae of the predicate calculus.

In connection with annotation, it is interesting to note that there is enough markup in the LaTeXsource files to separate the formulae from the text. The conventions in – for example – 'Z' to use schema boxes are not really necessary once the text is handled on-line.

I have since realized that attributing the traversable stack to Veloso is a mistake, the originator was Majster (see Exercise 7.3.2).

9.2 Answers

Answer 9.1.1 from page 214

$$Diary = Date \xrightarrow{m} Task^*$$

$$d_0 = [\,]$$

$UPD\ (dt{:}\ Date, t{:}\ Task)$
ext wr $d\ :\ Diary$
post $dt \in \mathbf{dom}\ \overleftarrow{d} \wedge d = \overleftarrow{d} \dagger \{dt \mapsto \overleftarrow{d}(dt) \frown [t]\} \vee$
$\qquad dt \notin \mathbf{dom}\ \overleftarrow{d} \wedge d = \overleftarrow{d} \cup \{dt \mapsto [t]\}$

$$Diarysys = Uid \xrightarrow{m} Diary$$

$UPDSYS\ (u{:}\ Uid, dt{:}\ Date, t{:}\ Task)$
ext wr $ds\ :\ Diarysys$
pre $u \in \mathbf{dom}\ ds$
post $post\text{-}UPD(dt, t, \overleftarrow{ds}(u), ds(u)) \wedge \{u\} \ntriangleleft ds = \{u\} \ntriangleleft \overleftarrow{ds}$

Answer 9.2.1 from page 216

Thus:

$ENTER\ (nm{:}\ Name)$
ext wr $nms\ :\ Name$-**set**
pre $nm \notin nms$
post $nms = \overleftarrow{nms} \cup \{nm\}$
errs NAME PRES: $nm \in nms$

$EXITS\ (nm\colon Name)$
ext wr $nms : Name$-**set**
pre $nm \in nms$
post $nms = \overleftarrow{nms} \Leftrightarrow \{nm\}$
errs NAME ABS: $nm \notin nms$

ISPRESENT unchanged

Answer 9.2.2 from page 216

$POP\ ()\ e\colon X$
ext wr $s : Stack$
pre $s \neq [\,]$
post $\overleftarrow{s} = [e] \frown s$
errs STACK EMPTY: $s = [\,]$

$PUSH\ (e\colon X)$
ext wr $s : Stackf$
pre len $s < 256$
post $s = [e] \frown \overleftarrow{s}$
errs FULL: **len** $s = 256$

POP
ext wr $s : X^*$
wr $i : \mathbb{N}$
pre $s \neq [\,] \wedge i =$ **len** s
post $s = \overleftarrow{s}(1, \ldots,$ **len** $\overleftarrow{s} \Leftrightarrow 1) \wedge i =$ **len** s
errs STACK EMPTY: $s = [\,]$
CURSOR ERROR: $i \neq$ **len** s

$PUSH\ (e\colon X)$
ext wr $s : X^*$
wr $i : \mathbb{N}$
pre $i =$ **len** s
post $s = \overleftarrow{s} \frown [e] \wedge i =$ **len** s
errs CURSOR ERROR: $i \neq$ **len** s

DOWN
ext rd $s : X^*$
wr $i : \mathbb{N}$
pre $i > 1$
post $i = \overleftarrow{i} \Leftrightarrow 1$
errs CURSOR END: $i \leq 1$

$READ\ ()\ r\colon X$
ext rd $s : X^*$
rd $i : \mathbb{N}$
pre $i \neq [\,]$
post $r = s(i)$
errs STACKEMPTY: $s = [\,]$

Answer 9.2.3 from page 216

$MKDIR\ (n\colon Name)$
ext wr $d\ :\ Directory$
pre $n \notin \mathbf{dom}\, d$
post $d = \overleftarrow{d} \cup \{n \mapsto \{\,\}\}$
errs DUPLICATE: $n \in \mathbf{dom}\, d$

$Path = Name^*$

$SHOWP\ (p\colon Path)\ m\colon Dirstatus$
ext rd $d\ :\ Node$
pre $d \in Directory\ \wedge$
$(p = [\,] \vee$
$p \neq [\,]\ \wedge\ \mathbf{hd}\, p \in \mathbf{dom}\, d \wedge pre\text{-}SHOWP(\mathbf{tl}\, p, d(\mathbf{hd}\, p)))$
post $p = [\,] \wedge post\text{-}SHOWP(d, m)\ \vee$
$p \neq [\,] \wedge post\text{-}SHOWP(\mathbf{tl}\, p, d(\mathbf{hd}\, p), m)$
errs NOTDIR: $d \notin Directory$
INVALIDPATH: $p \neq [\,] \wedge (\mathbf{hd}\, p \notin \mathbf{dom}\, d \vee \neg\, pre\text{-}SHOWP(\mathbf{tl}\, p, d(\mathbf{hd}\, p)))$

Answer 9.3.7 from page 222

Rational nos (biased or not?)

$Rat \subseteq Int \times Int$

where

$$Inv\text{-}Rat(i,j) \quad \triangleq \quad j \neq 0 \wedge \neg\, \exists n \in \mathbb{N} \cdot n \neq 1 \wedge n \mathbf{divides}\ i \wedge n \mathbf{divides}\ j$$

$rational\colon Int \to \mathbf{Rat}$

$$(m_1, n_1) + (m_2, n_2) \quad \triangleq \quad \mathbf{reduce}\ (m_1 * n_2 + m_2 * n_1, n_1 * n_2)$$

$rational(i) \quad \triangleq \quad (i, 1)$
$\_ + \_\colon \mathbf{Rat}\ \times \mathbf{Rat}\ \to \mathbf{Rat}$

$reduce : Int \times Int \to \mathbf{Rat}$
$reduce(m,n) \quad \triangleq \quad (m/gcd(m,n), n/gcd(m,n))$

10

Operation Decomposition

10.1 Comments

Because of the level of formality in the development examples, it is tempting to compare what is done here with the constructive approach to program design considered in [C$^+$86] or [BCMS89]. Their approach constructs a proof of satisfiability in such a way that an implementation can then be extracted from it. Since the approach here also yields a program, the principal advantage of the constructive approaches is to ensure that the housekeeping of garnering the implementation is conducted within the same formal system as the proof. The availability of the less formal approach of Section 10.2 has dissuaded VDM researchers from following the constructive approach.

See [AhK89] for proof rules about procedures with 'by location' parameters.

Regarding sequential composition (cf. ;-I), any reader who is knowledgeable enough to fear that this is assuming 'angelic non-determinism' deserves reassurance. Because of the satisfiability requirement on all *pre*/*post* pairs, the use of pre_2 as a conjunct of the post information of S_1 avoids the problem.

10.2 Answers

Answer 10.1.1 from page 239

pre true
    MAKEPOS
    ext wr $m, n: \mathbb{Z}$
    pre true
    post $0 \leq m \wedge m * n = \overleftarrow{m} * \overleftarrow{n}$
    ;
    POSMUL
    ext wr $m, n, r: \mathbb{Z}$
    pre $0 \leq m$
    post $r = \overleftarrow{m} * \overleftarrow{n}$
post $r = \overleftarrow{m} * \overleftarrow{n}$

follows from:

$$post\text{-}MAKEPOS \mid post\text{-}POSMUL \;\Leftrightarrow\; \exists m_i, n_i \in \mathbb{Z} \cdot m_i * n_i = \overleftarrow{m} * \overleftarrow{n} \wedge r = m_i * n_i$$

Answer 10.1.2 from page 239

MAKEPOS
ext wr $m, n: \mathbb{Z}$
pre true
  if $m < 0$ **then** $(m := -m; n := -n)$
post $0 \le m \wedge m * n = \overleftarrow{m} * \overleftarrow{n}$

Follows from:

$$\overleftarrow{m} < 0 \wedge m = -\overleftarrow{m} \wedge n = -\overleftarrow{n} \;\Rightarrow\; 0 \le m \wedge m * n = \overleftarrow{m} * \overleftarrow{n}$$
$$\overleftarrow{m} \ge 0 \wedge m = \overleftarrow{m} \wedge n = \overleftarrow{n} \;\Rightarrow\; 0 \le m \wedge m * n = \overleftarrow{m} * \overleftarrow{n}$$
$$\delta_l(m < 0)$$

Answer 10.1.3 from page 239

Follows from:

$$\{0 \le m \wedge m \ne 0\}\ (m := m-1; r := r+n)\ \{0 \le m \wedge r+m*n = \overleftarrow{r} + \overleftarrow{m} * \overleftarrow{n} \wedge n = \overleftarrow{n} \wedge m < \overleftarrow{m}\}$$

and:

$$r + m * n = \overleftarrow{r} + \overleftarrow{m} * \overleftarrow{n} \wedge m = 0 \;\Rightarrow\; r = \overleftarrow{r} + \overleftarrow{m} * \overleftarrow{n}$$

Answer 10.1.4 from page 239

The outer loop is identical with that in the text. The inner loop is straightforward (see answer to Exercise 10.2.1) with only the termination argument being unusual.

Answer 10.1.5 from page 240

See answer to Exercise 10.2.2.

Answer 10.1.6 from page 240

This exercise should have been starred!

Answer 10.2.1 from page 243

POSMUL
ext wr $m, n, r: \mathbb{Z}$
pre $0 \le m$
  $r := 0;$
  pre $0 \le m$
    (**while** $m \ne 0$ **do**
    inv $0 \le m$
      ext wr $m, n: \mathbb{Z}$
      pre $0 < m$
        while $is\text{-}even(m)$ **do**
        inv $1 \le m$
          $(m := m/2; n := n * 2)$
        sofar $m * n = \overleftarrow{m} * \overleftarrow{n} \wedge m < \overleftarrow{m}$
      post $m * n = \overleftarrow{m} * \overleftarrow{n} \wedge m \le \overleftarrow{m}$
      ;
      $r := r + n; m := m - 1)$
    sofar $r + m * n = \overleftarrow{r} + \overleftarrow{m} * \overleftarrow{n} \wedge m < \overleftarrow{m}$
  post $r = \overleftarrow{r} + \overleftarrow{m} * \overleftarrow{n}$
post $r = \overleftarrow{m} * \overleftarrow{n}$

Answer 10.2.2 from page 243

$IDIV$
pre $n \neq 0$
  $q := 0$
  ;
  pre $n \neq 0$
    while $n \leq m$ **do**
    inv true
      $(m := m - n; q := q + 1)$
    sofar $\overleftarrow{n} * q + m = \overleftarrow{n} * \overleftarrow{q} + \overleftarrow{m} \wedge n = \overleftarrow{n} \wedge m < \overleftarrow{m}$
  post $\overleftarrow{n} * q + m = \overleftarrow{n} * \overleftarrow{q} + \overleftarrow{m} \wedge m < \overleftarrow{n}$
post $\overleftarrow{n} * q + m = \overleftarrow{m} \wedge m < \overleftarrow{n}$

Answer 10.3.3 from page 251

pre $0 \leq n$
  $fn := 1;$
  pre $0 \leq n$
    while $n \neq 0$ **do**
    inv $0 \leq n$
      $(fn := fn * n; n := n - 1)$
    sofar $fn * n! = \overleftarrow{fn} * \overleftarrow{n}! \wedge n < \overleftarrow{n}$
  post $fn = \overleftarrow{fn} * \overleftarrow{n}!$
post $fn = \overleftarrow{n}!$

pre $0 \leq n$
  $fn := 1; t := 0$
  ;
  pre $t \leq n \wedge fn = t!$
    while $t \neq n$ **do**
    inv $t \leq n \wedge fn = t!$
      $(t := t + 1; fn := fn * t)$
    sofar $n = \overleftarrow{n} \wedge \overleftarrow{t} < t$
  post $fn = t! \wedge t = n = \overleftarrow{n}$
post $fn = \overleftarrow{n}!$

Answer 10.4.1 from page 257

pre $0 \leq n$
  $fn := 1;$
  pre $0 \leq n$
    while $0 \leq n$ **dp**
    inv $0 \leq n$
      $(fn := fn * n;\ n := n - 1)$
    toend $fn = \overleftarrow{fn} * \overleftarrow{n}!$
  post $fn = \overleftarrow{fn} * \overleftarrow{n}!$
post $fn = \overleftarrow{n}!$

Reformulating the algorithm with the temporary t for **while**-$I2$ results in two uses of factorial in **toend**!

11

A Small Case Study

11.1 Comments

The analogy presented at the beginning of this chapter prompts questions about the *constructive approach* (see [BCMS89]). It is interesting to note how the actual implementation provides, the existence proof that an implementation is possible. This has prompted some computer scientists to follow the idea of creating programs by constructively proving the existence of a result. In some sense, the Constructive approach provides a single formal framework in which all of this is captured. The alternative preferred here is to have one formal system for the inference rules and an independent machine-based system to manage the connections between specifications and code.

It is also important to see that the proper decomposition of a problem avoids the 'VCG trap'.

The claim in Section 11.3 that *root* is total over *Forest* (but not over arbitrary $X \stackrel{m}{\longleftrightarrow} X$) follows from the invariant is non-trivial to formalize and best left to intuition.

Furthermore, $collapse(f) \neq f^+$ but $inv\text{-}Forest(f)$ must be equal to $f \cap I = \{\,\}$: attempts to prove the result in Figure 11.3 at this level have failed (so far).

11.2 Answers

Answer 11.3.2 from page 271

$\boxed{\text{extract F2–6 from Acta}}$

$\boxed{\text{check no reliance on loops at roots}}$

$$\boxed{\text{L??}}\ \frac{f \in Forest;\ r_1, r_2 \in roots(f);\ r_1 \neq r_2}{is\text{-}disj(coll(r_1, f), coll(r_2, f))}$$

$$\boxed{\text{L??}}\ \frac{d, e \in X;\ f \in Forest;\ \neg\, is\text{-}before(e, d, f)}{(f \dagger \{d \mapsto e\}) \in Forest}$$

$$\boxed{\text{L??}}\ \frac{d, e \in X;\ f \in Forest}{e \in trace(d, f) \ \Leftrightarrow\ (e = root(d, f) \vee is\text{-}before(d, e, f))}$$

$$\boxed{\text{L??}}\ \frac{e \in X;\ f \in Forest}{trace(e, f) \subseteq collect(f, root(e, f))}$$

$$\boxed{\text{L??}}\ \frac{e \in x;\ f, f' \in Forest;\ trace(e, f) \lhd f' = trace(e, f) \lhd f}{root(e, f') = root(e, f)}$$

$$\boxed{\text{L??}}\ \frac{c, d, e \in X;\ f \in Forest;\ \neg\, is\text{-}before(e, d, f);\ d \in trace(c, f)}{root(c, f \dagger \{d \mapsto e\}) = root(e, f)}$$

Answer 11.4.1 from page 275

$f_0 = \{\,\}$

$TEST\ (es: \mathbb{N}\text{-}\mathbf{set})\ r: \mathbb{B}$
$\mathbf{ext\ rd}\ f\ :\ Forest$
$\mathbf{post}\ r\ \Leftrightarrow\ \exists rt \in \mathbb{N} \cdot \forall e \in es \cdot root(e, f) = rt$

$EQUATE\ (es: \mathbb{N}\text{-}\mathbf{set})$
$\mathbf{ext\ wr}\ f\ :\ Forest$
$\mathbf{pre}\ es \neq \{\,\}$
$\mathbf{post}\ \exists c \in es \cdot f = \overleftrightarrow{f} \dagger \{root(e, \overleftrightarrow{f}) \mapsto root(c, \overleftrightarrow{f}) \mid e \in es \wedge root(e, \overleftrightarrow{f}) \neq root(c, \overleftrightarrow{f})\}$

$GROUP\ (e: \mathbb{N})\ r: \mathbb{N}\text{-}\mathbf{set}$
$\mathbf{ext\ rd}\ f\ :\ Forest$
$\mathbf{post}\ r = collect(root(e, f), f)$

The reverse search implied by *collect* can *not* be implemented efficiently.

A

Known Errors

A.1 Remaining in third printing

Page	Line	From	To
121	11	$ldbl(tl)$	tl
129	9	$\subset$	$\subseteq$
146	−6	note all	not all
163	11	$rs(i) = s_b(i)$	$rs(i) = s_a(i)$
189	−1	$retr((f_r(r))$	$retr(f_r(r))$
207	2 $X \times Bag \times \mathbb{N} \times Bag$ $X \times Bag \times \mathbb{N}$		
274	−5	$roots(f)$	$roots(\overleftrightarrow{f})$
274	−4	$roots(f)$	$roots(\overleftrightarrow{f})$

A.2 Extra errors in first and second printings

Page 3, replace:

E_1	E_2	$E_1 \Rightarrow E_2$	$\neg E_1$	$\neg E_2$	$\neg E_2 \Rightarrow \neg E_1$
true	**true**	**true**	**false**	**false**	**true**
true	**false**	**false**	**false**	**true**	**true**
false	**true**	**true**	**true**	**false**	**true**
false	**false**	**true**	**false**	**false**	**true**

with:

E_1	E_2	$E_1 \Rightarrow E_2$	$\neg E_1$	$\neg E_2$	$\neg E_2 \Rightarrow \neg E_1$
true	**true**	**true**	**false**	**false**	**true**
true	**false**	**false**	**false**	**true**	**false**
false	**true**	**true**	**true**	**false**	**true**
false	**false**	**true**	**true**	**true**	**true**

Page	Line	From	To
3	–6	$\neg E_1 \Rightarrow \neg E_2$	$\neg E_2 \Rightarrow \neg E_1$
66	9	$add\ (i\colon \mathbb{N}, j\colon \mathbb{N})\ \mathbb{N}$	$add\ (i\colon \mathbb{N}, j\colon \mathbb{N})\ r\colon \mathbb{N}$
66	18	$mult\ (i\colon \mathbb{N}, j\colon \mathbb{N})\ \mathbb{N}$	$mult\ (i\colon \mathbb{N}, j\colon \mathbb{N})\ r\colon \mathbb{N}$
81	–6	how the the	how the
102	–5	$merge(p)$	$merge(p, t)$
135	9	$GROUP\ (e\colon \mathbb{N})\ \mathbb{N}\text{-}\mathbf{set}$	$GROUP\ (e\colon \mathbb{N})\ r\colon \mathbb{N}\text{-}\mathbf{set}$
147	–5	$m_1 \dagger m_2 = m_2 \cup m_1$	$m_1 \dagger m_2 = m_1 \cup m_2$
149	16	$cno_1 \neq cno$	$cno_1 \neq cno_2$
155	–9	$f(i)Rf(1+1)$	$f(i)\ R\ f(i+1)$
166	18	$cons(e_1, \cdots (cons(e_n, [\,])))$	$cons(e_1, \cdots (cons(e_n, [\,])) \cdots)$
167	13	$s \curvearrowright [\,]$	$s \curvearrowright [\,] = s$
253	–13	$\overset{\Leftrightarrow}{m} + \overset{\Leftrightarrow}{n}$	$\overset{\Leftrightarrow}{m} * \overset{\Leftrightarrow}{n}$
253	–12	$\overset{\Leftrightarrow}{m} + \overset{\Leftrightarrow}{n}$	$\overset{\Leftrightarrow}{m} * \overset{\Leftrightarrow}{n}$
262	–1	$\{s \in p \mid e_1 \notin s \wedge e_2 \notin s\}$	$\{s \in \overset{\Leftrightarrow}{p} \mid e_1 \notin s \wedge e_2 \notin s\}$
262	–1	$\{\bigcup\{s \in p \mid e_1 \in s \vee e_2 \in s\}\}$	$\{\bigcup\{s \in \overset{\Leftrightarrow}{p} \mid e_1 \in s \vee e_2 \in s\}\}$
264	–7	$e_1 \notin s \vee e_2 \notin s$	$e_1 \notin s \wedge e_2 \notin s$
276	1	$a(e)$	$a[e]$
276	17	$root(v, \overset{\Leftrightarrow}{a})$	$root(\overset{\Leftrightarrow}{v}, a)$
277	5	$\mathbf{var}\ v\colon X_0$	$\mathbf{var}\ v\colon X$
277	–13	$\mathbf{ext\ rd}\ a\colon \mathbf{array}\ X\ \mathbf{to}\ X$	$\mathbf{ext\ rd}\ a\colon \mathbf{array}\ X\ \mathbf{to}\ X_0$
277	–10	$\mathbf{var}\ v_1, v_2\colon X_0;$	$\mathbf{var}\ v_1, v_2\colon X;$
278	2	$\mathbf{ext\ wr}\ a\colon \mathbf{array}\ X\ \mathbf{to}\ X$	$\mathbf{ext\ wr}\ a\colon \mathbf{array}\ X\ \mathbf{to}\ X_0$

B

Axiomatization of LPF

These axioms still require careful checking against [Che86] because some problems were found in the use of the *mural* system which suggest that the rules are not complete.

B.1 Basic Rules

$$\boxed{\vee\text{-}I}\ \frac{E_i}{E_1 \vee E_2}\ 1 \leq i \leq 2$$

$$\boxed{\vee\text{-}E}\ \frac{E_1 \vee E_2;\ E_1 \vdash E;\ E_2 \vdash E}{E}$$

$$\boxed{\neg\vee\text{-}I}\ \frac{\neg E_1;\ \neg E_2}{\neg (E_1 \vee E_2)}$$

$$\boxed{\neg\vee\text{-}E}\ \frac{\neg (E_1 \vee E_2)}{\neg E_i}\ 1 \leq i \leq 2$$

$$\boxed{\neg\neg\text{-}I/E}\ \frac{E}{\neg\neg E}$$

$$\boxed{contr}\ \frac{E_1;\ \neg E_1}{E_2}$$

$$\boxed{\neg\,\mathbf{true}\text{-}E}\ \frac{\neg\,\mathbf{true}}{E}$$

$$\boxed{\mathbf{true}\text{-}I}\ \frac{}{\mathbf{true}}$$

$$\boxed{\exists\text{-}I}\ \frac{s \in X;\ E(s/x)}{\exists x \in X \cdot E(x)}$$

$$\boxed{\exists\text{-}E}\ \frac{\exists x \in X \cdot E(x);\ y \in X, E(y/x) \vdash E_1}{E_1}\ y \text{ is arbitrary}$$

$$\boxed{\neg\exists\text{-}I}\ \frac{x \in X \vdash \neg E(x)}{\neg (\exists x \in X \cdot E(x))}$$

$$\boxed{\neg\,\exists\text{-}E}\ \frac{\neg\,(\exists x \in X \cdot E(x));\ s \in X}{\neg\, E(s/x)}$$

$$\boxed{\text{var-}I}\ \frac{}{x \in X}\ x \text{ is arbitrary, } X \neq \{\,\}$$

$$\boxed{=\text{t-}subs}\ \frac{s_1 = s_2;\, p}{p[s_2/s_1]}$$

$$\boxed{=\text{-}term}\ \frac{s \in X}{s = s}$$

$$\boxed{=\text{-}comp}\ \frac{s_1, s_2 \in X}{(s_1 = s_2) \vee \neg\,(s_1 = s_2)}$$

$$\boxed{=\text{-}contr}\ \frac{\neg\,(s = s)}{E}$$

$$\boxed{\Delta\text{-I}}\ \frac{E}{\Delta E}$$

$$\boxed{\Delta\text{-I}}\ \frac{\neg\, E}{\Delta E}$$

$$\boxed{\Delta\text{-E}}\ \frac{\Delta E;\, E \vdash E_1;\, \neg\, E \vdash E_1}{E_1}$$

$$\boxed{\neg\,\Delta\text{-I}}\ \frac{\Delta E \vdash E_1;\, \Delta E \vdash \neg\, E_1}{\neg\,\Delta E}$$

$$\boxed{\neg\,\Delta\text{-E}}\ \frac{\neg\,\Delta E \vdash E_1;\, \neg\,\Delta E \vdash \neg\, E_1}{\Delta E}$$

$$\boxed{==\text{-}reflx}\ \frac{}{s == s}$$

$$\boxed{==\text{-}subs}\ \frac{s_1 == s_2;\, p}{p[s_2/s_1]}$$

$$\boxed{\neg\ ==\text{-}I}\ \frac{s_1 == s_2 \vdash E;\, s_1 == s_2 \vdash \neg\, E}{\neg\,(s_1 == s_2)}$$

$$\boxed{\neg\ ==\text{-}E}\ \frac{\neg\,(s_1 == s_2) \vdash E;\, \neg\,(s_1 == s_2) \vdash \neg\, E}{s_1 == s_2}$$

$$\boxed{==\text{-}comm}\ \frac{s_1 == s_2}{s_2 == s_1}$$

$$\boxed{==\text{-}trans}\ \frac{s_1 == s_2;\, s_2 == s_3}{s_1 == s_3}$$

$$\boxed{== \Rightarrow =}\ \frac{s_1 == s_2;\ s_i i \in X}{s_1 = s_2}\ 1 \leq i \leq 2$$

B.2 Definitions of Other Connectives

$$\boxed{\textbf{false}\text{-defn}}\ \frac{\neg\, \textbf{true}}{\textbf{false}}$$

$$\boxed{\wedge\text{-}\mathit{defn}}\ \frac{\neg\,(\neg\, E_1 \vee \neg\, E_2)}{E_1 \wedge E_2}$$

$$\boxed{\Rightarrow\text{-}\mathit{defn}}\ \frac{\neg\, E_1 \vee E_2}{E_1 \ \Rightarrow\ E_2}$$

$$\boxed{\Leftrightarrow\text{-}\mathit{defn}}\ \frac{(E_1 \ \Rightarrow\ E_2) \wedge (E_2 \ \Rightarrow\ E_1)}{E_1 \ \Leftrightarrow\ E_2}$$

$$\boxed{\forall\text{-}\mathit{defn}}\ \frac{\neg\,(\exists x \in X \cdot \neg\, E(x))}{\forall x \in X \cdot E(x)}$$

C

Other Proofs

This appendix contains a collection of proofs for general interest.

C.1 Propositional Calculus

from	$E_1 \wedge E_2$	
1	E_1	$\wedge$-E(h)
2	E_2	$\wedge$-E(h)
infer	$E_2 \wedge E_1$	$\wedge$-I(2,1)

Commutativity of conjunctions

from	$(E_1 \wedge E_2) \wedge E_3$	
1	$E_1 \wedge E_2$	$\wedge$-E(h)
2	E_1	$\wedge$-E(1)
3	E_2	$\wedge$-E(1)
4	E_3	$\wedge$-E(h)
5	$E_2 \wedge E_3$	$\wedge$-I(3,4)
infer	$E_1 \wedge (E_2 \wedge E_3)$	$\wedge$-I(2,5)

from	$E_1 \wedge (E_2 \wedge E_3)$	
1	$(E_2 \wedge E_3) \wedge E_1$	$\wedge$-*comm*(h)
2	$E_2 \wedge (E_3 \wedge E_1)$	$\wedge$-*ass*(1)
3	$(E_3 \wedge E_1) \wedge E_2$	$\wedge$-*comm*(2)
4	$E_3 \wedge (E_1 \wedge E_2)$	$\wedge$-*ass*(3)
infer	$(E_1 \wedge E_2) \wedge E_3$	$\wedge$-*comm*(4)

Associativity of conjunctions

from $E_1 \Rightarrow E_2, E_2 \Rightarrow E_3$

1	$\neg E_1 \vee E_2$	$\Rightarrow$-*defn*(h)
2	**from** $\neg E_1$	
	infer $E_1 \Rightarrow E_3$	$\Rightarrow$ *vac-I*(h2)
3	**from** E_2	
3.1	E_3	$\Rightarrow$ *vac-E*(h3,h)
	infer $E_1 \Rightarrow E_3$	$\Rightarrow$ *vac-I*(3.1)

infer $E_1 \Rightarrow E_3$ $\vee$-*E*(1,2,3)

Proof of $\Rightarrow$-trans

from $E_1 \Leftrightarrow E_2, E_2 \Leftrightarrow E_3$

1	$(E_1 \Rightarrow E_2) \wedge (E_2 \Rightarrow E_1)$	$\Leftrightarrow$-*defn*(h)
2	$E_1 \Rightarrow E_2$	$\wedge$-*E*(1)
3	$E_2 \Rightarrow E_1$	$\wedge$-*E*(1)
4	$(E_2 \Rightarrow E_3) \wedge (E_3 \Rightarrow E_2)$	$\Leftrightarrow$-*defn*(h)
5	$E_2 \Rightarrow E_3$	$\wedge$-*E*(4)
6	$E_3 \Rightarrow E_2$	$\wedge$-*E*(4)
7	$E_1 \Rightarrow E_3$	$\Rightarrow$-*trans*(2,5)
8	$E_3 \Rightarrow E_1$	$\Rightarrow$-*trans*(6,3)
9	$(E_1 \Rightarrow E_3) \wedge (E_3 \Rightarrow E_1)$	$\wedge$-*I*(7,8)

infer $E_1 \Leftrightarrow E_3$ $\Leftrightarrow$-*defn*(9)

Proof of $\Leftrightarrow$-trans

from $\neg(E_1 \Leftrightarrow E_2)$

1	$\neg((\neg E_1 \vee E_2) \wedge (E_1 \vee \neg E_2))$	$\Leftrightarrow$-$defn$(h)
2	$\neg(\neg E_1 \vee E_2) \vee \neg(E_1 \vee E_2)$	deM(h1)
3	**from** $\neg(\neg E_1 \vee E_2)$	
3.1	$\quad \neg\neg E_1 \wedge \neg E_2$	deM(h3)
3.2	$\quad E_1 \wedge \neg E_2$	$\wedge$-$subs$(3.1,$\neg\neg$-E)
	infer $E_1 \wedge \neg E_2 \vee \neg E_1 \wedge E_2$	$\vee$-I(3.2)
4	**from** $\neg(E_1 \vee \neg E_2)$	
4.1	$\quad \neg E_1 \wedge \neg\neg E_2$	deM(h4)
4.2	$\quad \neg E_1 \wedge E_2$	$\wedge$-$subs$(4.1,$\neg\neg$-E)
	infer $E_1 \wedge \neg E_2 \vee \neg E_1 \wedge E_2$	$\vee$-I(4.2)
infer	$E_1 \wedge \neg E_2 \vee \neg E_1 \wedge E_2$	$\vee$-E(2,3,4)

from $E_1 \wedge \neg E_2 \vee \neg E_1 \wedge E_2$

1	**from** $E_1 \wedge \neg E_2$	
	infer $\neg(E_1 \Leftrightarrow E_2)$	$\neg\Leftrightarrow$-I(h1)
2	**from** $\neg E_1 \wedge E_2$	
	infer $\neg(E_1 \Leftrightarrow E_2)$	$\neg\Leftrightarrow$-I(h2)
infer	$\neg(E_1 \Leftrightarrow E_2)$	$\vee$-E(h,1,2)

Proofs of $\neg\Leftrightarrow$-E

	from $E_1 \Leftrightarrow (E_2 \Leftrightarrow E_3)$	
1	$E_1 \wedge (E_2 \Leftrightarrow E_3) \vee \neg E_1 \wedge \neg (E_2 \Leftrightarrow E_3)$	$\Leftrightarrow$-E(h)
2	**from** $E_1 \wedge (E_2 \Leftrightarrow E_3)$	
2.1	E_1	$\wedge$-E(h2)
2.2	$E_2 \Leftrightarrow E_3$	$\wedge$-E(h2)
2.3	$E_2 \wedge E_3 \vee \neg E_2 \wedge \neg E_3$	$\Leftrightarrow$-E(2.2)
2.4	**from** $E_2 \wedge E_3$	
2.4.1	E_2	$\wedge$-E(h2.4)
2.4.2	E_3	$\wedge$-E(h2.4)
2.4.3	$E_1 \Leftrightarrow E_2$	$\Leftrightarrow$-I(2.1,2.4.1)
	infer $(E_1 \Leftrightarrow E_2) \Leftrightarrow E_3$	$\Leftrightarrow$-I(2.4.3,2.4.2)
2.5	**from** $\neg E_2 \wedge \neg E_3$	
2.5.1	$\neg E_2$	$\wedge$-E(h2.5)
2.5.2	$\neg E_3$	$\wedge$-E(h2.5)
2.5.3	$\neg (E_1 \Leftrightarrow E_2)$	$\neg \Leftrightarrow$-I(2.1,2.5.1)
	infer $(E_1 \Leftrightarrow E_2) \Leftrightarrow E_3$	$\Leftrightarrow$-I(2.5.3,2.5.2)
	infer $(E_1 \Leftrightarrow E_2) \Leftrightarrow E_3$	$\vee$-E(2.3,2.4,2.5)
3	**from** $\neg E_1 \wedge \neg (E_2 \Leftrightarrow E_3)$	
3.1	$\neg E_1$	$\wedge$-E(h3)
3.2	$\neg (E_2 \Leftrightarrow E_3)$	$\wedge$-E(h3)
3.3	$E_2 \wedge \neg E_3 \vee \neg E_2 \wedge E_3$	$\neg \Leftrightarrow$-E(3.2)
3.4	**from** $E_2 \wedge \neg E_3$	
3.4.1	E_2	$\wedge$-E(h3.4)
3.4.2	$\neg E_3$	$\wedge$-E(h3.4)
3.4.3	$\neg (E_1 \Leftrightarrow E_2)$	$\neg \Leftrightarrow$-I(3.1,3.4.1)
	infer $(E_1 \Leftrightarrow E_2) \Leftrightarrow E_3$	$\Leftrightarrow$-I(3.4.3,3.4.2)
3.5	**from** $\neg E_2 \wedge E_3$	
3.5.1	$\neg E_2$	$\wedge$-E(h3.5)
3.5.2	E_3	$\wedge$-E(h3.5)
3.5.3	$E_1 \Leftrightarrow E_2$	$\Leftrightarrow$-I(3.1,3.5.1)
	infer $(E_1 \Leftrightarrow E_2) \Leftrightarrow E_3$	$\Leftrightarrow$-I(3.5.2,3.5.3)
	infer $(E_1 \Leftrightarrow E_2) \Leftrightarrow E_3$	$\vee$-I(3.3,3.4,3.5)
	infer $(E_1 \Leftrightarrow E_2) \Leftrightarrow E_3$	$\vee$-E(1,2,3)

Proof of $\Leftrightarrow$-*ass*

Notice that the reverse direction of $\Leftrightarrow$-*ass* follows by commutativity.

from	$E_1 \wedge \neg (E_2 \Leftrightarrow E_3)$	
1	E_1	$\wedge$-E(h)
2	$\neg (E_2 \Leftrightarrow E_3)$	$\wedge$-E(h)
3	$E_2 \wedge \neg E_3 \vee \neg E_2 \wedge E_3$	$\neg \Leftrightarrow$-E(2)
4	**from** $E_2 \wedge \neg E_3$	
4.1	E_2	$\wedge$-E(h4)
4.2	$\neg E_3$	$\wedge$-E(h4)
4.3	$E_1 \wedge E_2$	$\wedge$-I(1,4.1)
4.4	$\neg (E_1 \wedge E_3)$	$\neg \wedge$-I(4.2)
	infer $\neg (E_1 \wedge E_2 \Leftrightarrow E_1 \wedge E_3)$	$\neg \Leftrightarrow$-I(4.3,4.4)
5	**from** $\neg E_2 \wedge E_3$	
5.1	$\neg E_2$	$\wedge$-E(h5)
5.2	E_3	$\wedge$-E(h5)
5.3	$\neg (E_1 \wedge E_2)$	$\neg \wedge$-I(5.1)
5.4	$E_1 \wedge E_3$	$\wedge$-I(1,5.2)
	infer $\neg (E_1 \wedge E_2 \Leftrightarrow E_1 \wedge E_3)$	$\neg \Leftrightarrow$-I(5.3,5.4)
infer	$\neg (E_1 \wedge E_2 \Leftrightarrow E_1 \wedge E_3)$	$\vee$-E(3,4,5)

from	$\neg (E_1 \wedge E_2 \Leftrightarrow E_1 \wedge E_3)$	
1	$\neg (E_1 \wedge E_2) \wedge E_1 \wedge E_3 \vee E_1 \wedge E_2 \wedge \neg (E_1 \wedge E_3)$	$\neg \Leftrightarrow$-E(h)
2	**from** $\neg (E_1 \wedge E_2) \wedge E_1 \wedge E_3$	
2.1	$\neg (E_1 \wedge E_2)$	$\wedge$-E(h2)
2.2	$\neg E_1 \vee \neg E_2$	deM(2.1)
2.3	E_1	$\wedge$-E(h2)
2.4	E_3	$\wedge$-E(h2)
2.5	$\neg E_2$	$\Rightarrow$ *vac*-E(2.2,2.3)
2.6	$\neg (E_2 \Leftrightarrow E_3)$	$\neg \Leftrightarrow$-I(2.4,2.5)
	infer $E_1 \wedge \neg (E_2 \Leftrightarrow E_3)$	$\wedge$-I(2.3,2.6)
3	**from** $E_1 \wedge E_2 \wedge \neg (E_1 \wedge E_3)$	
3.1	E_1	$\wedge$-E(h3)
3.2	E_2	$\wedge$-E(h3)
3.3	$\neg (E_1 \wedge E_3)$	$\wedge$-E(h3)
3.4	$\neg E_1 \vee \neg E_3$	deM(3.3)
3.5	$\neg E_3$	$\Rightarrow$ *vac*-E(3.4,3.1)
3.6	$\neg (E_2 \Leftrightarrow E_3)$	$\neg \Leftrightarrow$-I(3.2,3.5)
	infer $E_1 \wedge \neg (E_2 \Leftrightarrow E_3)$	$\wedge$-I(3.1,3.6)
infer	$E_1 \wedge \neg (E_2 \Leftrightarrow E_3)$	$\vee$-E(1,2,3)

Proofs of $\wedge\neg \Leftrightarrow$-*dist*

from $\delta(E_1), E_1 \vee (E_2 \Leftrightarrow E_3)$		
1	$E_1 \vee \neg E_1$	h, δ
2	**from** E_1	
2.1	$E_1 \vee E_2$	$\vee$-I(h2)
2.2	$E_1 \vee E_3$	$\vee$-I(h2)
2.3	$(E_1 \vee E_2) \wedge (E_1 \vee E_3)$	$\wedge$-I(2.1,2.2)
	infer $E_1 \vee E_2 \Leftrightarrow E_1 \vee E_3$	$\Leftrightarrow$-I(2.3)
3	**from** $\neg E_1$	
3.1	$E_2 \Leftrightarrow E_3$	$\Rightarrow vac$-E(h3,h)
3.2	$E_2 \wedge E_3 \vee \neg E_2 \wedge \neg E_3$	$\Leftrightarrow$-E(3.1)
3.3	**from** $E_2 \wedge E_3$	
3.3.1	E_2	$\wedge$-E(h3.3)
3.3.2	E_3	$\wedge$-E(h3.3)
3.3.3	$E_1 \vee E_2$	$\vee$-I(3.3.1)
3.3.4	$E_1 \vee E_3$	$\vee$-I(3.3.2)
3.3.5	$(E_1 \vee E_2) \wedge (E_1 \vee E_3)$	$\wedge$-I(3.3.3,3.3.4)
	infer $E_1 \vee E_2 \Leftrightarrow E_1 \vee E_3$	$\Leftrightarrow$-I(3.3.5)
3.4	**from** $\neg E_2 \wedge \neg E_3$	
3.4.1	$\neg E_2$	$\wedge$-E(h3.4)
3.4.2	$\neg E_3$	$\wedge$-E(h3.4)
3.4.3	$\neg (E_1 \vee E_2)$	$\neg \vee$-I(h3,3.4.1)
3.4.4	$\neg (E_1 \vee E_3)$	$\neg \vee$-I(h3,3.4.2)
3.4.5	$\neg (E_1 \vee E_2) \wedge \neg (E_1 \vee E_3)$	$\wedge$-I(3.4.3,3.4.4)
	infer $E_1 \vee E_2 \Leftrightarrow E_1 \vee E_3$	$\Leftrightarrow$-I(3.4.5)
	infer $E_1 \vee E_2 \Leftrightarrow E_1 \vee E_3$	$\vee$-E(3.2,3.3,3.4)
infer $E_1 \vee E_2 \Leftrightarrow E_1 \vee E_3$		$\vee$-E(1,2,3)

or distributes over equivalence

from $\neg(E_1 \vee E_2 \Leftrightarrow E_1 \vee E_3)$		
1	$\neg(E_1 \vee E_2) \wedge (E_1 \vee E_3) \vee (E_1 \vee E_2) \wedge \neg(E_1 \vee E_3)$	
2	**from** $\neg(E_1 \vee E_2) \wedge (E_1 \wedge E_3)$	
2.1	$\quad \neg(E_1 \vee E_2)$	$\wedge$-E(h2)
2.2	$\quad \neg E_1 \wedge \neg E_2$	deM(2.1)
2.3	$\quad E_1 \vee E_3$	$\wedge$-E(h2)
2.4	$\quad$ **from** E_1	
	$\quad$ **infer** $E_1 \vee \neg(E_2 \Leftrightarrow E_3)$	$\vee$-I(h2.4)
2.5	$\quad$ **from** E_3	
2.5.1	$\qquad \neg E_2$	$\wedge$-E(2.2)
2.5.2	$\qquad \neg(E_2 \Leftrightarrow E_3)$	$\neg \Leftrightarrow$-I(h2.5,2.5.1)
	$\quad$ **infer** $E_1 \vee \neg(E_2 \Leftrightarrow E_3)$	$\vee$-I(2.5.2)
	infer $E_1 \vee \neg(E_2 \Leftrightarrow E_3)$	$\vee$-E(2.3,2.4,2.5)
3	**from** $(E_1 \vee E_2) \wedge \neg(E_1 \vee E_3)$	
	$\quad$ *similar*	
	infer $E_1 \vee \neg(E_2 \Leftrightarrow E_3)$	
infer $E_1 \vee \neg(E_2 \Leftrightarrow E_3)$		$\vee$-E(1,2,3)

Proof of $\vee\neg\Leftrightarrow$-*dist*

Notice, the converse of $\vee\neg\Leftrightarrow$-*dist* is false! (Consider $E_1 = t$).

C.2 Predicate Calculus

from $\neg \forall x \in X \cdot \neg p(x)$		
1	$\neg\neg \exists x \in X \cdot \neg\neg p(x)$	$\forall$-*defn*(1)
2	$\exists x \in X \cdot \neg\neg p(x)$	$\neg\neg$-E(2)
infer $\exists x \in X \cdot p(x)$		$\exists$-*subs*/$\neg\neg$-E(2)

Proofs of $\forall$-*deM*

from $x \in X \vdash p(x)$		
1	**from** $x \in X$	
1.1	$\quad p(x)$	h,h1
	infer $\neg\neg p(x)$	$\neg\neg$-I(1.1)
2	$\neg \exists x \in X \cdot \neg p(x)$	$\neg\exists$-I(1)
infer $\forall x \in X \cdot p(x)$		$\forall$-*defn*(2)

Proof of $\forall$-I

from	$\forall x \in X \cdot p(x); s \in X$	
1	$\neg \exists x \in X \cdot \neg p(x)$	$\forall$-*defn*(h)
2	$\neg \neg p(s/x)$	$\neg \exists$-E(h,1)
infer	$p(s/x)$	$\neg \neg$-E(2)

Proof of $\forall$-E

C.3 Non-monotonic part

Bibliography

[ACJ72] C. D. Allen, D. N. Chapman, and C. B. Jones. A formal definition of ALGOL 60. Technical Report 12.105, IBM Laboratory Hursley, August 1972.

[Acz82] P. Aczel. A note on program verification. Manuscript, Manchester, January 1982.

[AhK89] J. A. AhKee. *Operation Decomposition Proof Obligations for Blocks and Procedures.* PhD thesis, Manchester University, 1989.

[And87] D. Andrews. Data Reification and Program Decomposition. In D. Bjørner, C.B. Jones, M. Mac an Airchinnigh, and E.J. Neuhold, editors, *VDM '87: VDM — A Formal Method at Work (Proceedings of the VDM-Europe Symposium 1987, Belgium, March 1987)*, pages 389–422. Springer-Verlag, 1987. Lecture Notes in Computer Science, Vol. 252.

[Bac86] R.C. Backhouse. *Program Construction and Verification.* Prentice-Hall International, 1986.

[BCJ84] H. Barringer, J.H. Cheng, and C. B. Jones. A logic covering undefinedness in program proofs. *Acta Informatica*, 21:251–269, 1984.

[BCMS89] R. C. Backhouse, P. Chisholm, G. Malcolm, and E. Saaman. Do-it-Yourself type theory. *Formal Aspects of Computing*, 1:19–84, 1989.

[BIJW75] H. Bekič, H. Izbicki, C. B. Jones, and F. Weissenböck. Some experiments with using a formal language definition in compiler development. Laboratory Note LN 25.3.107, IBM Laboratory, Vienna, December 1975.

[BJ78] D. Bjørner and C. B. Jones, editors. *The Vienna Development Method: The Meta-Language*, volume 61 of *Lecture Notes in Computer Science*. Springer-Verlag, 1978.

[BJ82] D. Bjørner and C. B. Jones. *Formal Specification and Software Development.* Prentice Hall International, 1982.

[BJ84] H. Bekič and C. B. Jones, editors. *Programming Languages and Their Definition: Selected Papers of H. Bekič*, volume 177 of *Lecture Notes in Computer Science*. Springer-Verlag, 1984.

[BJM88] R. Bloomfield, R. B. Jones, and L. S. Marshall, editors. *VDM'88: VDM – The Way Ahead*, volume 328 of *Lecture Notes in Computer Science*. Springer-Verlag, 1988.

[BJMN87] D. Bjørner, C. B. Jones, M. Mac an Airchinnigh, and E. J. Neuhold, editors. *VDM – A Formal Definition at Work*, volume 252 of *Lecture Notes in Computer Science*. Springer-Verlag, 1987.

[Bjø77a] D. Bjørner. Programming languages: Formal development of interpreters and compilers. In *International Computing Symposium 77*, pages 1–21. European ACM, North-Holland Publ.Co., Amsterdam, 1977.

[Bjø77b] D. Bjørner. Programming languages: Linguistics and semantics. In *International Computing Symposium 77*, pages 511–536. European ACM, North-Holland Publ.Co., Amsterdam, 1977.

[Bjø79a] D. Bjørner. The systematic development of compiling algorithm. In Amirchahy and Neel, editors, *Le Point sur la Compilation*, pages 45–88. INRIA Publ. Paris, 1979.

[Bjø79b] D. Bjørner. *The Vienna Development Method: Software Abstraction and Program Synthesis*, volume 75: Math. Studies of Information Processing of *Lecture Notes in Computer Science*. Springer-Verlag, 1979.

[Bjo80a] D. Bjorner, editor. *Abstract Software Specifications*, volume 86 of *Lecture Notes in Computer Science*. Springer-Verlag, 1980.

[Bjø80b] D. Bjørner. Application of formal models. In *Data Bases*. INFOTECH Proceedings, October 1980.

[Bjø80c] D. Bjørner. Experiments in block-structured goto-modelling: Exits vs. continuations. *[Bjo80a]*, pages 216–247, 1980.

[Bjø80d] D. Bjørner. Formal description of programming concepts: a software engineering viewpoint. In *MFCS '80, Lecture Notes Vol. 88*, pages 1–21. Springer-Verlag, 1980.

[Bjø81] D. Bjørner. The VDM principles of software specification and program design. In *TC2 Work.Conf. on Formalization of Programming Concepts*, pages 44–74, LNCS Vol. 107, 1981. IFIP, Springer-Verlag.

[BL84] R. Bahlke and T. Letschert. Ausführbare denotationale semantik. In *Proc. 4*, pages 3–19. Gl-Fachgespräch Implementierung von Programmiersprachen, Zürich, 1984.

[BO80a] D. Bjørner and O.Oest. The DDC Ada compiler development project. *[BO80b]*, pages 1–19, 1980.

[BO80b] D. Bjørner and O.Oest, editors. *Towards a Formal Description of Ada*, volume 98 of *Lecture Notes in Computer Science*. Springer-Verlag, 1980.

[BS85] R. Bahlke and G. Snelting. The psg-system: From formal language definitions to interactive programming environments. Technical Report PÜ2R4/85, Fachbereich Informatik, Fachgebiet Programmiersprachen und Übersetzer II, Technische Hochschule Darmstadt, 1985.

[C+86] R.L. Constable et al. *Implementing Mathematics with the Nuprl Proof Development System*. Prentice-Hall, 1986.

[Che86] J. H. Cheng. *A Logic for Partial Functions*. PhD thesis, University of Manchester, 1986.

[CJ90] J.H. Cheng and C.B. Jones. On the usability of logics which handle partial functions. In *Proceedings of Third Refinement Workshop*, 1990.

[CKK87] C. Chedgey, S. Kearney, and H.J. Kugler. Using VDM in an Object-Oriented Development Method for Ada Software. In D. Bjørner, C.B. Jones, M. Mac an Airchinnigh, and E.J. Neuhold, editors, *VDM '87: VDM — A Formal Method at Work (Proceedings of the VDM-Europe Symposium 1987, Belgium, March 1987)*, pages 63–76. Springer-Verlag, 1987. Lecture Notes in Computer Science, Vol. 252.

[Cri87] R. J. Crispin. Experience using VDM in STC. In *[BJMN87]*, pages 19–32, 1987.

[DB80] O. Dommergaard and S. Bodilsen. A formal definition of p-code. Technical report, Dept. of Comp. Sci., Techn. Univ. of Denmark, 1980.

[Den86] T. Denvir. *Introduction to Discrete Mathematics for Software Engineering*. Macmillan Education Ltd, 1986.

[Dij76] E. W. Dijkstra. *A Discipline of Programming*. Prentice-Hall, 1976.

[Fan84] A. Fantechi. On combining meta-iv and ccs. Technical report, Dept. of Comp. Sci., Techn. Univ. of Denmark, May 1984.

[FB80] P. Folkjær and D. Bjørner. A formal model of a generalized csp-like language. In S.H. Lavington, editor, *Proc. IFIP'80*, pages 95–99. North-Holland Publ.Co., Amsterdam, 1980.

[Fie80] E. Fielding. The specification of abstract mappings and their implementation as B^+-trees. Technical Report PRG-18, Oxford University Computing Laboratory, Programming Research Group, September 1980.

[Fis84] Michael D. Fisher. An investigation into the use of edinburgh LCF in VDM data type refinement. Master's thesis, University of Manchester, Department of Computer Science, October 1984.

[Fle81] J. Flensholt. Conceptual graphs: A denotational semantics approach. Master's thesis, Dept. of Comp. Sci., Copenhagen Univ., December 1981.

[Geo87] C. George. Heap Storage Specification and Development. In D. Bjørner, C.B. Jones, M. Mac an Airchinnigh, and E.J. Neuhold, editors, *VDM '87: VDM — A Formal Method at Work (Proceedings of the VDM-Europe Symposium 1987, Belgium, March 1987)*, pages 97–105. Springer-Verlag, 1987. Lecture Notes in Computer Science, Vol. 252.

[GMW79] M. Gordon, R. Milner, and C. Wadsworth. *Edinburgh LCF*, volume 78 of *Lecture Notes in Computer Science*. Springer-Verlag, 1979.

[Gor79] M.J.C. Gordon. *The Denotational Description of Programming Languages*. Springer-Verlag, 1979.

[Gor88] M. J. C. Gordon. *Programming Language Theory and its Implementation*. Prentice-Hall International, 1988.

[Gri81] D. Gries. *The Science of Programming*. Springer-Verlag, 1981.

[Ham82] A. G. Hamilton. *Numbers, Sets and Axioms: the apparatus of Mathematics*. Cambridge University Press, 1982.

[Hay87] I. Hayes, editor. *Specification Case Studies*. Prentice-Hall International, 1987.

[Heh] E.C.R. Hehner. A simple view of variables and parameters.

[Hen70a] W. Henhapl. On the interpretation of GOTO statements in ULD. Technical Report LN 25.3.065, IBM Vienna, March 1970.

[Hen70b] W. Henhapl. A run-time mechanism for referencing variables. Technical Report LN 25.3.074, IBM Vienna, July 1970.

[Hen86] W. Henhapl. Generierung von programmierumgebunden – konzepte und erfahrungen. methoden und werkzeuge zur entwicklung von programmsystemen. Fachberichte und Referate Band 16, Oldenbourg, Fachbereich Informatik, Fachgebiet Programmeiersprachen und Übersetzer II, Technische Hochschule Darmstadt, 1985/1986.

[HJ71] W. Henhapl and C. B. Jones. A run-time mechanism for referencing variables. *Information Processing Letters*, 1(1):14–16, 1971.

[HJ89] I. J. Hayes and C. B. Jones. Specifications are not (necessarily) executable. *IEE, Software Engineering Journal*, 4(6):320–338, November 1989.

[HL85] W. Henhapl and Th. Letschert. Vdm in research, development and education: Local experiences formal models in programming. In *Formal Models in Programming, Work. Conf. The Role of Abstract Models in Information Processing, Wien 1985*, pages 157–180. IFIP, North-Holland Publ.Co., Amsterdam, 1985.

[HO87] P. Haff and A. Olsen. Use of VDM within CCITT. In D. Bjørner, C.B. Jones, M. Mac an Airchinnigh, and E.J. Neuhold, editors, *VDM '87: VDM — A Formal Method at Work (Proceedings of the VDM-Europe Symposium 1987, Belgium, March 1987)*, pages 324–330. Springer-Verlag, 1987. Lecture Notes in Computer Science, Vol. 252.

[Hon86] P.M.Y. Hong. The user interface of a proof editor. PhD. Conversion Report, University of Manchester, September 1986.

[HS84] W. Henhapl and G. Snelting. Context relations – a concept for incremental context analysis in program fragments. In *Proc. 8 GL-Fachtagung Programmiersprachen und Programmentwicklung*, pages 128–143, Informatik Fachberichte, Band 77, 1984. Springer-Verlag.

[Inc88] D.C. Ince. *An Introduction to Discrete Mathematics and Formal System Specification*. Oxford University Press, 1988.

[Izb75] H. Izbicki. On a consistency proof of a chapter of the formal definition of a pl/i subset. Technical Report 25.142, IBM Laboratory, Vienna, Feb. 1975.

[JL71] C. B. Jones and P. Lucas. Proving correctness of implementation techniques. In E. Engeler, editor, *A Symposium on Algorithmic Languages*, volume 188 of *Lecture Notes in Mathematics*, pages 178–211. Springer-Verlag, 1971.

[J.M81] J.Madsen. A computer system supporting data abstraction, pts. 1-2. *ACM SIGOPS*, 15(1-2):45–72, 38–78, 1981.

[Jon70a] C. B. Jones. A technique for showing that two functions preserve a relation between their domains. Technical Report LR 25.3.067, IBM Laboratory, Vienna, April 1970.

[Jon70b] C. B. Jones. Yet another proof of the block concept. Technical Report LN 25.3.075, IBM Laboratory, Vienna, August 1970.

[Jon72] C. B. Jones. Formal development of correct algorithms: an example based on Earley's recogniser. In *SIGPLAN Notices, Volume 7 Number 1*, pages 150–169. ACM, January 1972.

[Jon73] C. B. Jones. Formal development of programs. Technical Report 12.117, IBM Laboratory Hursley, June 1973.

[Jon75] C. B. Jones. Formal definition in program development. In *Programming Methodology*, volume 23 of *Lecture Notes in Computer Science*, pages 384–443. Springer-Verlag, 1975.

[Jon76] C. B. Jones. Formal definition in compiler development. Technical Report 25.145, IBM Laboratory Vienna, February 1976.

[Jon77a] C. B. Jones. Implementation bias in constructive specification of abstract objects. typescript, September 1977.

[Jon77b] C. B. Jones. Program specification and formal development. In E. Morlet and D. Ribbens, editors, *International Computing Symposium 1977*, pages 537–553. North-Holland, 1977.

[Jon77c] C. B. Jones. Structured design and coding: Theory versus practice. *Informatie*, 19(6):311–319, June 1977.

[Jon78] C. B. Jones. The meta-language: A reference manual. In *[BJ78]*, pages 218–277. Springer-Verlag, 1978.

[Jon79a] C. B. Jones. Constructing a theory of a data structure as an aid to program development. *Acta Informatica*, 11:119–137, 1979.

[Jon79b] C. B. Jones. The Vienna Development Method: Examples of compiler development. In M. Amirchahy and D. Neel, editors, *Le Point sur la Compilation*, pages 89–114. IRIA-SEFI, 1979.

[Jon80a] C. B. Jones. Models of programming language concepts. In *Abstract Software Specifications*, volume 86 of *Lecture Notes in Computer Science*, pages 100–143. Springer-Verlag, 1980.

[Jon80b] C. B. Jones. *Software Development: A Rigorous Approach*. Prentice Hall International, 1980. ISBN 0-13-821884-6.

[Jon81a] C. B. Jones. *Development Methods for Computer Programs including a Notion of Interference*. PhD thesis, Oxford University, June 1981. Printed as: Programming Research Group, Technical Monograph 25.

[Jon81b] C. B. Jones. Specification as a design base. In *Trends in Information Processing*, volume 123 of *Lecture Notes in Computer Science*, pages 103–105. Springer-Verlag, 1981.

[Jon81c] C. B. Jones. Towards more formal specifications. In C. Floyd and H. Kopetz, editors, *Software Engineering – Entwurf und Specifikation*, pages 19–45. Teubner Verlag, 1981.

[Jon83a] C. B. Jones. Specification and design of (parallel) programs. In *Proceedings of IFIP'83*, pages 321–332. North-Holland, 1983.

[Jon83b] C. B. Jones. Tentative steps toward a development method for interfering programs. *ACM Transactions on Programming Languages and Systems*, 5(4):596–619, 1983.

[Jon85a] C. B. Jones. The role of proof obligations in software design. In *Formal Methods and Software Development*, volume 186 of *Lecture Notes in Computer Science*, pages 27–41. Springer-Verlag, 1985.

[Jon85b] C. B. Jones. Specification, verification and testing in software development. In T. Anderson, editor, *Software Requirements Specification and Testing*, pages 1–13. Blackwell Scientific Publications, 1985.

[Jon85c] K.D. Jones. *The Application of a Formal Development Method to a Parallel Machine Environment.* PhD thesis, University of Manchester, October 1985.

[Jon86a] C. B. Jones. Program specification and verification in VDM. Technical Report UMCS 86-10-5, University of Manchester, 1986. extended version of [?] (includes the full proofs).

[Jon86b] C. B. Jones. Proof obligations for data reification. In O. G. Folberth and C. Hackl, editors, *Der Informationsbegriff in Technik und Wissenschaft*, pages 77–96. Oldenbourg Verlag, 1986. Festschrift–Prof. Karl E. Ganzhorn.

[Jon86c] C. B. Jones. Systematic program development. In J. W. de Bakker, M. Hazewinkel, and J. K. Lenstra, editors, *Mathematics and Computer Science*, pages 19–50. North-Holland, 1986.

[Jon86d] C. B. Jones. *Systematic Software Development Using VDM.* Prentice Hall International, 1986.

[Jon86e] C. B. Jones. Teaching notes for systematic software development using VDM. Technical Report UMCS 86-4-2, University of Manchester, 1986.

[Jon87a] C. B. Jones. VDM proof obligations and their justification. In *[BJMN87]*, pages 260–286. Springer-Verlag, 1987.

[Jon87b] K. D. Jones. A formal semantics for a DataFlow Machine – using VDM. In *[BJMN87]*, pages 331–355. 1987.

[Jon87c] K.D. Jones. Support Environments for VDM. In D. Bjørner, C.B. Jones, M. Mac an Airchinnigh, and E.J. Neuhold, editors, *VDM '87: VDM — A Formal Method at Work (Proceedings of the VDM-Europe Symposium 1987, Belgium, March 1987)*, pages 110–117. Springer-Verlag, 1987. Lecture Notes in Computer Science, Vol. 252.

[Jon90] C. B. Jones. *Systematic Software Development using VDM.* Prentice Hall International, second edition, 1990. ISBN 0-13-880733-7.

[JS90] C. B. Jones and R. C. F. Shaw, editors. *Case Studies in Systematic Software Development.* Prentice Hall International, 1990. ISBN 0-13-116088-5.

[LB80] H.H. Løvengreen and D. Bjørner. On a formal model of the tasking concepts in Ada. In *ACM SIGPLAN Ada Symp.*, Boston, 1980.

[Let84] Th. Letschert. Language implementation as data type refinement. Technical Report PÜ2R7/84, Fachbereich Informatik, fachgebiet Programmiersprachen und Übersetzer II, Technische Hochschule Darmstadt, 1984.

[Let87] T. Letschert. VDM as a Specification Method for Telecommunications Software. In D. Bjørner, C.B. Jones, M. Mac an Airchinnigh, and E.J. Neuhold, editors, *VDM '87: VDM — A Formal Method at Work (Proceedings of the VDM-Europe Symposium 1987, Belgium, March 1987)*, pages 106–109. Springer-Verlag, 1987. Lecture Notes in Computer Science, Vol. 252.

[Lin81] J. Lindenau. Eine deskriptive anfragesprache für das netzwerk-datenmodell mit formaler definition der semantik in meta-iv. Master's thesis, Inst. f. Informatik, Christian-Albrechts-Univ., Kiel, March 1981.

[Luc68] P. Lucas. Two constructive realizations of the block concept and their equivalence. Technical Report TR 25.085, IBM Laboratory Vienna, June 1968.

[Luc73] P. Lucas. On program correctness and the stepwise development of implementations. In *Proc. Convegno di Informatica Teorica*, pages 219–251, Italy, March 1973. Univ. of Pisa.

[Luc78] P. Lucas. On the formalization of programming languages: Early history and main approaches. In *[Bjø79a]*. INRIA Publ. Paris, 1978.

[Luc80] P. Lucas. On the structure of application programs. In *[Bjø80c]*. Springer-Verlag, Lecture Notes in Computer Science, Vol. 86, 1980.

[Luc87] P. Lucas. VDM: Origins, Hopes, and Achievements. In *[BJMN87]*, pages 1–18, 1987.

[Lun89] R. Lunnon. Transfer report: A general induction schema with necessary and sufficient conditions for its validity, February 16 1989.

[Mac87a] M. Mac an Airchinnigh. Introduction to the VDM Tutorial. In D. Bjørner, C.B. Jones, M. Mac an Airchinnigh, and E.J. Neuhold, editors, *VDM '87: VDM — A Formal Method at Work (Proceedings of the VDM-Europe Symposium 1987, Belgium, March 1987)*, pages 356–361. Springer-Verlag, 1987. Lecture Notes in Computer Science, Vol. 252.

[Mac87b] M. Mac an Airchinnigh. Mathematical Structures and their Morphisms in Meta-IV. In D. Bjørner, C.B. Jones, M. Mac an Airchinnigh, and E.J. Neuhold, editors, *VDM '87: VDM — A Formal Method at Work (Proceedings of the VDM-Europe Symposium 1987, Belgium, March 1987)*, pages 287–320. Springer-Verlag, 1987. Lecture Notes in Computer Science, Vol. 252.

[Mac87c] M. Mac an Airchinnigh. Specification by Data Types. In D. Bjørner, C.B. Jones, M. Mac an Airchinnigh, and E.J. Neuhold, editors, *VDM '87: VDM — A Formal Method at Work (Proceedings of the VDM-Europe Symposium 1987, Belgium, March 1987)*, pages 362–388. Springer-Verlag, 1987. Lecture Notes in Computer Science, Vol. 252.

[Mad77] J. Madsen. An experiment in formal definition of operating system facilities. *IPL*, 6(6):187–189, Dec. 1977.

[Mad80] J. Madsen. *Modular Operating System Design*. PhD thesis, Dept. of Comp.Sci., Tech. Univ. of Denmark, Rept.no. ID805, Aug. 1980.

[Mar85] L. S. Marshall. A formal specification of line representations on graphics devices. In H. Ehrig et al., editors, *Formal Methods and Software Development. Proceedings of the International Joint Conference on Theory and Practice of Software Development. (TAPSOFT), Berlin, March 1985. Volume 2: Colloqium on Software Engineering (CSE)*, volume 186 of *Lecture Notes in Computer Science*. 1985.

[MH87] C. Minkowitz and P. Henderson. A Formal Description of Object-Oriented Programming using VDM. In D. Bjørner, C.B. Jones, M. Mac an Airchinnigh, and E.J. Neuhold, editors, *VDM '87: VDM — A Formal Method at Work (Proceedings of the VDM-Europe Symposium 1987, Belgium, March 1987)*, pages 237–259. Springer-Verlag, 1987. Lecture Notes in Computer Science, Vol. 252.

[Mon87] B. Q. Monahan. A Type Model for VDM. In *[BJMN87]*, pages 210–236. 1987.

[Mor86] T. Morgan. The semantics of logic programming. PhD. Conversion Report, University of Manchester, September 1986.

[MRW+84] M.Jäger, R.Bahlke, W.Henhapl, M.Hunkel, Th.Letschert, and G.Snelting. Psg-programming system generator. In *Proc. Programmierumgebungen und Compiler, München 1984*, pages 285–291, Berichte German Chapter ACM, Band 18, 1984. Teubner Verlag.

[Nip86] T. Nipkow. Non-deterministic data types: Models and implementations. *Acta Informatica*, 22:629–661, 1986.

[Nip87] T. Nipkow. *Behavioural Implementation Concepts for Nondeterministic Data Types.* PhD thesis, University of Manchester, May 1987.

[NS85] W.H. Newton-Smith. *Logic: An Introductory Course.* Routledge and Kegan Paul, 1985.

[Oln81] Ths. Olnhoff. *Funktionale Semantikbeschreibung von Anfrageoperationen in einem drei-schictigen relationaler Datenbanksystem.* PhD thesis, Inst. f. Informatik, Stuttgart/Hamburg Univ., May 1981.

[Pau87] L.C. Paulson. *Logic and Computation: Interactive Proof with Cambridge LCF.* Cambridge University Press, 1987.

[Ped87] J. S. Pedersen. VDM in three generations of Ada formal descriptions. In *[BJMN87]*, pages 33–48, 1987.

[Pre87] S. Prehn. From VDM to RAISE. In D. Bjørner, C.B. Jones, M. Mac an Airchinnigh, and E.J. Neuhold, editors, *VDM '87: VDM — A Formal Method at Work (Proceedings of the VDM-Europe Symposium 1987, Belgium, March 1987)*, pages 141–150. Springer-Verlag, 1987. Lecture Notes in Computer Science, Vol. 252.

[Ras90] A. Rasmussen. VDM bibliography. Technical Report ??, Technical University of Denmark, 1990.

[Rey81] J. C. Reynolds. *The Craft of Programming.* Prentice Hall International, 1981.

[RG85] R.Bahlke and G.Snelting. The psg-programming system generator. *ACM SIGPLAN Notices*, 20(7):28–33, 1985.

[rS87] D. Bjørner and J Storbank Pedersen. The draft formal definition of ada — the rôle and scope of the formal definition of ada, March 1987. Draft.

[SC87] R.G. Stone and D.J. Cooke. *Program Construction.* Cambridge University Press, 1987.

[Sch86] D.A. Schmidt. *Denotational Semantics: a Methodology for Language Development.* Allyn & Bacon, 1986.

[Sen87] D. Sen. Objectives of the British Standardisation of a Language to support the Vienna Development Method — The BSI VDM Specification Language Standarisation Panel — United Kingdom. In D. Bjørner, C.B. Jones, M. Mac an Airchinnigh, and E.J. Neuhold, editors, *VDM '87: VDM — A Formal Method at Work (Proceedings of the VDM-Europe Symposium 1987, Belgium, March 1987)*, pages 321–323. Springer-Verlag, 1987. Lecture Notes in Computer Science, Vol. 252.

[Sne84] G. Snelting. Die spezifikationssprache für kontextbedingungen im psg-system. Technical Report PÜ2R1/84, Fachbereich Informatik, Fachgebiet Programmiersprachen und Übersetzer II, Technische Hochschule Darmstadt, 1984.

[Sne85] G. Snelting. Experiences with psg – programming system generator. In *Vol. 186 of Lecture Notes in Computer Science: Proc. Joint Conf. on Theory and Practice of Software Development, Berlin 1985*, pages 148–162. Springer-Verlag, 1985.

[Sne86] G. Snelting. *Inkrementelle Semantische Analyse in unvollständigen Programmfragmenten mit Kontextrelationen.* PhD thesis, Fachbereich Informatik, Technische Hochschule Darmstadt, 1986.

[Spi88] J.M. Spivey. *Understanding Z—A Specification Language and its Formal Semantics.* Cambridge Tracts in Computer Science 3. Cambridge University Press, 1988.

[SV81] U. Schmidt and U. Völler. Die formale entwicklung der maschin-unabhängigen zwischensprache cat. *Informatik Fachberichte*, GI-1 Jahrestagung:57–64, 1981.

[Vad86] S. Vadera. A theory of unification. Master's thesis, Department of Computer Science, University of Manchester, July 1986.

[Wei75] F. Weissenböck. A formal interface specification. Technical Report 25.141, IBM Laboratory, Vienna, Feb. 1975.

[Wel82] A. Welsh. The specification, design and implementation of NDB. Master's thesis, University of Manchester, October 1982.

[Wel84] A. Welsh. *A Database Programming Language: Definition, Implementation and Correctness Proofs.* PhD thesis, Department of Computer Science, University of Manchester, October 1984. Also published as technical report UMCS-84-10-1.

[WH75] F. Weissenböck and W. Henhapl. A formal mapping description. Technical Report 25.105, IBM Laboratory, Vienna, Feb. 1975.

[WL88] J. Woodcock and M. Loomes, editors. *Software Engineering Mathematics.* Pitman, 1988.