A Concise Summary of the Event B mathematical toolkit [1]

Each construct will be given in its presentation form, as displayed in the Rodin toolkit, followed by the `ASCII form` that is used for input to Rodin.

In the following: P, Q and R denote *predicates*;
x and y denote single variables;
z denotes a single or comma-separated list of variables;
p denotes a pattern of variables, possibly including $\mapsto$ and parentheses;
S and T denote set expressions;
U denotes a set of sets;
m and n denote integer expressions;
f and g denote functions;
r denotes a relation;
E and F denote expressions;
E, F is a recursive pattern, *ie* it matches e_1, e_2 and also $e_1, e_2, e_3 \ldots$; similarly for x, y;

Freeness: The meta-predicate $\neg free(z, E)$ means that none of the variables in z occur free in E. This meta-predicate is defined recursively on the structure of E, but that will not be done here explicitly. The base cases are: $\neg free(z, \forall z \cdot P \Rightarrow Q)$, $\neg free(z, \exists z \cdot P \wedge Q)$, $\neg free(z, \{z \cdot P \mid F\})$, $\neg free(z, \lambda z \cdot P|E)$, and $free(z, z)$.

In the following the statement that P *must constrain* z means that the type of z must be at least inferrable from P.

In the following, parentheses are used to show syntactic structure; they may of course be omitted when there is no confusion.

Note: Event-B has a formal syntax and this summary does not attempt to describe that syntax. What it attempts to do is to *explain* Event-B *constructs*. Some words like *expression* collide with the formal syntax. Where a syntactical entity is intended the word will appear in *italics*, *e.g. expression*, *predicate*.

1 Predicates

1. False $\bot$ `false`
2. True $\top$ `true`
3. Conjunction: $P \wedge Q$ `P & Q`
   Left associative.
4. Disjunction: $P \vee Q$ `P or Q`
   Left associative.
5. Implication: $P \Rightarrow Q$ `P => Q`
   Non-associative: this means that $P \Rightarrow Q \Rightarrow R$ must be parenthesised or an error will be diagnosed.
6. Equivalence: $P \Leftrightarrow Q$ `P <=> Q`.
   $P \iff Q = P \Rightarrow Q \wedge Q \Rightarrow P$
   Non-associative: this means that $P \Leftrightarrow Q \Leftrightarrow R$ must be parenthesised or an error will be diagnosed.
7. Negation: $\neg P$ `not P`
8. Universal quantification:
   $\forall z \cdot P \Rightarrow Q$ `!z.P => Q`
   Strictly, $\forall z \cdot P$, but usually an implication.
   For all values of z, satisfying P, Q is satisfied.
   The types of z must be inferrable from the *predicate* P.
9. Existential quantification:
   $\exists z \cdot P \wedge Q$ `#z.P & Q`
   Strictly, $\exists z \cdot P$, but usually a conjunction.
   There exist values of z, satisfying P, that satisfy Q.
   The type of z must be inferrable from the *predicate* P.
10. Equality: $E = F$ `E = F`
11. Inequality: $E \neq F$ `E /= F`

2 Sets

1. Singleton set: $\{E\}$ `{E}`
2. Set enumeration: $\{E, F\}$ `{E, F}`
   See note on the pattern E, F at top of summary.
3. Empty set: $\varnothing$ `{}`
4. Set comprehension: $\{ z \cdot P \mid F \}$ `{ z . P | F }`
   General form: the set of all values of F for all values of z that satisfy the *predicate* P. P must *constrain* the variables in z.
5. Set comprehension: $\{ F \mid P \}$ `{ F | P }`
   Special form: the set of all values of F that satisfy the *predicate* P. In this case the set of bound variables z are all the free variables in F.
   $\{ F \mid P \} = \{ z \cdot P \mid F \}$, where z is all the variables in F.
6. Set comprehension: $\{ x \mid P \}$ `{ x | P }`
   A special case of item 5: the set of all values of x that satisfy the *predicate* P.
   $\{ x \mid P \} = \{ x \cdot P \mid x \}$
7. Union: $S \cup T$ `S \/ T`
8. Intersection: $S \cap T$ `S /\ T`

[1] Version October 7, 2010©1996-2010 Ken Robinson

9. Difference: $S \setminus T$ `S \ T`
$S \backslash T = \{x \mid x \in S \wedge x \notin T\}$

10. Ordered pair: $E \mapsto F$ `E |-> F`.
$E \mapsto F \neq (E, F)$
Left associative.
In all places where an ordered pair is required, $E \mapsto F$ must be used. E, F will not be accepted as an ordered pair, it is always a list. $\{x, y \cdot P \mid x \mapsto y\}$ illustrates the different usage.

11. Cartesian product: $S \times T$ `S ** T`
$S \times T = \{x \mapsto y \mid x \in S \wedge y \in T\}$
Left-associative.

12. Powerset: $\mathbb{P}(S)$ `POW(S)`
$\mathbb{P}(S) = \{s \mid s \subseteq S\}$

13. Non-empty subsets: $\mathbb{P}_1(S)$ `POW1(S)`
$\mathbb{P}_1(S) = \mathbb{P}(S) \backslash \{\varnothing\}$

14. Cardinality: $\mathrm{card}(S)$ `card(S)`
Defined only for $\mathit{finite}(S)$.

15. Generalized union: $\mathrm{union}(U)$ `union(U)`
The union of all the elements of U.
$\forall U \cdot U \in \mathbb{P}(\mathbb{P}(S)) \Rightarrow$
$\mathrm{union}(U) = \{x \mid x \in S \wedge \exists s \cdot s \in U \wedge x \in s\}$
where $\neg \mathit{free}(x, s, U)$

16. Generalized intersection: $\mathrm{inter}(U)$ `inter(U)`
The intersection of all the elements of U.
$U \neq \varnothing$,
$\forall U \cdot U \in \mathbb{P}(\mathbb{P}(S)) \Rightarrow$
$\mathrm{inter}(U) = \{x \mid x \in S \wedge \forall s \cdot s \in U \Rightarrow x \in s\}$
where $\neg \mathit{free}(x, s, U)$

17. Quantified union:
$\cup z \cdot P \mid S$ `UNION z.P | S`
P must *constrain* the variables in z.
$\forall z \cdot P \Rightarrow S \subseteq T \Rightarrow$
$\cup(z \cdot P \mid E) = \{x \mid x \in T \wedge \exists z \cdot P \wedge x \in S\}$
where $\neg \mathit{free}(x, z, T)$, $\neg \mathit{free}(x, P)$, $\neg \mathit{free}(x, S)$, $\neg \mathit{free}(x, z)$

18. Quantified intersection:
$\cap z \cdot P \mid S$ `INTER z.P | S`
P must *constrain* the variables in z,
$\{z \mid P\} \neq \varnothing$,
$(\forall z \cdot (P \Rightarrow S \subseteq T)) \Rightarrow$
$\cap z \cdot P \mid S = \{x \mid x \in T \wedge (\forall z \cdot P \Rightarrow x \in S)\}$
where $\neg \mathit{free}(x, z)$, $\neg \mathit{free}(x, T)$, $\neg \mathit{free}(x, P)$, $\neg \mathit{free}(x, S)$.

2.1 Set predicates

1. Set membership: $E \in S$ `E : S`
2. Set non-membership: $E \notin S$ `E /: S`
3. Subset: $S \subseteq T$ `S <: T`
4. Not a subset: $S \not\subseteq T$ `S /<: T`
5. Proper subset: $S \subset T$ `S <<: T`
6. Not a proper subset: $s \not\subset t$ `S /<<: T`
7. Finite set: $\mathit{finite}(S)$ `finite(S)`
$\mathit{finite}(S) \Leftrightarrow$ *S is finite.*
8. Partition: $\mathit{partition}(S, x, y)$ `partition(S,x,y)`
x and y partition the set S, *ie* $S = x \cup y \wedge x \cap y = \varnothing$
Specialised use for enumerated sets: $\mathit{partition}(S, \{A\}, \{B\}, \{C\})$.
$S = \{A, B, C\} \wedge A \neq B \wedge B \neq C \wedge C \neq A$

3 BOOL and bool

BOOL is the enumerated set: $\{\mathrm{FALSE}, \mathrm{TRUE}\}$ and bool is defined on a predicate P as follows:

1. P is provable: $\mathrm{bool}(P) = \mathrm{TRUE}$
2. $\neg P$ is provable: $\mathrm{bool}(P) = \mathrm{FALSE}$

4 Numbers

The following is based on the set of integers, the set of natural numbers (non-negative integers), and the set of positive (non-zero) natural numbers.

1. The set of integer numbers: $\mathbb{Z}$ `INT`
2. The set of natural numbers: $\mathbb{N}$ `NAT`
3. The set of positive natural numbers: $\mathbb{N}_1$ `NAT1`
$\mathbb{N}_1 = \mathbb{N} \backslash \{0\}$
4. Minimum: $\min(S)$ `min(S)`
$S \subset \mathbb{Z}$ and $\mathit{finite}(S)$ or S must have a lower bound.
5. Maximum: $\max(S)$ `max(S)`
$S \subset \mathbb{Z}$ and $\mathit{finite}(S)$ or S must have an upper bound.
6. Sum: $m + n$ `m + n`
7. Difference: $m - n$ `m - n`
$n \leq m$
8. Product: $m \times n$ `m * n`
9. Quotient: m/n `m / n`
$n \neq 0$
10. Remainder: $m \mathbin{mod} n$ `m mod n`
$n \neq 0$
11. Interval: $m \,..\, n$ `m .. n`
$m \,..\, n = \{\ i \mid m \leq i \wedge i \leq n\ \}$

4.1 Number predicates

1. Greater: $m > n$ `m > n`
2. Less: $m < n$ `m < n`
3. Greater or equal: $m \geq n$ `m >= n`
4. Less or equal: $m \leq n$ `m <= n`

5 Relations

A relation is a set of ordered pairs; a many to many mapping.

1. Relations: $S \leftrightarrow T$ `S <-> T`
   $S \leftrightarrow T = \mathbb{P}(S \times T)$
   Associativity: relations are *right associative*:
   $r \in X \leftrightarrow Y \leftrightarrow Z = r \in X \leftrightarrow (Y \leftrightarrow Z)$.

2. Domain: $\mathrm{dom}(r)$ `dom(r)`
   $\forall r \cdot r \in S \leftrightarrow T \Rightarrow$
   $\mathrm{dom}(r) = \{x \cdot (\exists y \cdot x \mapsto y \in r)\}$

3. Range: $\mathrm{ran}(r)$ `ran(r)`
   $\forall r \cdot r \in S \leftrightarrow T \Rightarrow$
   $\mathrm{ran}(r) = \{y \cdot (\exists x \cdot x \mapsto y \in r)\}$

4. Total relation: $S \twoheadleftarrow\!\!\rightarrow T$ `S <<-> T`
   if $r \in S \twoheadleftarrow\!\!\rightarrow T$ then $\mathrm{dom}(r) = S$

5. Surjective relation: $S \leftarrow\!\!\twoheadrightarrow T$ `S <->> T`
   if $r \in S \leftarrow\!\!\twoheadrightarrow T$ then $\mathrm{ran}(r) = T$

6. Total surjective relation: $S \twoheadleftarrow\!\!\twoheadrightarrow T$ `S <<->> T`
   if $r \in S \twoheadleftarrow\!\!\twoheadrightarrow T$ then $\mathrm{dom}(r) = S$ and $\mathrm{ran}(r) = T$

7. Forward composition: $p \,;\, q$ `p ; q`
   $\forall p, q \cdot p \in S \leftrightarrow T \wedge q \in T \leftrightarrow U \Rightarrow$
   $p \,;\, q = \{x \mapsto y \mid (\exists z \cdot x \mapsto z \in p \wedge z \mapsto y \in q)\}$

8. Backward composition: $p \circ q$ `p circ q`
   $p \circ q = q \,;\, p$

9. Identity: id `id`
   $S \lhd \mathrm{id} = \{x \mapsto x \mid x \in S\}$.
   id is generic and the set S is inferred from the context.

10. Domain restriction: $S \lhd r$ `S <| r`
    $S \lhd r = \{x \mapsto y \mid x \mapsto y \in r \wedge x \in S\}$.

11. Domain subtraction: $S \ntriangleleft r$ `S <<| r`
    $S \ntriangleleft r = \{x \mapsto y \mid x \mapsto y \in r \wedge x \notin S\}$.

12. Range restriction: $r \rhd T$ `r |> T`
    $r \rhd T = \{x \mapsto y \mid x \mapsto y \in r \wedge y \in T\}$.

13. Range subtraction: $r \ntriangleright T$ `r |>> T`
    $r \ntriangleright T = \{x \mapsto y \mid y \in r \wedge y \notin T\}$.

14. Inverse: r^{-1} `r~`
    $r^{-1} = \{y \mapsto x \mid x \mapsto y \in r\}$.

15. Relational image: $r[S]$ `r[S]`
    $r[S] = \{y \mid \exists x \cdot x \in S \wedge x \mapsto y \in r\}$.

16. Overriding: $r_1 \Lleftarrow r_2$ `r1 <+ r2`
    $r_1 \Lleftarrow r_2 = r_2 \cup (\mathrm{dom}(r_2) \ntriangleleft r_1)$.

17. Direct product: $p \otimes q$ `p >< q`
    $p \otimes q = \{x \mapsto (y \mapsto z) \mid x \mapsto y \in p \wedge x \mapsto z \in q)\}$.

18. Parallel product: $p \parallel q$ `p || q`
    $p \parallel q = \{x, y, m, n \cdot x \mapsto m \in p \wedge y \mapsto n \in q \mid (x \mapsto y) \mapsto (m \mapsto n)\}$.

19. Projection: prj_1 `prj1`
    prj_1 is generic.
    $(S \times T) \lhd \mathrm{prj}_1 = \{(x \mapsto y) \mapsto x \mid x \mapsto y \in S \times T\}$.

20. Projection: prj_2 `prj2`
    prj_2 is generic.
    $(S \times T) \lhd \mathrm{prj}_2 = \{(x \mapsto y) \mapsto y \mid x \mapsto y \in S \times T\}$.

5.1 Iteration and Closure

Iteration and closure are important functions on relations that are not currently part of the kernel Event-B language. They can be defined in a Context, but not polymorphically.

Note: iteration and irreflexive closure will be implemented in a proposed extension of the mathematical language. The operators will be non-associative.

1. Iteration: r^n ☐
   $r \in S \leftrightarrow S \Rightarrow r^0 = S \lhd \mathrm{id} \wedge r^{n+1} = r \,;\, r^n$.
   Note: to avoid inconsistency S should be the finite *base* set for r, ie the smallest set for which all $r \in S \leftrightarrow S$.
   Could be defined as a function $\mathit{iterate}(r \mapsto n)$.

2. Reflexive Closure: r^* ☐
   $r^* = \cup n \cdot (n \in \mathbb{N} \mid r^n)$.
   Could be defined as a function $\mathit{rclosure}(r)$.
   Note: $r^0 \subseteq r^*$.

3. Irreflexive Closure: r^+ ☐
   $r^+ = \cup n \cdot (n \in \mathbb{N}_1 \mid r^n)$.
   Could be defined as a function $\mathit{iclosure}(r)$.
   Note: $r^0 \not\subseteq r^+$ by default, but may be present depending on r.

5.2 Functions

A function is a relation with the restriction that each element of the domain is related to a unique element in the range; a many to one mapping.

1. Partial functions: $S \pfun T$ `S +-> T`
   $S \pfun T = \{r \cdot r \in S \leftrightarrow T \wedge r^{-1} \,;\, r \subseteq T \lhd \mathrm{id}\}$.

2. Total functions: $S \rightarrow T$ `S --> T`
   $S \rightarrow T = \{f \cdot f \in S \pfun T \wedge \mathrm{dom}(f) = S\}$.

3. Partial injections: $S \pinj T$ `S >+> T`
   $S \pinj T = \{f \cdot f \in S \pfun T \wedge f^{-1} \in T \pfun S\}$.
   One-to-one relations.

4. Total injections: $S \rightarrowtail T$ `S >-> T`
   $S \rightarrowtail T = S \pinj T \cap S \rightarrow T$.

5. Partial surjections: $S \psurj T$ `S +->> T`
   $S \psurj T = \{f \cdot f \in S \pfun T \wedge \mathrm{ran}(f) = T\}$.
   Onto relations.

6. Total surjections: $S \twoheadrightarrow T$ `S -->> T`
   $S \twoheadrightarrow T = S \psurj T \cap S \rightarrow T$.

7. Bijections: $S \rightarrowtail\!\!\!\!\twoheadrightarrow T$ `S >->> T`
   $S \rightarrowtail\!\!\!\!\twoheadrightarrow T = S \rightarrowtail T \cap S \twoheadrightarrow T$.
   One-to-one and onto relations.

8. Lambda abstraction: $(\lambda p \cdot P \mid E)$ `(%p.P|E)`
   P must *constrain* the variables in p.
   $(\lambda p \cdot P \mid E) = \{z \cdot P \mid p \mapsto E\}$, where z is a list of variables that appear in the pattern p.

9. Function application: $f(E)$ `f(E)`
   $E \mapsto y \in f \Rightarrow E \in \mathrm{dom}(f) \wedge f \in X \rightarrow\!\!\!\!\!+ Y$, where $type(f) = \mathbb{P}(X \times Y)$.
   Note: in Event-B, relations and functions only ever have one argument, but that argument may be a pair or tuple, hence $f(E \mapsto F)$ `f(E |-> F)`
   $f(E, F)$ is never valid.

6 Models

1. Contexts: contain sets and constants used by other contexts or machines.

CONTEXT	Identifier
EXTENDS	Machine_Identifiers
SETS	Identifiers
CONSTANTS	Identifiers
AXIOMS	Predicates
END	

   Note: *theorems* can be presented in the AXIOMS part of a context.

2. Machines: contain events.

MACHINE	Identifier
REFINES	Machine_Identifiers
SEES	Context_Identifiers
VARIABLES	Identifiers
INVARIANT	Predicates
VARIANT	Expression
EVENTS	Events
END	

   Note: *theorems* can be presented in the INVARIANT section of a machine and the WHERE part of an event.

6.1 Events

Event_name	
REFINES	Event_identifiers
ANY	Identifiers
WHERE	Predicates
WITH	Witnesses
THEN	Actions
END	

There is one distinguished event named *INITIALISATION* used to initialise the variables of a machine, thus establishing the invariant.

6.2 Actions

Actions are used to change the state of a machine. There may be multiple actions, but they take effect concurrently, that is, in parallel. The semantics of events are defined in terms of *substitutions*. The substitution $[G]P$ defines a predicate obtained by replacing the values of the variables in P according to the action G. General substitutions are not available in the Event-B language.

Note on concurrency: any single variable can be modified in at most one action, otherwise the effect of the actions would, in general, be inconsistent.

1. *skip*, the null action:
   skip denotes the empty set of actions for an event.

2. Simple assignment action: $z := E$ `x := E`
   $:=$ = *"becomes equal to"*: replace free occurrences of x by E.

3. Choice from set: $x :\in S$ `x :: S`
   $:\in$ = *"becomes in"*: arbitrarily choose a value from the set S.

4. Choice by predicate: $z :\mid P$ `z :| P`
   $:\mid$ = *"becomes such that"*: arbitrarily choose values for the variable in z that satisfy the predicate P. Within P, x refers to the value of the variable x before the action and x' refers to the value of the variable after the action.

5. Functional override: $f(x) := E$ `f(x) := E`
   Substitute the value E for the function/relation f at the point x.
   This is a shorthand:
   $f(x) := E \;=\; f := f \Leftarrow\!\!\!+ \{x \mapsto E\}$.

Acknowledgement: Jean-Raymond Abrial, Laurent Voisin and Ian Hayes have all given valuable feedback and corrections at various stages of the evolution of this summary.