

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 $e_1, e_2, e_3 \dots$; similarly for x, y ;

Freeness: The meta-predicate $\neg \text{free}(z, E)$ 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: $\neg \text{free}(z, \forall z \cdot P \Rightarrow Q)$, $\neg \text{free}(z, \exists z \cdot P \wedge Q)$, $\neg \text{free}(z, \{z \cdot P \mid F\})$, $\neg \text{free}(z, \lambda z \cdot P[E])$, and $\text{free}(z, z)$.

In the following the statement that P *must constrain* z means that the type of z must be at least inferable 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 \perp 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.
 $P \Leftrightarrow Q = P \Rightarrow Q \wedge Q \Rightarrow P$
Non-associative: this means that $P \Leftrightarrow Q \Leftrightarrow R$ must be parenthesised or an error will be diagnosed.
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 inferable 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 inferable 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: \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: 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 \vee T
8. Intersection: $S \cap T$ S /\ T

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

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

7. Finite set: $finite(S)$ $finite(S)$
 $finite(S) \Leftrightarrow S \text{ is finite.}$

10. Ordered pair: $E \mapsto F$ $E \mapsto F$
 $E \mapsto F \neq (E, F)$
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.

8. Partition: $partition(S, x, y)$ $partition(S, x, y)$
 x and y partition the set S , ie $S = x \cup y \wedge x \cap y = \emptyset$
Specialised use for enumerated sets:
 $partition(S, \{A\}, \{B\}, \{C\})$.
 $S = \{A, B, C\} \wedge A \neq B \wedge B \neq C \wedge C \neq A$

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

3 BOOL and bool

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

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

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

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

14. Cardinality: $card(S)$ $card(S)$
Defined only for $finite(S)$.

4 Numbers

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

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.

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

1. The set of integer numbers: \mathbb{Z} \mathbb{Z}

2. The set of natural numbers: \mathbb{N} \mathbb{N}

3. The set of positive natural numbers: \mathbb{N}_1 \mathbb{N}_1
 $\mathbb{N}_1 = \mathbb{N} \setminus \{0\}$

4. Minimum: $min(S)$ $min(S)$
 $S \subset \mathbb{Z}$ and $finite(S)$ or S must have a lower bound.

5. Maximum: $max(S)$ $max(S)$
 $S \subset \mathbb{Z}$ and $finite(S)$ or S must have an upper bound.

17. Quantified union: $\cup z \cdot P \mid S$
 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 $\neg free(x, z, T)$, $\neg free(x, P)$, $\neg free(x, S)$,
 $\neg free(x, z)$

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
 $n \neq 0$

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

10. Remainder: $m \bmod n$ $m \bmod n$
 $n \neq 0$

11. Interval: $m .. n$ $m .. n$
 $m .. n = \{i \mid m \leq i \wedge i \leq n\}$

2.1 Set predicates

1. Set membership: $E \in S$ $E \in S$

2. Set non-membership: $E \notin S$ $E \notin S$

3. Subset: $S \subseteq T$ $S \subseteq T$

4. Not a subset: $S \not\subseteq T$ $S \not\subseteq T$

5. Proper subset: $S \subset T$ $S \subset T$

6. Not a proper subset: $s \not\subset t$ $s \not\subset t$

4.1 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: $m \leq n$ $m \leq n$

5 Relations

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

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

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

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

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

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

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

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

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

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

10. Domain restriction: $S \triangleleft r$ $S \triangleleft r$
 $S \triangleleft r = \{x \mapsto y \mid x \mapsto y \in r \wedge x \in S\}$.

11. Domain subtraction: $S \triangleleft r$ $S \triangleleft r$
 $S \triangleleft r = \{x \mapsto y \mid x \mapsto y \in r \wedge x \notin S\}$.

12. Range restriction: $r \triangleright T$ $r \triangleright T$
 $r \triangleright T = \{x \mapsto y \mid x \mapsto y \in r \wedge y \in T\}$.

13. Range subtraction: $r \triangleright T$ $r \triangleright T$
 $r \triangleright T = \{x \mapsto y \mid y \in r \wedge y \notin T\}$.

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

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

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: $p \otimes q$ $p \otimes q$
 $p \otimes q = \{x \mapsto (y \mapsto z) \mid x \mapsto y \in p \wedge x \mapsto z \in q\}$.

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

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

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

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^n
 $r \in S \leftrightarrow S \Rightarrow r^0 = S \triangleleft \text{id} \wedge r^{n+1} = r ; r^n$.
 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 $\text{iterate}(r \mapsto n)$.

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

3. Irreflexive Closure: r^+ r^+
 $r^+ = \bigcup n. (n \in \mathbb{N}_1 \mid r^n)$.
 Could be defined as a function $\text{iclosure}(r)$.
 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: $S \rightarrow T$ $S \rightarrow T$
 $S \rightarrow T = \{r. r \in S \leftrightarrow T \wedge r^{-1} ; r \subseteq T \triangleleft \text{id}\}$.

2. Total functions: $S \rightarrow T$ $S \rightarrow T$
 $S \rightarrow T = \{f. f \in S \rightarrow T \wedge \text{dom}(f) = S\}$.

3. Partial injections: $S \rightarrow T$ $S \rightarrow T$
 $S \rightarrow T = \{f. f \in S \rightarrow T \wedge f^{-1} \in T \rightarrow S\}$.
One-to-one relations.

4. Total injections: $S \rightarrow T$ $S \rightarrow T$
 $S \rightarrow T = S \rightarrow T \cap S \rightarrow T$.

5. Partial surjections: $S \rightarrow T$ $S \rightarrow T$
 $S \rightarrow T = \{f. f \in S \rightarrow T \wedge \text{ran}(f) = T\}$.
Onto relations.

6. Total surjections: $S \rightarrow T$ $S \rightarrow T$
 $S \rightarrow T = S \rightarrow T \cap S \rightarrow T$.

7. Bijections: $S \rightarrow T$ $S \rightarrow T$
 $S \rightarrow T = S \rightarrow T \cap S \rightarrow T$.
One-to-one and onto relations.

8. Lambda abstraction:

$(\lambda p. P \mid E)$

$(\%p. P \mid E)$

P must *constrain* the variables in p .

$(\lambda p. P \mid E) = \{z. P \mid p \mapsto E\}$, where z is a list of variables that appear in the pattern p .

9. Function application: $f(E)$

$f(E)$

$E \mapsto y \in f \Rightarrow E \in \text{dom}(f) \wedge f \in X \rightarrow Y$, where $\text{type}(f) = \mathbb{P}(X \times Y)$.

Note: in Event-B, relations and functions only ever have one argument, but that argument may be a pair or tuple, hence $f(E \mapsto F)$ $f(E \mapsto F)$ $f(E, F)$ 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: $z \mid P$

$z \mid P$

\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.