BETA Quick Reference Card

A summary of all special characters in BETA, and a short list of the syntax of the language is given below along with a short description of their semantics:

Special characters	Semantics
`:`	Declaration
`: @`	Static object reference declaration
`: ^`	Dynamic object reference declaration
`: ##`	Pattern reference declaration
`: @|`	Static component declaration
`: ^|`	Dynamic component declaration
`: [range]`	Declaration of repetition range must be an integer evaluation
`:<`	Virtual declaration
`::<`	Extended binding of virtual declaration
`::`	Final binding of virtual declaration
`&`	Dynamic creation of item; new
`&|`	Dynamic creation of component
`->`	Assignment
`[]`	Reference
`##`	Pattern reference
`(#`	Object descriptor begin
`#)`	Object descriptor end
`//`	Selection in if-imperative

Keywords
`do else enter exit inner leave none repeat restart suspend then this (if if) (for for)` Additional keywords (for their usage, see below)

Short syntax	Semantics
`P: (# … do … #)`	Definition of a pattern
`PP: P(# … do … #)`	Definition of a subpattern
`enter …`	Specification of enter-parameters
`exit …`	Specification of exit-parameters
`inner P`	Execute the actions in the subpattern. P is an optional name of an enclosing pattern.
`this(P)`	Denotation of this object
`this(P)[]`	Reference to this object
`E.P`	Remote name
`(E).P`	Computed remote name
`L: Imp`	In action part: labelled imperative
`L: (# … do … #)`	In action part: labelled imperative (descriptor)
`leave L`	Terminate labelled imperative or object instance L
`restart L`	Goto beginning of labelled imperative or object instance L
`suspend`	Component suspension
`E1 -> E2`	Assignment imperative
`(if E` `// E1 then Imp`$_1$ `// En then Imp`$_n$ `else Imp` `if)`	General selection imperative: Sequential evaluation of E, E1, … En First Impi is executed where Ei=E If no Ei=E, then Imp is executed 'else Imp' is optional
`(if E then` `Imp`$_1$ `else Imp`$_2$ `if)`	Simple if imperative: Evaluation of E (must exit a single boolean value); Execute Imp1 if E is true; Otherwise Imp2 is executed 'else Imp2' is optional
`(for I: range repeat` `Imp` `for)`	Repetition imperative: I is a locally scoped integer variable within Imp. Execute Imp with I assigned each value in [1..range]
`NONE`	The nil reference value
`R[i:j]`	Repetition slice
`R[i]`	Indexed repetition element
`(e1, e2, …, en)`	Evaluation list

Please note, that the above description is by no means complete, and in some cases ambiguous. The ultimate reference is naturally the BETA grammar as defined in the BETA book [BETA93].