Event-B mathematical basis: predicate calculus

- A predicate is a logical expression, which can be evaluated to the constants *TRUE* or *FALSE* (of the predefined type *BOOL*), for example, $x \geq y$, $n \geq 0$, $x \in S$, $y \subseteq S$, or $z = Exp(x, y)$
- Standard logical constants and operations in Event-B (graphical notation, followed by the equivalent ascii notation):

$\wedge$	&	logical conjunction
$\vee$	or	logical disjunction
$\Rightarrow$	=>	logical implication
$\Leftrightarrow$	<=>	logical equivalence
$\neg$	not	logical negation
$\forall z.\ P \Rightarrow Q$	!z. P => Q	universal quantification
$\exists z.\ P \wedge Q$	#z. P & Q	existential quantification

Some set constants and operations

Graphical notation, followed by the equivalent ascii notation:

$\varnothing$	$\{\}$	empty set
$\{e_1, e_2, ..., e_n\}$	$\{e_1, e_2, ..., e_n\}$	enumerated set
$n_1..n_2$	$n_1..n_2$	interval set between numbers n_1 and n_2
$e \in S$	$e : S$	set membership
$e \notin S$	$e /: S$	"e does not belong to S", i.e.
$S \subseteq T$	$S <: T$	set inclusion
$S \not\subseteq T$	$S /<: T$	"S is not included in T"
$S \cup T$	S \/ T	set union
$S \cap T$	S /\ T	set intersection
$S \setminus T$	S \ T	set difference (subtraction)

Predefined sets like $\mathbb{N}(NAT)$ for natural numbers, $\mathbb{N}1(NAT1)$ for positive natural numbers, $BOOL$=$\{TRUE, FALSE\}$ for truth values, etc.

Some other set expressions

$\{z \mid z \in R \wedge P\}$	Set comprehension: "the subset of R such that P"
$S \times T$ (ascii S**T)	cartesian product $S \times T = \{(x, y) \mid x \in S \wedge x \in T\}$
$card(S)$	cardinality: the number of set elements
$\mathbb{P}(S)$ (ascii POW(S))	power set: the set of all subsets of S
$\mathbb{P}1(S)$ (ascii POW1(S))	all non-empty subsets of S

Event-B conventions

- *partition* is a shorthand definition of a set that can be partitioned into separate, disjoint parts (subsets)
- If the parts (subsets) are singleton sets, e.g., of the form $\{element\}$, such a definition defines of an enumerated set
- For instance, *partition(Set,{element1}, {element2}, {element3})* is a shorthand for the axioms
  - $element1 \in Set$,
  - $element2 \in Set$,
  - $element3 \in Set$,
  - $element1 \neq element2$,
  - $element1 \neq element3$,
  - $element2 \neq element3$,
  - $Set = \{element1, element2, element3\}$.
- In general, any disjoint subsets instead of *{element}*, e.g., *partition(ITEM, Available, Sold)*

Relations (cont.)

- A relation R between sets S and T can be represented as a set of pairs (s, t) representing those elements of S and T that are related
- A pair is syntactically represented in Event-B as $(s \mapsto t)$ or (s,, t) in ascii
- Mathematically, a relation between sets S and T is a member of $\mathbb{P}(S \times T)$, i.e., a subset of $S \times T$
- Reminder: $S \times T$ – all possible pairs from S and T
- Shorthand notation: $S \leftrightarrow T \equiv \mathbb{P}(S \times T)$
- In other words, $R \in S \leftrightarrow T$ is equivalent to $R \in \mathbb{P}(S \times T)$ or $R \subseteq S \times T$

Relation domain and range

- The *domain* of a relation $R \in S \leftrightarrow T$ is the subset of elements of S that are related to something in T
- Relation domain (denoted as $dom(R)$) is defined by $\{x \mid x \in S \land \exists y.\ (y \in T \land (x, y) \in R)\}$
- Example: $dom(owns\_camera) = \{Jonas, Vaidas, Vaiva, Sandra\}$
- The *range* of a relation $R \in S \leftrightarrow T$ is the subset of elements of T that are related to something in S
- Relation range (denoted as $ran(R)$) is defined by $\{y \mid y \in T \land \exists x.\ (x \in S \land (x, y) \in R)\}$
- Example: $ran(owns\_camera) = \{Canon, Nikon, Sony, Pentax\}$

Relation filtering operations

Graphical notation, followed by the equivalent ascii notation:

$S \lhd R$	S <| R	domain restriction
$S \hlhd R$	S <<| R	domain subtraction
$R \rhd S$	R |> S	range restriction
$R \rhd\!\!\rhd S$	R |>> S	range subtraction

Examples:

$\{Jonas\} \lhd owns\_camera = \{Jonas \mapsto Canon,\ Jonas \mapsto Sony\}$
$\{Mindaugas\} \lhd owns\_camera = \varnothing$
$\{Jonas,\ Vaiva\} \hlhd owns\_camera = \{Vaidas \mapsto Nikon,\ Sandra \mapsto Pentax\}$
$\{Mindaugas\} \hlhd owns\_camera = owns\_camera$

$owns\_camera \rhd \{Sony\} = \{Vaiva \mapsto Sony,\ Jonas \mapsto Sony\}$
$owns\_camera \rhd\!\!\rhd \{Sony, Canon, Pentax\} = \{Vaidas \mapsto Nikon\}$

Basic operations with relations

- Initialising (for $R \in S \leftrightarrow T$):
  $R := \{s_1 \mapsto t_1,\ s_2 \mapsto t_2,\ ...\}$ or $R := S \times \{t_0\}$
  where $s_i \mapsto t_i$ are constructed pairs from $S \leftrightarrow T$
- Adding, merging elements (using set operations):
  $R := R \cup \{s \mapsto t\}$ or $R := R \cup Q$
  where Q is a relation of the same type, i.e., $Q \in S \leftrightarrow T$
- Checking membership or inclusion:
  $s \mapsto t \in R$ or $R \subseteq Q$
- Removing elements (filtering on the relation domain):
  $R := \{s_1, s_2, ...\} \lhd R$ or $R := \{s_1, s_2, ...\} \lhd\!\!\!\!- R$
- Removing elements (filtering on the relation range):
  $R := R \rhd \{t_1, t_2, ...\}$ or $R := R \rhd\!\!\!\!- \{t_1, t_2, ...\}$

More operations on relations

- Inverse relation R^{-1} (ascii R~).
  Includes all such $(x \mapsto y)$ that $(y \mapsto x) \in R$
- Example: for the relation
  $owns\_camera = \{Jonas \mapsto Canon, Vaidas \mapsto Nikon,$
  $Vaiva \mapsto Sony, Jonas \mapsto Sony, Sandra \mapsto Pentax\}$
  its inverse is
  $owns\_camera^{-1} = \{Canon \mapsto Jonas, Nikon \mapsto Vaidas,$
  $Sony \mapsto Vaiva, Sony \mapsto Jonas, Pentax \mapsto Sandra\}$
- Relational image $R[S]$. Returns a subset of the relation range related to any element from the given set S
- Example: for the relation *owns_camera* defined above
  $owns\_camera[\{Jonas, Vaiva\}] = \{Canon, Sony\}$

More operations on relations (cont.)

- Relational composition $R_1; R_2$.
  Composition of relations $R_1 \in S \leftrightarrow T$ and $R_2 \in T \leftrightarrow U$ is relation
  $\{(s \mapsto u) \mid \exists t.\ (s \mapsto t) \in R_1 \land (t \mapsto u) \in R_2\}$

- Example: Suppose we are given a relation between camera models and their brands $cmodels \in CMODEL \leftrightarrow CAMERA$,
  for instance,
  $cmodels = \{350 \mapsto Canon, 60D \mapsto Canon, D5200 \mapsto Nikon, \ldots\}$

  Then the previously defined $owns\_camera$ can be built as a result of a composition of the relations $owns\_cmodel \in PERSON \leftrightarrow CMODEL$ and $cmodels \in CMODEL \leftrightarrow CAMERA$, i.e.,

  $owns\_camera = owns\_cmodel ; cmodels$

More operations on relations (cont.)

- Relational overriding $R_1 \lhd\!\!+ R_2$ (ascii R_1 <+ R_2).

  Updating the relation R_1 by R_2:
  $R_1 \lhd\!\!+ R_2 \;=\; (dom(R_2) \lhd R_1) \cup R_2$

- Example:
  $owns\_camera \lhd\!\!+ \{Jonas \mapsto Nikon\}$

  Two pairs $Jonas \mapsto Canon$ and $Jonas \mapsto Sony$ are removed, one pair $Jonas \mapsto Nikon$ is added

More operations on relations (cont.)

- Identity relation: $id \in S \leftrightarrow S$

  The relation contains pairs $(s \mapsto s)$ for all $s \in S$
  (each element is only related to itself)

- Example: suppose we have the relation

$$connected \in CITY \leftrightarrow CITY$$

  containing all the road connections between cities
  (encoding a graph representation of roads between cities)

  Then the requirements "Roads can only be between different cities" can be formulated as

$$connected \cap id = \varnothing$$

Functions

- Functions form a special class of relations that satisfy additional requirement: any element of the source set can be related to no more than 1 element of the target
- Functionality requirement mathematically:

$$\forall x, y, z.\ \ (x \mapsto y) \in R \ \wedge\ (x \mapsto z) \in R \ \Rightarrow\ \ y = z$$

- Any operation applicable to a relation or a set is also applicable to a function. For example, we can talk about the domain and the range of a function or a function as a set of pairs
- If f is a function, then $f(x)$ is the result of the function f for the argument x

Total and partial functions

- Functions are called *total* if their domain is the whole source set
  Syntax: $f \in S \rightarrow T$ (or ascii $f : S \text{ --> } T$)
  where $dom(f) = S$ and $ran(f) \subseteq T$
- Example: the camera model relation is actually a total function
  $cmodels \in CMODEL \rightarrow CAMERA$
  (any camera model identifier is uniquely associated with its brand)
- Functions are called *partial* if their domain is a subset of the source set
  Syntax: $f \in S \nrightarrow T$ (or ascii $f : S \text{ +-> } T$)
  where $dom(f) \subseteq S$ and $ran(f) \subseteq T$
- Example: room reservation relation is actually a partial function
  $reserved \in ROOM \nrightarrow CUSTOMER$
  (each reserved room has the unique customer that served it, however, not all rooms must be reserved)

Arrays

- An array is a named, indexed collection of values of a given type.
- The array values can be accessed (read and updated) by using appropriate indexes.
- If we use $1..n$ (for some $n \in \mathbb{N}$) as our index set, then an array (containing elements of type S) can be modelled as a function from $1..n$ to S.
- In fact, any set can be used as the index set for arrays. Therefore, arrays can be usually modelled as total functions from S (index set) to T (the type of array values).

Functional (array) assignment

- The notation used to describe machine actions (i.e., assignments in the machine events) allows us to directly assign values to indexed elements of arrays:

$$a(i) := E$$

- This is just syntactic sugaring for the following assignment:

$$a := a \lhd \{(i \mapsto E)\}$$

- The assignment also works if a is modelled as a partial function. However, if we want to check/read values from such an array, we have to ensure/prove (by using the event guards and/or machine invariants) that the used index belongs to the function domain, i.e., $i \in dom(a)$

- $reserved(r) := FALSE$ **OK!**
- $nItems(j) := nItems(i) + 1$ **only if** $i \in dom(nItems)$

Varieties of functions

Suppose we have a function f (from the source X to the target Y). Then it is called

Total function	$\rightarrow$	-->	if $dom(f) = X,\ ran(f) \subseteq Y$
Partial function	$\pfun$	+->	if $dom(f) \subseteq X,\ ran(f) \subseteq Y$
Total injection	$\rightarrowtail$	>->	if $dom(f) = X,\ ran(f) \subseteq Y$ and one-to-one function
Partial injection	$\pinj$	>+>	if $dom(f) \subseteq X,\ ran(f) \subseteq Y$ and one-to-one function
Total surjection	$\twoheadrightarrow$	–>>	if $dom(f) = X,\ ran(f) = Y$
Partial surjection	$\psurj$	+->>	if $dom(f) \subseteq X,\ ran(f) = Y$
(Total) Bijection	$\bij$	>->>	if $dom(f) = X,\ ran(f) = Y$ and one-to-one function

Varieties of functions: examples

- Injection: a function with 1-to-1 relationship between the source and target sets (e.g., an array without repeating elements)
- Example: $VU\_id \in PERSON \rightarrowtail\!\!\!\!\!\!\!\!\rightarrow ID$

  It is a partial injection: not all persons have a VU identification number, however, id is unique for each person
- Advantage: a reverse relation for an injection is also a function!
- Example: $VU\_id^{\sim} \in ID \rightarrowtail PERSON$

  A total injection this time
- Other examples:
  $Capital \in \text{COUNTRY} \rightarrowtail CITY$, and
  $Capital\_of \in CITY \rightarrowtail\!\!\!\!\!\!\!\!\rightarrow COUNTRY$ (where $Capital\_of = Capital^{\sim}$)

Varieties of functions: examples

- Surjection: a function that covers all the target set
- Example:
  $married \in WIFE \twoheadrightarrow HUSBAND$

  It is a total surjection
- Another example:
  $Capital\_of \in CITY \nrightarrow COUNTRY$

  A partial surjection this time

Varieties of functions: examples

- Bijection: a total function that is both injection and surjection
- Example:
  $married \in WIFE \rightarrowtail\!\!\!\!\!\rightarrow HUSBAND$

  It is a bijection (in many countries)
- Bijections relate sets with the same power (length)
- Another example:
  $VU\_account \in ID \rightarrowtail\!\!\!\!\!\rightarrow VU\_ACCOUNT$

  A bijection: both sets are of the same length and one-to-one relationships in both directions