Note on Prefix Operators

R.D. Arthan
Lemma 1 Ltd.
2nd Floor, 31A Chain Street,
Reading UK RG1 2HX
rda@lemma-one.com

22 May 2000

Z Standards Document References

Version 1	19th March 2000	N/A
Version 2	22nd May 2000	D-257

1 Introduction

This note responds to an action at ZSRC meeting 54 to look into the use of prefix operator templates in the Z mathematical toolkit. The note was discussed at the subsequent meeting. This resulted in a slight change to the end section 4 — the previous version did not take into account non-applicative uses of functions such as *dom* etc.

Comment UK103 observes that arithmetic negation has been assigned a precedence of 50, which is at odds with the other prefix operators. In discussing this at the meeting, it was pointed out that prefix operators are defined so that expressions like '$- -x$' and '$-f\ x$' parse as one would like them to: '$-(-x)$' and '$-(f\ x)$'. However, there are several functions that have been widely used in the literature as if they were prefix operators, which are not defined as such in the CD (or in Spivey). Examples include '$\#$' in '$\# \bigcup X$' and '*head*' in '$head\ tail\ s$'. This note reports on a brief investigation of this issue.

2 Existing Prefix Operators

Section 11 and Appendix B of the 1999 CD defines the prefix operators listed in the following table:

Operator	Precedence
$\mathbb{P}$	80
$\mathbb{P}_1$	80
$\mathbb{F}$	80
$\mathbb{F}_1$	80
id	80
$-$	50
seq	80
seq_1	80
$iseq$	80

3 Precedence of Arithmetic Negation

Examination of the precedences of the various infix and postfix operators indicates that it is just an oddity to take arithmetic negation with precedence 50. It is recommended that the precedence be changed to 80 to agree with all the other prefix operators. The basis for this recommendation is as follows:

- With the existing toolkit definitions, expressions affected by changing the precedence to 80 are ill-typed whichever way the precedences go, e.g., '$-x \lhd y$'.
- The change would make the CD uniform with Spivey, who defines arithmetic negation as a special case in the grammar at the same level as power set and prefix generic symbols.
- With this change, there is a simple *aide-memoire*: all postfix operators have higher precedence than all prefix operators, which, in turn, have higher precedence than all infix operators.

4 New Candidates for Prefix Operator Status

The toolkit functions which might arguably be changed to prefix operators are listed in the following table. Along with each operator, I have given the signature constraint of the function. I have also categorised the operators into groups according to my views about them as described after the table.

Function	Signature Constraint	Group
$\bigcup$	$\mathbb{P}\mathbb{P}X \to \mathbb{P}X$	B
$\bigcap$	$\mathbb{P}\mathbb{P}X \to \mathbb{P}X$	B
first	$X \times Y \to X$	D
second	$X \times Y \to Y$	D
dom	$(X \leftrightarrow Y) \to \mathbb{P}X$	B
ran	$(X \leftrightarrow Y) \to \mathbb{P}Y$	B
succ	$\mathbb{N} \to \mathbb{N}$	A
$\#$	$\mathbb{F}X \to \mathbb{N}$	A
min	$\mathbb{P}\mathbb{A} \nrightarrow \mathbb{A}$	A
max	$\mathbb{P}\mathbb{A} \nrightarrow \mathbb{A}$	A
items	$(L \twoheadrightarrow\!\!\!\!\!\!\not\;\; X) \to X \nrightarrow \mathbb{N}_1$	E
rev	$seq\ X \to seq\ X$	C
hd	$seq_1\ X \to X$	D
last	$seq_1\ X \to X$	D
tail	$seq_1\ X \to seq\ X$	C
front	$seq_1\ X \to seq\ X$	C
squash	$(\mathbb{Z} \twoheadrightarrow\!\!\!\!\!\!\not\;\; X) \to seq\ X$	C
$\frown/$	$seq\ seq\ X \to seq\ X$	C

The question is for each of these functions, 'F', say, do we want '$F\ x\ y$' to parse as '$F\ (x\ y)$' or as '$(F\ x)\ y$'. Put another way, we are asking whether it makes sense to treat '$F\ x$' as a function and apply it to y.

My views on this are as follows, group by group:

Group A: These are functions that return numbers. It is always ill-typed to use a number as a function, so the case for treating these functions as prefix operators is completely compelling and gives the only useful reading of expressions such as '$\# \bigcup A$' or '$min\ ran\ f$'.

Group B: These are functions that can return functions but almost never do in practice. I suspect that very few existing specifications would be influenced by the change to prefix operator status. The change would give what I think is the natural reading of: '$\bigcup \bigcup X$' or '$dom \bigcup X$'.

Group C: These are the functions that return sequences (which definitely are functions). Given a free rein, I would much prefer brackets to be needed in '*rev s 2*' rather than '*tail* (*front s*)'. I.e., I would rather the syntax encouraged dealing with sequences abstractly using the sequence operators. Unfortunately, I suspect quite a few existing specifications would be impacted by the change to prefix operator status for the functions in this group.

Group D: These are functions that return a value whose type is given as one of the generic formal arguments of the function. While it would be quite nice not to need brackets in things like '$-(hd\ s)$', I cannot think of any very strong justification for the change (although I would rather not have to put the brackets in in '*last* (*front s*)'). I suspect that not many specifications would be impacted.

Group E: This group comprises the function *items* which always returns a function. I cannot see any justification for changing *items* into a prefix operator.

The change to prefix operator status will affect any specification that has a non-applicative use of the function in question (e.g., "$x \in dom\ succ$" would have to be changed to "$x \in dom\ (succ\_)$").

Given the considerable amount of work that is still required to address the other comments, my overall recommendation is to make the change for group A operators only. This change legitimises several useful and natural forms of expression. Specifications that have non-applicative uses of the group A operators are rare, and the change to bring them into line is very easy.

5 Miscellaneous Remarks

- It seems bizarre that number range and '#' are defined in the sequences section. They should be in the integers section.
- My current reading of the CD is that '$f\ \ -x$' is correct and parses as '$f(-x)$', i.e., apply function f to the negation of x. Is that the intention?