

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 variables;

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; H and I denote a generalized substitutions; 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 $\neg notfree(z, z)$.

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

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

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**: $\mathbf{bool}(\perp) = \mathbf{FALSE}$ and $\mathbf{bool}(\top) = \mathbf{TRUE}$.

1. Conjunction: $P \wedge Q$ P & Q
2. Disjunction: $P \vee Q$ P or Q
3. Implication: $P \Rightarrow Q$ P => Q
4. Equivalence: $P \Leftrightarrow Q$ P <=> Q.
 $P \Leftrightarrow Q = P \Rightarrow Q \wedge Q \Rightarrow P$
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:
 $\exists z \cdot P \wedge Q$ #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: $\{z | P\}$ { z | P }
The set of all values of z that satisfy the predicate P . P must *constrain* the variables in z .
5. Union: $S \cup T$ S \vee T
6. Intersection: $S \cap T$ S /\ T
7. Difference: $S \setminus T$ S \ T
 $S - T = \{x | x \in S \wedge x \notin T\}$
8. Ordered pair: $E \mapsto F$ E |-> F.
 $E \mapsto F = E, F$
Note: in most places $E \mapsto F$ must be used, and E, F will not be accepted, but there are a few contexts, for example set comprehension: $\{x, y | P\}$, where $E \mapsto F$ is not accepted!
9. Cartesian product: $S \times T$ S ** T
 $S \times T = \{x, y | x \in S \wedge y \in T\}$
10. Powerset: $\mathbb{P}(S)$ POW(S)
 $\mathbb{P}(S) = \{s | s \subseteq S\}$
11. Non-empty subsets: $\mathbb{P}_1(S)$ POW1(S)
 $\mathbb{P}_1(S) = \mathbb{P}(S) - \{\emptyset\}$
12. Finite set: *finite* S finite S
Declares S to be finite.
13. Cardinality: $\mathbf{card}(S)$ card(S)
Defined only for finite sets
14. Generalized union: $\mathbf{union}(U)$ union(U)
The union of all the elements of U .
 $\forall U \cdot U \in \mathbb{P}(\mathbb{P}(S)) \Rightarrow$
 $\mathbf{union}(U) = \{x | x \in S \wedge (\exists s \cdot s \in U \wedge x \in s)\}$
where x, s *not free* U

¹Version March 26, 2008©1996-2008 Ken Robinson

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

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

17. Generalized intersection:
 $\bigcap z \cdot (P \mid E)$ $\text{INTER } z.P \mid E$
 P must *constrain* the variables in z ,
 $\{z \mid P\} \neq \emptyset$,
 $(\forall z \cdot (P \Rightarrow E \subseteq T)) \Rightarrow$
 $\bigcap z \cdot (P \mid E) = \{x \mid x \in T \wedge (\forall z \cdot (P \Rightarrow x \in E))\}$
 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: $s \not\subset t$ $S /<<: T$

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: $\text{min}(S)$ $\text{min}(S)$
 $S \subset \mathbb{Z}$ and $\text{finite}(S)$
5. Maximum: $\text{max}(S)$ $\text{max}(S)$
 $S \subset \mathbb{Z}$ 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: $m \bmod n$ $m \bmod n$
11. Interval: $m .. n$ $m .. n$
 $m .. n = \{i \mid m \leq i \leq n\}$.

3.1 Number predicates

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

4 Relations

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

1. Relations: $S \leftrightarrow T$ $S \leftrightarrow T$
 $S \leftrightarrow T = \mathbb{P}(S \times T)$
2. Domain: $\text{dom}(r)$ $\text{dom}(r)$
 $\forall r \cdot r \in S \leftrightarrow T \Rightarrow$
 $\text{dom}(r) = \{x \mid (\exists y \cdot x \mapsto y \in r)\}$
3. Range: $\text{ran}(r)$ $\text{ran}(r)$
 $\forall r \cdot r \in S \leftrightarrow T \Rightarrow$
 $\text{ran}(r) = \{y \mid (\exists x \cdot x \mapsto y \in r)\}$
4. Forward composition: $p ; q$ $p ; q$
 $\forall p, q \cdot p \in S \leftrightarrow T \wedge q \in T \leftrightarrow U \Rightarrow$
 $p ; q = \{x, y \mid (\exists z \cdot x \mapsto z \in p \wedge z \mapsto y \in q)\}$
5. Backward composition: $p \circ q$ $p \circ q$
 $p \circ q = q ; p$
6. Identity: $\text{id}(S)$ $\text{id}(S)$
 $\text{id}(S) = \{x, y \mid x \in S \wedge y \in S \wedge x = y\}$.
7. Domain restriction: $S \triangleleft r$ $S \triangleleft r$
 $S \triangleleft r = \{x, y \mid x \mapsto y \in r \wedge x \in S\}$.
8. Domain substraction: $S \triangleleft\!\!\triangleleft r$ $S \triangleleft\!\!\triangleleft r$
 $S \triangleleft\!\!\triangleleft r = \{x, y \mid x \mapsto y \in r \wedge x \notin S\}$.
9. Range restriction: $r \triangleright T$ $r \triangleright T$
 $r \triangleright T = \{x, y \mid x \mapsto y \in r \wedge y \in T\}$.
10. Range substraction: $r \triangleright\!\!\triangleright T$ $r \triangleright\!\!\triangleright T$
 $r \triangleright\!\!\triangleright T = \{x, y \mid x \mapsto y \in r \wedge y \notin T\}$.
11. Inverse: r^{-1} r^{-1}
 $r^{-1} = \{y, x \mid x \mapsto y \in r\}$.
12. Relational image: $r[S]$ $r[S]$
 $r[S] = \{y \mid \exists x \cdot x \in S \wedge x \mapsto y \in r\}$.
13. Overriding: $r_1 \triangleleft r_2$ $r_1 \triangleleft r_2$
 $r_1 \triangleleft r_2 = r_2 \cup (\text{dom}(r_2) \triangleleft r_1)$.
14. Direct product: $p \otimes q$ $p \otimes q$
 $p \otimes q = \{x, (y, z) \mid x \mapsto y \in p \wedge x \mapsto z \in q\}$.
15. Parallel product: $p \parallel q$ $p \parallel q$
 $p \parallel q = \{(x, y), (m, n) \mid x \mapsto m \in p \wedge y \mapsto n \in q\}$.
16. Iteration: r^n $\text{iterate}(r, n)$
 $r \in S \leftrightarrow S \Rightarrow r^0 = \text{id}(S) \wedge r^{n+1} = r ; r^n$.
 Note: to avoid inconsistency S should be the finite *base* set for r , ie the smallest set for which all $r \in S \leftrightarrow S$.

17. Closure: r^* `closure(x)`
 $r^* = \bigcup n \cdot (n \in \mathbb{N} \mid r^n).$
18. Projection: $\text{prj}_1(S, T)$ `prj1(S, T)`
 $\text{prj}_1(S, T) =$
 $\{(x, y), z \mid x, y \in S \times T \wedge z = x\}.$
19. Projection: $\text{prj}_2(S, T)$ `prj2(S, T)`
 $\text{prj}_2(S, T) =$
 $\{(x, y), z \mid x, y \in S \times T \wedge z = y\}.$

4.1 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 <-> T`
 $S \leftrightarrow T = \{r \mid r \in S \leftrightarrow T \wedge r^{-1}; r \subseteq \text{id}(T)\}.$
2. Total functions: $S \rightarrow T$ `S --> T`
 $S \rightarrow T = \{f \mid f \in S \rightarrow T \wedge \text{dom}(f) = S\}.$
3. Partial injections: $S \mapsto T$ `S >> T`
 $S \mapsto T = \{f \mid f \in S \rightarrow T \wedge f^{-1} \in T \rightarrow S\}.$
One-to-one relations.
4. Total injections: $S \mapsto T$ `S >-> T`
 $S \mapsto T = S \mapsto T \cap S \rightarrow T.$
5. Partial surjections: $S \twoheadrightarrow T$ `S +->> T`
 $S \twoheadrightarrow T = \{f \mid f \in S \rightarrow T \wedge \text{ran}(f) = T\}.$
Onto relations.
6. Total surjections: $S \twoheadrightarrow T$ `S -->> T`
 $S \twoheadrightarrow T = S \twoheadrightarrow T \cap S \rightarrow T.$
7. Bijections: $S \simeq T$ `S >->> T`
 $S \simeq T = S \mapsto T \cap S \twoheadrightarrow T.$
One-to-one and onto relations.
8. Lambda abstraction:
 $\lambda z \cdot (P \mid E)$ `%z. (P|E)`
 P must *constrain* the variables in z .
 $\lambda z \cdot (P \mid E) = \{z, y \mid z \in \{z \mid P\} \wedge y = E\}$, where
not free P and *not free* E .
9. Function application: $f(E)$ `f(E)`
 $E \mapsto y \in f \Rightarrow f(E) = y.$

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 name *INITIALISATION* used to initialise the variables of a machine.

5.2 Substitutions

The actions referred to above are also known as *substitutions*, because of their semantics and the fact that the state of a machine is changed by substituting (storing) values into the variables of a machine. The following substitutions formalize a number of alternative ways of achieving this.

Only substitutions 1 to 7 below are used in an Event B event; 8 is used implicitly since all actions for a particular event are executed concurrently.

1. Substitution: $[G]P$ `[G]P`
 $[G]P$ is a predicate obtained by replacing the values of the variables in P according to the substitution G .
2. The null substitution: *skip* `skip`
 $[skip]R = R.$
3. Simple substitution: $x := E$ `x := E`
 Replace free occurrences of x by E .
4. Choice from set: $x \in S$ `x :: S`
 Arbitrarily choose a value from the set S .
5. Choice by predicate: $x \mid P$ `x :| P`
 Arbitrarily choose a value that satisfies the predicate P . P must *constrain* the variable x .
6. Functional override: $f(x) := E$ `f(x) := E`
 Substitute the value E for the expression f at point x .
 $f(x) := E = f := f \triangleleft \{x \mapsto E\}.$
7. Multiple substitution:
 $x, y := E, F$ `x, y := E, F`
 Concurrent substitution of the values E and F for the free occurrences of x and y , respectively.
8. Parallel substitution: $G \parallel H$ `G || H`
 Apply the substitutions G and H concurrently.
 Parallel substitution is not given a general substitution semantics; it is eliminated by rewriting rules. Notice
 $x := E \parallel y := F = x, y := E, F.$

