

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 list of comma-separated 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 lists of 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 $e_1, e_2, e_3 \dots$; similarly for x, y ;

Freeness: The meta-predicate $notfree(z, E)$ means that none of the variables in z occur *free* in E . This meta-predicate is defined recursively on the structure of E , but we won't do that here. The base cases are: $notfree(z, \forall z \cdot P)$, $notfree(z, \exists z \cdot P)$, $notfree(z, \{z|P\})$, $notfree(z, \lambda z \cdot (P|E))$, and $\neg notfree(z, z)$.

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

1 Predicates

A predicate is a function from some set X to Boolean (bool)

1. False: \perp false
2. True: \top true

Boolean cannot be used as a type for constants and variables. Instead EventB provides a set BOOL defined as an enumeration

$$\text{BOOL} = \{\text{FALSE}, \text{TRUE}\},$$

which can be used for *concrete* representations of *false* and *true*.

There is also a function bool that maps Boolean expressions into values in BOOL: $\text{bool}(\perp) = \text{FALSE}$ and $\text{bool}(\top) = \text{TRUE}$.

1. Conjunction: $P \wedge Q$ P & Q
Left associative.
2. Disjunction: $P \vee Q$ P or Q
Left associative.
3. 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.
4. Equivalence: $P \Leftrightarrow Q$ P <=> Q
 $P \Leftrightarrow Q = P \Rightarrow Q \wedge Q \Rightarrow P$
Non-associative: this means that $P \Leftrightarrow Q \Leftrightarrow R$ must be parenthesised or an error will be diagnosed.
5. Negation: $\neg P$ not P
6. Universal quantification:
 $(\forall z \cdot P) \Rightarrow Q$ (!z.P) => Q
For all values of z satisfying P , Q (is true)
 P must *constrain* the variables in z .

7. Existential quantification:

$$(\exists z \cdot P) \wedge Q \quad \text{span style="border: 1px solid black; padding: 2px;">(#z.P) \& Q}$$

There exists some values of z satisfying P for which Q . P must *constrain* the variables in z .

8. Substitution: $[G] P$ [G] P
9. Equality: $E = F$ E = F
10. 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: \emptyset { }
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 1: 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: $\{x \cdot P \mid F\}$ { x | P }
Special form 2: 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 \vee T
8. Intersection: $S \cap T$ S \wedge T

¹Version March 30, 2009©1996-2009 Ken Robinson

9. Difference: $S \setminus T$ **S \ T**
 $S - T = \{x \cdot x \in S \wedge x \notin T\}$

10. Ordered pair: $E \mapsto F$ **E |-> F**
 $E \mapsto F = E, F$
 Note: in all places $E \mapsto F$ must be used. E, F will not be accepted as a mapping, it is always a list; for example: $\{x, y \cdot P \mid x \mapsto y\}$.

11. Cartesian product: $S \times T$ **S ** T**
 $S \times T = \{x, y \cdot x \in S \wedge y \in T \mid x \mapsto y\}$

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

13. Non-empty subsets: $\mathbb{P}_1(S)$ **POW1(S)**
 $\mathbb{P}_1(S) = \mathbb{P}(S) - \{\emptyset\}$

14. Cardinality: $\text{card}(S)$ **card(S)**
 Defined only for finite sets

15. Generalized union: $\text{union}(U)$ **union(U)**
 The union of all the elements of U .
 $\forall U \cdot U \in \mathbb{P}(\mathbb{P}(S)) \Rightarrow$
 $\text{union}(U) = \{x \cdot x \in S \wedge (\exists s \cdot s \in U \wedge x \in s)\}$
 where x, s not free U

16. Generalized intersection: $\text{inter}(U)$ **inter(U)**
 The intersection of all the elements of U .
 $U \neq \emptyset,$
 $\forall U \cdot U \in \mathbb{P}(\mathbb{P}(S)) \Rightarrow$
 $\text{inter}(U) = \{x \cdot x \in S \wedge (\forall s \cdot s \in U \Rightarrow x \in s)\}$
 where x, s not free U

17. Generalized union: **UNION z.P | E**
 $\cup z \cdot P \mid E$
 P must *constrain* the variables in z .
 $(\forall z \cdot (P \Rightarrow E \subseteq T)) \Rightarrow$
 $\cup(z \cdot P \mid E) = \{x \cdot x \in T \wedge (\exists z \cdot (P \wedge x \in E))\}$
 where z not free T, x not free T, P, E, z

18. Generalized intersection: **INTER z.P | E**
 $\cap z \cdot P \mid E$
 P must *constrain* the variables in z ,
 $\{z \cdot P\} \neq \emptyset,$
 $(\forall z \cdot (P \Rightarrow E \subseteq T)) \Rightarrow$
 $\cap z \cdot P \mid E = \{x \cdot x \in T \wedge (\forall z \cdot P \Rightarrow x \in E)\}$
 where x not free z, T, P, E

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: $\text{finite}(S)$ **finite((S))**
 Declares S to be finite.

3 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**
 $\mathbb{N}_1 = \mathbb{N} - \{0\}$

4. Minimum: $\text{min}(S)$ **min(S)**
 $S \subseteq \mathbb{Z}$ and $\text{finite}(S)$

5. Maximum: $\text{max}(S)$ **max(S)**
 $S \subseteq \mathbb{Z}$ and $\text{finite}(S)$

6. Sum: $m + n$ **m + n**

7. Difference: $m - n$ **m - n**

8. Product: $m \times n$ **m * n**

9. Quotient: m/n **m / n**

10. Remainder: $m \bmod n$ **m mod n**

11. Interval: $m .. n$ **m .. n**
 $m .. n = \{i \cdot m \leq i \leq n\}$.

3.1 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**

4 Relations

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

1. Relations: $S \leftrightarrow T$ **S <-> T**
 $S \leftrightarrow T = \mathbb{P}(S \times T)$

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

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

4. Total relation: $S \leftrightarrow T$ **S <<-> T**
 if $r \in S \leftrightarrow T$ then $\text{dom}(r) = S$

5. Surjective relation: $S \leftrightarrow T$ **S <->> T**
 if $r \in S \leftrightarrow T$ then $\text{ran}(r) = T$

6. Total surjective relation: $S \leftrightarrow T$ **S <<->> T**
 if $r \in S \leftrightarrow T$ then $\text{dom}(r) = S$ and $\text{ran}(r) = T$

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

8. Backward composition: $p \circ q$ $p \text{ circ } q$
 $p \circ q = q ; p$
9. Identity: $\text{id}(S)$ $\text{id}(S)$
 $\text{id}(S) = \{x, y \cdot x \in S \wedge y \in S \wedge x = y \mid x \mapsto y\}$.
Note: in Rodin 1.0 id will be polymorphic, and the set will be inferred from the context.
10. Domain restriction: $S \triangleleft r$ $S \triangleleft r$
 $S \triangleleft r = \{x, y \cdot x \mapsto y \in r \wedge x \in S \mid x \mapsto y\}$.
11. Domain subtraction: $S \triangleleft\!\!\triangleleft r$ $S \triangleleft\!\!\triangleleft r$
 $S \triangleleft\!\!\triangleleft r = \{x, y \cdot x \mapsto y \in r \wedge x \notin S \mid x \mapsto y\}$.
12. Range restriction: $r \triangleright T$ $r \triangleright T$
 $r \triangleright T = \{x, y \cdot x \mapsto y \in r \wedge y \in T \mid x \mapsto y\}$.
13. Range subtraction: $r \triangleright\!\!\triangleright T$ $r \triangleright\!\!\triangleright T$
 $r \triangleright\!\!\triangleright T = \{x, y \cdot x \mapsto y \in r \wedge y \notin T \mid x \mapsto y\}$.
14. Inverse: r^{-1} r^{-1}
 $r^{-1} = \{x, y \cdot x \mapsto y \in r \mid y \mapsto x\}$.
15. Relational image: $r[S]$ $r[S]$
 $r[S] = \{y \cdot \exists x \cdot x \in S \wedge x \mapsto y \in r\}$.
16. Overriding: $r_1 \triangleleft r_2$ $r_1 \triangleleft r_2$
 $r_1 \triangleleft r_2 = r_2 \cup (\text{dom}(r_2) \triangleleft r_1)$.
17. Direct product: $p \otimes q$ $p \otimes q$
 $p \otimes q = \{x, y, z \cdot x \mapsto y \in p \wedge x \mapsto z \in q \mid x \mapsto (y \mapsto z)\}$.
18. Parallel product: $p \parallel q$ $p \parallel q$
 $p \parallel q = \{x, y, m, n \cdot x \mapsto m \in p \wedge y \mapsto n \in q \mid (x \mapsto y) \mapsto (m \mapsto n)\}$.
19. Projection: $\text{prj}_1(S, T)$ $\text{prj}_1(S, T)$
 $\text{prj}_1(S, T) = \{x, y, z \cdot x \mapsto y \in S \times T \wedge z = x \mid (x \mapsto y), z\}$.
20. Projection: $\text{prj}_2(S, T)$ $\text{prj}_2(S, T)$
 $\text{prj}_2(S, T) = \{x, y, z \cdot x \mapsto y \in S \times T \wedge z = y \mid (x \mapsto y) \mapsto z\}$.
Projections are expected to become polymorphic.

4.1 Iteration and Closure

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

1. Iteration: r^n □
 $r \in S \leftrightarrow S \Rightarrow r^0 = \text{id}(S) \wedge r^{n+1} = r ; r^n$.
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 $\text{iterate}(r \mapsto n)$.
2. Reflexive Closure: r^* □
 $r^* = \cup n \cdot (n \in \mathbb{N} \mid r^n)$.
 Could be defined as a function $\text{rclosure}(r)$
3. Irreflexive Closure: r^+ □
 $r^+ = \cup n \cdot (n \in \mathbb{N}_1 \mid r^n)$.
 Could be defined as a function $\text{iclosure}(r)$.

4.2 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: $S \leftrightarrow T$ $S \leftrightarrow T$
 $S \leftrightarrow T = \{r \cdot r \in S \leftrightarrow T \wedge r^{-1} ; r \subseteq \text{id}(T)\}$.
2. Total functions: $S \rightarrow T$ $S \rightarrow T$
 $S \rightarrow T = \{f \cdot f \in S \leftrightarrow T \wedge \text{dom}(f) = S\}$.
3. Partial injections: $S \mapsto T$ $S \mapsto T$
 $S \mapsto T = \{f \cdot f \in S \leftrightarrow T \wedge f^{-1} \in T \leftrightarrow S\}$.
One-to-one relations.
4. Total injections: $S \mapsto T$ $S \mapsto T$
 $S \mapsto T = S \mapsto T \cap S \rightarrow T$.
5. Partial surjections: $S \twoheadrightarrow T$ $S \twoheadrightarrow T$
 $S \twoheadrightarrow T = \{f \cdot f \in S \leftrightarrow T \wedge \text{ran}(f) = T\}$.
Onto relations.
6. Total surjections: $S \twoheadrightarrow T$ $S \twoheadrightarrow T$
 $S \twoheadrightarrow T = S \twoheadrightarrow T \cap S \rightarrow T$.
7. Bijections: $S \xrightarrow{\sim} T$ $S \xrightarrow{\sim} T$
 $S \xrightarrow{\sim} T = S \mapsto T \cap S \twoheadrightarrow T$.
One-to-one and onto relations.
8. Lambda abstraction: $\lambda p \cdot P \mid E$
 $\lambda p \cdot P \mid E$
 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: $f(E)$ $f(E)$
 $E \mapsto y \in f \Rightarrow f(E) = y$.
Note: in EventB, relations and functions will only ever have one argument, but that argument may be a pair, hence $f(E \mapsto F)$ $f(E \mapsto F)$
 $f(E, F)$ is never valid.

5 Machines

1. Context machines: contain sets and constants used by other machines.

CONTEXT	Identifier
EXTENDS	Machine_Identifiers
SETS	Identifiers
CONSTANTS	Identifiers
AXIOMS	Predicates
THEOREMS	Predicates
END	
2. Event machines: contain events.

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

5.1 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.

5.2 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 EventB language.

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

1. *skip*, the null action:
 $skip$ denotes the empty set of actions for an event.
2. Simple assignment action: $x := E$ $\mathbf{x := E}$
 $:=$ = “becomes equal to”: replace free occurrences of x by E .
3. Choice from set: $x \in S$ $\mathbf{x :: S}$
 \in = “becomes in”: arbitrarily choose a value from the set S .
4. Choice by predicate: $x | P$ $\mathbf{x :| P}$
 $|$ = “becomes such that”: arbitrarily choose a value that satisfies the predicate P . P must *constrain* the variable x . Within P , x refers to the value of the variable before the action and x' refers to the value of the variable after the action.
5. Functional override: $f(x) := E$ $\mathbf{f(x) := E}$
Substitute the value E for the expression f at point x .
This is a shorthand for $f(x) := E = f := f \Leftarrow \{x \mapsto E\}$.
6. Multiple action:
 $x, y := E, F$ $\mathbf{x, y := E, F}$
Concurrent assignment of the values E and F to the variables x and y , respectively. This is equivalent multiple single actions.

