

Language Manual for FPL

1. Introduction.

In this manual we shall be describing a new function level programming language called FPL (Functional Programming Language). The first main section gives an overview of the language along with the motivation for its features and a number of examples. The next section following this overview is a terse reference section; it describes the syntax and semantics of a subset of the complete language called the *proper language*. This is followed by a detailed explanation of the features of the proper language with examples of their use.

The next section [another reference section] describes the syntax of the complete language and translation rules for transforming any complete language program into a proper language program. Thus the complete language is essentially a "sugaring" of the proper language that contains a number of convenient abbreviations for expressions of the proper language as well as all the expressions of the proper language itself. This is followed by a detailed explanation of the meaning of the various complete language expressions and examples of their use.

The complete language described in this manual will use a large character set. We shall refer to this language as the *reference language*. We recognize that in some contexts it will not be practical to use such a large character set; therefore we provide a transliteration of reference language symbols into others that require only the ASCII character set. This transliteration will transform a reference language program into an ASCII language program. The intent is that it will be relatively easy to understand programs using only ASCII characters if one understands programs in the reference language.

This manual has the following main sections:

- 1) Overview, motivation and examples of the complete FPL language.
- 2) The syntax and semantics of the proper language.
- 3) Explanation and examples of the proper language.

- 4) The syntax of the complete language and its translation into the proper language.
- 5) Explanation and examples of the complete language.

2. Overview, motivation and examples of the complete FPL language.

.

-----to be done later-----

3. The syntax and semantics of the proper language.

3.1 Introduction.

In this section we describe the syntax and semantics of expressions in the proper language. In the course of doing so we indicate the complete language syntax for the synonyms for some of the most elementary constructs such as atoms and strings. [Since the complete language contains all proper language expressions, the expressions in the complete language that do not belong to the complete language will have proper language synonyms that they translate into.]

3.1.1 Notation used for BNF syntax productions.

In describing the syntax of constructs, we shall use these conventions:

- 1) Metalinguistic variables are written in italics.
- 2) The symbol $\equiv$ separates the left side of a production from the right.
- 3) The vertical bar $|$ is used as "or".
- 4) Parentheses are used for grouping; parentheses followed by an asterisk denotes "zero or more instances" of the constructs denoted within them. Two asterisks denotes "one or more instances".
- 5) Expressions are separated by spaces.
- 6) All symbols, other than those few reserved for productions, denote

themselves. Reserved symbols that must be denoted are written out in italics; eg *space* denotes a single space, *left-parenthesis* denotes the symbol (.

- 7) A variable followed by parentheses enclosing a capitalized italic letter is used to denote a schema in which the capital letter can be replaced by various syntactic variables. Thus *sequence(X)* might be defined and later we might refer to *sequence(atom)*, meaning "sequence of atoms" in which *X* is replaced by *atom*.
- 8) Italic capital words are used for comments or self-explanatory operations [eg *MINUS* is the subtraction operation for sets].

3.1.1.1 Examples.

$$abc(X) ::= | abc(X) (,X)^*$$

This production defines *abc(letter)* to contain the empty string or any string of one or more letters separated by commas. Similarly, it defines *abc(digit)* to contain strings of digits separated by commas.

$$cde ::= " (a | b | c \text{ space } d \text{ vertical-bar } e) "$$

means that *cde* denotes the set of strings containing "a" and "b" and "c d|e".

3.2 Objects.

3.2.1 Basic character set.

The following list contains all the *basic characters* [we reserve the term "character" for certain atoms, see below] used in the [proper and complete] reference language:

Digits: 0 1 2 3 4 5 6 7 8 9

Italic digits: 0 1 2 3 4 5 6 7 8 9

Letters: a b c d e f g h i j k l m n o p q r s t u v w x y z

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Italic letters: a b c d e f g h i j k l m n o p q r s t u v w x y z

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

[greek letters, lower and upper case]

^ ! @ # \$ % ^ & * _ = \ |

± Δ ∇ ~ + - × ÷ √ ~ ° ↑ ↓ ← → ⊂ ∩ ⊃ ⊄ ≤ ≥ ∈ ≡ /

() [] { } () []

. , ; ? ' " :

space [blank]

3.2.2 Atoms.

An *atom* is a sequence of ~~basic~~ ^{basic} characters that begins and ends with the character ^{basic} "!". This sequence must not contain any of the four special characters ^{basic} ! ' " \

! ' " \

unless each one is preceded by a single "\" character. The backslash character is used as an escape character; thus \! denotes the character ! within an atom and \\ denotes the character \ etc. All other characters denote themselves. [In the complete language a sequence of only *italic* letters and digits denotes the atom comprising the same sequence of roman characters enclosed by !'s, eg, the atom ab13 in the complete language is an abbreviation or synonym for !ab13!]

3.2.2.1 Syntax of atoms.

basic-character $\equiv$ THE LIST OF BASIC CHARACTERS ABOVE

special-character $\equiv$! | ' | " | \

escape-pair $\equiv$ \ special-character

atom-character $\equiv$ basic-character MINUS special-character

proper-atom $\equiv$! (atom-character | escape-pair)** !

complete-atom $\equiv$ proper-atom | (italic-letter | italic digit)**

3.2.2.2 Examples.

!This is an atom!	denotes This is an atom
![12] + ab!	denotes [12] + ab
! ! !	denotes three blank spaces
!\!)!	denotes !)
!,\'\! !	denotes ,'\
!\! ! !	not a well-formed atom
!ax3b	stands for the proper atom !!ax3b! in complete language

3.2.3 Characters.

A character is an atom with only one ^{basic} ~~basic character~~ in it [not counting the escape character "\"].

3.2.3.1 Syntax of characters.

proper-character $\equiv$! (atom-character | escape-pair) !

complete-character $\equiv$ proper-character | italic-letter | italic-digit

3.2.3.2 Examples.

!x!	denotes x
!\! !	denotes \
!\! ! !	denotes !
b	stands for the proper character !b! in the complete language

3.2.4 Sequences.

If $x_1, \dots, x_n$ are n expressions [atoms are objects and objects are expressions], then

$$\langle x_1, \dots, x_n \rangle$$

is a sequence [of length n]. A sequence of expressions is an expression; a sequence of objects is an object.

3.2.4.1 Syntax of sequences.

```
spaces := (space)*
row(X) := | X ( , spaces X)*
sequence(X) := < row(X) >
```

where X stands for various syntactic classes such as *proper-character*; thus *sequence(proper-character)* denotes the class of all sequences of proper characters.

3.2.4.2 Examples.

$\langle \rangle$	the empty sequence
$\langle !a!, !bcd!, !4! \rangle$	this sequence of atoms could be written as $\langle a, bcd, 4 \rangle$ in the complete language

3.2.5 Strings.

A *string* is a sequence of characters. In the proper language it must be written as such; in the complete language it can be written using any characters [with the use of $\backslash$ to denote the special characters (! ' " \)] enclosed in single quotes (')

3.2.5.1 Syntax of strings.

proper-string $\equiv$ sequence(proper-characters)
 complete-string $\equiv$ proper-string | '(atom-character | escape-pair)*'

3.2.5.2 Examples.

Proper language string	Complete language synonym string
<!a!, !\! , !3!>	'a\'!3'
<>	' '
<!A!, !b!>	'Ab'

3.3 Function names.

Primitive and user-defined functions are denoted by *function names*. In the proper language these function names are strings of characters enclosed in double quotes [again, the \ escape character is used to denote any one of the four special characters: ! ' " \]. In the complete language the quotes can be omitted if the string of characters contains only letters and digits.

3.3.1 Syntax of function names.

proper-function-name $\equiv$ " (atom-character | escape-pair)** "
 complete-function-name $\equiv$ proper-function-name | (letter | digit)**

3.3.2 Examples.

Proper function name	Complete language synonym
"ab12"	ab12
"A→B"	"A→B"

3.4 Patterns.