

7. Multiple substitution:

$x, y := E, F$

$\boxed{x, y := E, F}$

Concurrent substitution of the values E and F for the free occurrences of x and y , respectively.

8. Parallel substitution: $G \parallel H$

$\boxed{G \parallel H}$

Apply the substitutions G and H concurrently. Parallel substitution is not given a general substitution semantics; it is eliminated by rewriting rules. Notice

$x := E \parallel y := F = x, y := E, F.$

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 $\boxed{\text{ASCII form}}$ 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;

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 $\text{notfree}(z, E)$ 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: $\text{notfree}(z, \forall z \cdot P)$, $\text{notfree}(z, \exists z \cdot P)$, $\text{notfree}(z, \{z|P\})$, $\text{notfree}(z, \lambda z \cdot (P|E))$, and $\neg \text{notfree}(z, z)$.

1 Predicates

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

1. False: \perp $\boxed{\text{false}}$
2. True: \top $\boxed{\text{true}}$

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**: `bool(\perp) = FALSE` and `bool(\top) = TRUE`.

1. Conjunction: $P \wedge Q$ $\boxed{P \ \& \ Q}$
2. Disjunction: $P \vee Q$ $\boxed{P \ \text{or} \ Q}$
3. Implication: $P \Rightarrow Q$ $\boxed{P \Rightarrow Q}$
4. Equivalence: $P \Leftrightarrow Q$
 $P \Leftrightarrow Q = P \Rightarrow Q \wedge Q \Rightarrow P$ $\boxed{P \ \Leftrightarrow \ Q}$
5. Negation: $\neg P$ $\boxed{\text{not } P}$
6. Universal quantification:
 $\forall z \cdot P \Rightarrow Q$ $\boxed{\forall z \cdot P \Rightarrow Q}$
For all values of z satisfying P , Q (is true)
 P must *constrain* the variables in z .
7. Existential quantification:
 $\exists z \cdot P \wedge Q$ $\boxed{\exists z \cdot P \ \& \ Q}$
There exists some values of z satisfying P for which Q . P must *constrain* the variables in z .
8. Substitution: $[G] P$ $\boxed{[G] \ P}$
9. Equality: $E = F$ $\boxed{E = F}$
10. Inequality: $E \neq F$ $\boxed{E \ / = \ F}$

2 Sets

1. Singleton set: $\{E\}$ $\boxed{\{E\}}$
2. Set enumeration: $\{E, F\}$ $\boxed{\{E, F\}}$
See note on the pattern E, F at top of summary.
3. Empty set: \emptyset $\boxed{\{\}}$
4. Set comprehension: $\{z \cdot P \mid E\}$ $\boxed{\{z \ . \ P \mid E\}}$
The set of all values of the expression E for z satisfying the predicate P . P must *constrain* the variables in z .
5. Union: $S \cup T$ $\boxed{S \ \cup \ T}$
6. Intersection: $S \cap T$ $\boxed{S \ \& \ T}$
7. Difference: $S \setminus T$
 $S - T = \{x \mid x \in S \wedge x \notin T\}$ $\boxed{S \ \setminus \ T}$
8. Ordered pair: $E \mapsto F$ $\boxed{E \ \mapsto \ F}$
 $E \mapsto F = E, F$
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: $\{x \mapsto y \mid P\}$.
9. Cartesian product: $S \times T$ $\boxed{S \ \ast \ T}$
 $S \times T = \{x \mapsto y \mid x \in S \wedge y \in T\}$
10. Powerset: $\mathbb{P}(S)$ $\boxed{\text{POW}(S)}$
 $\mathbb{P}(S) = \{s \mid s \subseteq S\}$
11. Non-empty subsets: $\mathbb{P}_1(S)$ $\boxed{\text{POW1}(S)}$
 $\mathbb{P}_1(S) = \mathbb{P}(S) - \{\emptyset\}$
12. Finite set: `finite S` $\boxed{\text{finite } S}$
Declares S to be finite.
13. Cardinality: `card(S)` $\boxed{\text{card}(S)}$
Defined only for finite sets
14. Generalized union: `union(U)` $\boxed{\text{union}(U)}$
The union of all the elements of U .
 $\forall U \cdot U \in \mathbb{P}(\mathbb{P}(S)) \Rightarrow$
 $\text{union}(U) = \{x \mid x \in S \wedge (\exists s \cdot s \in U \wedge x \in s)\}$
where x, s *notfree* U

¹Version May 12, 2008©1996-2008 Ken Robinson

11. Interval: $m \dots n$ `m .. n`
10. Remainder: $mod n$ `m mod n`
9. Quotient: m/n `m / n`
8. Product: $m \times n$ `m * n`
7. Difference: $m - n$ `m - n`
6. Sum: $m + n$ `m + n`
5. Maximum: $max(S)$
 $S \subset \mathbb{Z}$ and finite(S) `max(S)`
4. Minimum: $min(S)$
 $S \subset \mathbb{Z}$ and finite(S) `min(S)`
3. The set of positive natural numbers: \mathbb{N}_1 `NAT1`
2. The set of natural numbers: \mathbb{N} `NAT`
1. The set of integer numbers: \mathbb{Z} `INT`

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.

3 Numbers

6. Not a proper subset: $s \not\subset t$ `S /<<: T`
5. Proper subset: $S \subset T$ `S <<: T`
4. Not a subset: $S \not\subseteq T$ `S /<: T`
3. Subset: $S \subseteq T$ `S <: T`
2. Set non-membership: $E \notin S$ `E /: S`
1. Set membership: $E \in S$ `E : S`

2.1 Set predicates

17. Generalized intersection:
 $\bigcup z.P | E$
where $z \text{ notfree } T, x \text{ notfree } T, P, E, z$
 $\bigcup z.P | E = \{x \mid x \in T \wedge (\exists z.P \wedge x \in E)\}$
 $(\forall z.P \Rightarrow E \subseteq T) \Leftrightarrow$
 $\{z \mid P\} \neq \emptyset, (z.P \Rightarrow E \subseteq T) \Leftrightarrow$
 $\bigcup z.P | E = \{x \mid x \in T \wedge (\forall s.s \in U \Rightarrow x \in s)\}$
where $x, s \text{ notfree } U$
16. Generalized union:
 $\bigcup z.P | E$
 $\bigcup z.P | E = \mathbb{P}(\mathbb{P}(S)) \Leftrightarrow$
 $\text{inter}(U) = \{x \mid x \in S \wedge (\forall s.s \in U \Rightarrow x \in s)\}$
where $x, s \text{ notfree } U$
15. Generalized intersection: $\text{inter}(U)$ `inter(U)`
The intersection of all the elements of U .

4 Relations

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`

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

1. Relations: $S \leftrightarrow T$ `S <-> T`
2. Domain: $\text{dom}(r)$
 $\forall r.r \in S \leftrightarrow T \Leftrightarrow$
 $\text{dom}(r) = \{x \mid (\exists y.x \mapsto y \in r)\}$
3. Range: $\text{ran}(r)$
 $\forall r.r \in S \leftrightarrow T \Leftrightarrow$
 $\text{ran}(r) = \{y \mid (\exists x.x \mapsto y \in r)\}$
4. Total relation: $S \leftrightarrow T$
if $r \in S \leftrightarrow T$ then $\text{dom}(r) = S$
5. Surjective relation: $S \leftrightarrow T$
if $r \in S \leftrightarrow T$ then $\text{ran}(r) = T$
6. Total surjective relation: $S \leftrightarrow T$
if $r \in S \leftrightarrow T$ then $\text{dom}(r) = S$ and $\text{ran}(r) = T$
7. Forward composition: $d : p$
 $\forall p, q. p \in S \leftrightarrow T \vee q \in T \leftrightarrow U \Leftrightarrow$
 $d = \{x \mapsto y \mid (\exists z.z \in p \vee z \mapsto y \in q)\}$
8. Backward composition: $p \circ q$
 $d \circ q = p$
9. Identity: $\text{id}(S)$
 $\text{id}(S) = \{x \mapsto y \mid x \in S \wedge y \in S \wedge x = y\}$
10. Domain restriction: $S \triangleright r$
 $S \triangleright r = \{x \mapsto y \mid x \mapsto y \in r \wedge x \in S\}$
11. Domain subtraction: $S \triangleleft r$
 $S \triangleleft r = \{x \mapsto y \mid x \mapsto y \in r \wedge x \notin S\}$
12. Range restriction: $r \triangleleft T$
 $r \triangleleft T = \{x \mapsto y \mid x \mapsto y \in r \wedge y \in T\}$
13. Range subtraction: $r \triangleleft T$
 $r \triangleleft T = \{x \mapsto y \mid x \mapsto y \in r \wedge y \notin T\}$
14. Inverse: r^{-1}
 $r^{-1} = \{y \mapsto x \mid x \mapsto y \in r\}$
15. Relational image: $r[S]$
 $r[S] = \{y \mid \exists x.x \in S \wedge x \mapsto y \in r\}$
16. Overriding: $r_1 \triangleleft \triangleright r_2$
 $r_1 \triangleleft \triangleright r_2 = r_2 \cup (\text{dom}(r_2) \triangleleft r_1)$
17. Direct product: $p \otimes q$
 $p \otimes q = \{x \mapsto y \mid (x \mapsto y \in p \vee x \mapsto z \in q)\}$

3.1 Number predicates

18. Parallel product: $p \parallel q$ `p || q`
 $p \parallel q = \{(x \mapsto y) \mapsto (m \mapsto n) \mid x \mapsto m \in p \wedge y \mapsto n \in q\}$
19. Iteration: r^n `iterate(r,n)`
 $r \in S \leftrightarrow S \Rightarrow r^0 = \text{id}(S) \wedge r^{n+1} = r \circ r^n$
Note: to avoid inconsistency S should be the finite base set for r , ie the smallest set for which all $r \in S \leftrightarrow S$.
20. Closure: r^* `closure(r)`
 $r^* = \bigcup n.(n \in \mathbb{N} \mid r^n)$
21. Projection: $\text{prj}_1(S, T)$ `prj1(S,T)`
 $\text{prj}_1(S, T) = \{(x \mapsto y), z \mid x \mapsto y \in S \times T \wedge z = x\}$
22. Projection: $\text{prj}_2(S, T)$ `prj2(S,T)`
 $\text{prj}_2(S, T) = \{(x \mapsto y) \mapsto z \mid x \mapsto y \in S \times T \wedge z = y\}$

4.1 Functions

1. Partial functions: $S \mapsto T$ `S -> T`
 $S \mapsto T = \{r \in S \leftrightarrow T \vee r^{-1} : r \subseteq \text{id}(T)\}$
2. Total functions: $S \mapsto T$ `S ->> T`
 $S \mapsto T = \{f \mid f \in S \mapsto T \wedge \text{dom}(f) = S\}$
3. Partial injections: $S \mapsto T$ `S >+ T`
 $S \mapsto T = \{f \mid f \in S \mapsto T \wedge f^{-1} \subseteq \text{id}(T) \wedge S \mapsto T\}$
4. Total injections: $S \mapsto T$ `S >-> T`
 $S \mapsto T = S \mapsto T \cup S \mapsto T$
5. Partial surjections: $S \mapsto T$ `S +->> T`
 $S \mapsto T = \{f \mid f \in S \mapsto T \vee \text{ran}(f) = T\}$
6. Total surjections: $S \mapsto T$ `S -->> T`
 $S \mapsto T = S \mapsto T \cup S \mapsto T$
7. Bijections: $S \mapsto T$ `S >->> T`
 $S \mapsto T = S \mapsto T \cup S \mapsto T$
8. Lambda abstraction:
 $\lambda z.(P | E)$
where $y \text{ notfree } P$ and $y \text{ notfree } E$.
9. Function application: $f(E)$ `f(E)`
 $E \mapsto y \in f \Leftrightarrow f(E) = y$

1. Context machines: contain sets and constants
used by other machines.
2. The null substitution: $skip$ `skip`
3. Simple substitution: $x := E$ `x := E`
4. Choice from set: $x \in S$ `x :: S`
5. Choice by predicate: $x : P$ `x : | P`
6. Functional override: $f(x) := E$ `f(x) := E`

5 Machines

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`
3. Simple substitution: $x := E$ `x := E`
4. Choice from set: $x \in S$ `x :: S`
5. Arbitrarily choose a value that satisfies the predicate P . P must *constrain* the variable x .
Choose by predicate: $x : P$ `x : | P`
6. Functional override: $f(x) := E$ `f(x) := E`
Substitute the value E for the expression f at point x .
 $f(x) := E = f := f \triangleleft \triangleright \{x \mapsto E\}$.

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 (string) 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. There is one distinguished event name *INITIALISA* used to initialise the variables of a machine.
Event name: `INITIALISA`
Event identifiers: `REFINES`
ANY Identifiers
WHERE Predicates
WITH Witnesses
THEN Actions
END
2. Event machines: contain events.
MACHINE Identifier
REFINES Machine identifiers
VARIABLES Identifiers
INVARIANT Predicates
THEOREMS Predicates
VARIANT Expression
EVENTS Events
END
3. Event machines: contain events.
MACHINE Identifier
REFINES Machine identifiers
VARIABLES Identifiers
INVARIANT Predicates
THEOREMS Predicates
VARIANT Expression
EVENTS Events
END
4. Event machines: contain events.
MACHINE Identifier
REFINES Machine identifiers
VARIABLES Identifiers
INVARIANT Predicates
THEOREMS Predicates
VARIANT Expression
EVENTS Events
END
5. Event machines: contain events.
MACHINE Identifier
REFINES Machine identifiers
VARIABLES Identifiers
INVARIANT Predicates
THEOREMS Predicates
VARIANT Expression
EVENTS Events
END
6. Event machines: contain events.
MACHINE Identifier
REFINES Machine identifiers
VARIABLES Identifiers
INVARIANT Predicates
THEOREMS Predicates
VARIANT Expression
EVENTS Events
END
7. Event machines: contain events.
MACHINE Identifier
REFINES Machine identifiers
VARIABLES Identifiers
INVARIANT Predicates
THEOREMS Predicates
VARIANT Expression
EVENTS Events
END
8. Event machines: contain events.
MACHINE Identifier
REFINES Machine identifiers
VARIABLES Identifiers
INVARIANT Predicates
THEOREMS Predicates
VARIANT Expression
EVENTS Events
END
9. Event machines: contain events.
MACHINE Identifier
REFINES Machine identifiers
VARIABLES Identifiers
INVARIANT Predicates
THEOREMS Predicates
VARIANT Expression
EVENTS Events
END
10. Event machines: contain events.
MACHINE Identifier
REFINES Machine identifiers
VARIABLES Identifiers
INVARIANT Predicates
THEOREMS Predicates
VARIANT Expression
EVENTS Events
END
11. Event machines: contain events.
MACHINE Identifier
REFINES Machine identifiers
VARIABLES Identifiers
INVARIANT Predicates
THEOREMS Predicates
VARIANT Expression
EVENTS Events
END
12. Event machines: contain events.
MACHINE Identifier
REFINES Machine identifiers
VARIABLES Identifiers
INVARIANT Predicates
THEOREMS Predicates
VARIANT Expression
EVENTS Events
END
13. Event machines: contain events.
MACHINE Identifier
REFINES Machine identifiers
VARIABLES Identifiers
INVARIANT Predicates
THEOREMS Predicates
VARIANT Expression
EVENTS Events
END
14. Event machines: contain events.
MACHINE Identifier
REFINES Machine identifiers
VARIABLES Identifiers
INVARIANT Predicates
THEOREMS Predicates
VARIANT Expression
EVENTS Events
END
15. Event machines: contain events.
MACHINE Identifier
REFINES Machine identifiers
VARIABLES Identifiers
INVARIANT Predicates
THEOREMS Predicates
VARIANT Expression
EVENTS Events
END
16. Event machines: contain events.
MACHINE Identifier
REFINES Machine identifiers
VARIABLES Identifiers
INVARIANT Predicates
THEOREMS Predicates
VARIANT Expression
EVENTS Events
END
17. Event machines: contain events.
MACHINE Identifier
REFINES Machine identifiers
VARIABLES Identifiers
INVARIANT Predicates
THEOREMS Predicates
VARIANT Expression
EVENTS Events
END
18. Event machines: contain events.
MACHINE Identifier
REFINES Machine identifiers
VARIABLES Identifiers
INVARIANT Predicates
THEOREMS Predicates
VARIANT Expression
EVENTS Events
END
19. Event machines: contain events.
MACHINE Identifier
REFINES Machine identifiers
VARIABLES Identifiers
INVARIANT Predicates
THEOREMS Predicates
VARIANT Expression
EVENTS Events
END
20. Event machines: contain events.
MACHINE Identifier
REFINES Machine identifiers
VARIABLES Identifiers
INVARIANT Predicates
THEOREMS Predicates
VARIANT Expression
EVENTS Events
END
21. Event machines: contain events.
MACHINE Identifier
REFINES Machine identifiers
VARIABLES Identifiers
INVARIANT Predicates
THEOREMS Predicates
VARIANT Expression
EVENTS Events
END
22. Event machines: contain events.
MACHINE Identifier
REFINES Machine identifiers
VARIABLES Identifiers
INVARIANT Predicates
THEOREMS Predicates
VARIANT Expression
EVENTS Events
END