Lecture 3

The B Method

AMN, GSL, Examples, Tools

Lecture Outline

- Abstract Machine Notation (AMN)
- Consistency of an Abstract Machine
- Generalised Substitution Language (GSL)
- B Mathematical Notation - Relations, Functions, Sequences
- *Deliveries* Example (see attached requirements/AMN spec/proof files)
- AtelierB tool (see `https://www.atelierb.eu/en/`)

Abstract Machine Notation (AMN)

- Consists of a number of clauses allowing the definition of *abstract machines*: `MACHINE`, `CONSTRAINTS`, `SETS`, `CONSTANTS`, ...

```
MACHINE
  M(X, x)
CONSTRAINTS
  C
SETS
  S;
  T = {a, b}
(ABSTRACT_)CONSTANTS
  c
PROPERTIES
  P
DEFINITIONS
  D
(CONCRETE_)VARIABLES
  v
INVARIANT
  I
ASSERTIONS
  J
INITIALIZATION
  U
OPERATIONS
  u ← O(w) ≙
    PRE
      Q
    THEN
      V
    END;
  ...
END
```

Consistency of an Abstract Machine

- Establishing the internal consistency of an abstract machine involves proving that
  - its context and invariant ensure the assertions

$$A \wedge B \wedge C \wedge P \wedge I \Rightarrow J \tag{1}$$

  - within the context in question, the initialization ensures the invariant and each of the operations preserves it

$$A \wedge B \wedge C \wedge P \Rightarrow [U]I \tag{2}$$

$$A \wedge B \wedge C \wedge P \wedge I \wedge J \wedge Q \Rightarrow [V]I \tag{3}$$

- Contextual abbreviations used by the above proof obligations

Abbreviation	**Definition**
A	$X \in \mathbb{P}_1(\mathrm{INT})$
B	$S \in \mathbb{P}_1(\mathrm{INT}) \wedge T \in \mathbb{P}_1(\mathrm{INT}) \wedge T = \{a, b\} \wedge a \neq b$

Generalised Substitution Language (GSL)

- Consists of a number of *generalised substitutions* allowing the definition of abstract machine operations
- Generalised substitutions are an extension of the usual substitutions used in mathematics, the effect of which consists in the transformation of formulas on which they are applied
- A generalised substitution S is defined by the way it transforms an arbitrary predicate P, the meaning of the new predicate being axiomatized
  - Notation $[S]P$, read "S establishes P"

Elementary Substitutions

Substitution (syntax)		**Matching axiom** (semantics)
$v := E$	(simple substitution)	$[v := E]\, R \;\Leftrightarrow\; R$ with all free occurrences of v replaced by E
skip	(no effect substitution)	$[\text{skip}]\, R \;\Leftrightarrow\; R$
$P \mid S$	(preconditioned substitution)	$[P \mid S]\, R \;\Leftrightarrow\; (P \wedge [S]\, R)$
$P \Longrightarrow S$	(guarded substitution)	$[P \Longrightarrow S]\, R \;\Leftrightarrow\; (P \Rightarrow [S]\, R)$
$S \,[]\, T$	(bounded choice substitution)	$[S \,[]\, T]\, R \;\Leftrightarrow\; ([S]\, R \wedge [T]\, R)$
$@\, x \,.\, S$	(unbounded choice substitution)	$[@\, x \,.\, S]\, R \;\Leftrightarrow\; (\forall\, x \,.\, [S]\, R)$, if $x \backslash R$

Syntactic Extensions of Elementary Substitutions

Syntax	**Definition**
BEGIN S END	S
$x := E \,\|\, y := F$	$x, y := E, F$
PRE P THEN S END	$P \mid S$
IF P THEN S ELSE T END	$(P \Longrightarrow S) \, [] \, (\neg P \Longrightarrow T)$
IF P THEN S END	IF P THEN S ELSE skip END
CHOICE S OR $\ldots$ OR T END	$S \, [] \, \ldots \, [] \, T$
VAR x IN S END	$@ x \,.\, S$
ANY x WHERE P THEN S END	$@ x \,.(P \Longrightarrow S)$
$x :\in E$	ANY z WHERE $z \in E$ THEN $x := z$ END, if $z \backslash E$

Syntactic Extensions of Elementary Substitutions

Syntax	Definition
SELECT P THEN S	CHOICE $P \Longrightarrow S$
WHEN Q THEN $T \ldots$	OR $Q \Longrightarrow T \ldots$
WHEN R THEN U	OR $R \Longrightarrow U$
END	END
SELECT P THEN $S \ldots$	SELECT P THEN $S \ldots$
WHEN Q THEN T	WHEN Q THEN T
ELSE U	WHEN $\neg(P \vee \ldots \vee Q)$ THEN U
END	END

Multiple Generalised Substitution

- It is a generalisation of the $||$ operator, introduced previously as a mere "syntactic sugar" for multiple simple substitutions
- It is the basic ingredient allowing to build large specifications
- Construct: $S||T$, read "S with T", where S and T are generalised substitutions supposed to work on two abstract machines M and N, working with the respective distinct variables x and y.
- There is no rule for calculating $[S||T]P$ from $[S]P$ and $[T]P$
- When occuring in proof obligations, the generalised substitution must be reduced to a form in which the parallel operator has been removed
- There are reduction rules (equivalences) used to move a parallel operator inside choices and conditionals, until reaching a point where it is only applied on simple assignments, which can be rewritten to remove it completely

Multiple Generalised Substitution (cont.)

- Basic reduction rules

$x := E \,\|\|\, y := F$	$= x, y := E, F$
$S \,\|\|\, T$	$= T \,\|\|\, S$
$S \,\|\|\, skip$	$= S$
$S \,\|\|\, (P \mid T)$	$= P \mid (S \,\|\|\, T)$
$S \,\|\|\, (P \Longrightarrow T)$	$= P \Longrightarrow (S \,\|\|\, T)$
$S \,\|\|\, (T \,[]\, U)$	$= (S \,\|\|\, T) \,[]\, (S \,\|\|\, U)$
$S \,\|\|\, (@z.T)$	$= @z.(S \,\|\|\, T)$, if $z \backslash S$

- Similar rules apply when elementary substitutions are replaced by their syntactic extensions

B Mathematical Notation - Relations

- AMN notation allows us to declare variables and constants having as type a relation type
- A *relation* R between sets S and T is a member of the powerset $\mathbb{P}(S \times T)$, i.e. a set of pairs (s, t), where $s \in S$ and $t \in T$, representing those elements that are related
- Shorthand notation: $S \leftrightarrow T \equiv \mathbb{P}(S \times T)$
- The *domain* of a relation $R \in S \leftrightarrow T$ is the set of elements in S that are related to something in T by means of R
  - $dom(R) = \{x \,|\, x \in S \wedge \exists y \cdot (y \in T \wedge (x, y) \in R)\}$
- The *range* of a relation $R \in S \leftrightarrow T$ is the set of elements in T that are related to something in S by means of R
  - $ran(R) = \{y \,|\, y \in T \wedge \exists x \cdot (x \in S \wedge (x, y) \in R)\}$

Operations on Relations

- $S <\!| R$ - restriction of R by S, aka *domain restriction*: retain only those pairs from R whose first component is in S
- $R |\!> T$ - co-restriction of R by T, aka *range restriction*: retain only those pairs from R whose second component is in T
- $S <\!<\!| R$ - anti-restriction of R by S, aka *domain substraction*: retain only those pairs from R whose first component is not in S
- $R |\!>\!> T$ - anti-co-restriction of R by T, aka *range substraction*: retain only those pairs from R whose second component is not in T
- $R_1 <\!+ R_2$ - *relational overriding*: updating of $R1$ according to R_2
- $\sim R$ - *inverse* of R: $\{(t, s) | (s, t) \in R\}$
- $R[U]$ - *relational image* of U by R: the set of all elements of T related by R to some element of U
- $R_1; R_2$ - *composition* of relations $R_1 : S \leftrightarrow T$ and $R_2 : T \leftrightarrow U$: $\{(s, u) \,|\, \exists t \in T \cdot (s, t) \in R_1 \wedge (t, u) \in R_2\}$

Functions

- A *function* is a special type of relation, in which an element of the source set can be related to at most one element of the target set
- If f is a function then $f(x)$ is the result of applying f on the argument x
- Concepts and operations discussed for relations apply to functions as well (e.g. domain or range)
- *Total* functions are functions whose domain is the entire source set
- *Partial* functions are functions whose domain is a subset of the source set
- *Injective* (one-to-one) functions are functions in case of which for any y from the range there is at most one x from their domain such that $y = f(x)$
- *Surjective* functions are functions in case of which the range is the entire target set

Function Types

- Let f be a function from source set X to target set Y. Then f is:
  - *total* (notation $-->$) if $dom(f) = X$, $ran(f) \subseteq Y$
  - *partial* (notation $+->$) if $dom(f) \subseteq X$, $ran(f) \subseteq Y$
  - *total injection* (notation $>->$) if $dom(f) = X$, $ran(f) \subseteq Y$ and one-to-one function
  - *partial injection* (notation $>+>$) if $dom(f) \subseteq X$, $ran(f) \subseteq Y$ and one-to-one function
  - *total surjection* (notation $-->>$) if $dom(f) = X$, $ran(f) = Y$
  - *partial surjection* (notation $+->>$) if $dom(f) \subseteq X$, $ran(f) = Y$
  - *bijection* (notation $>->>$) if $dom(f) = X$, $ran(f) = Y$ and one-to-one function

Sequences

- *Sequences* are ordered finite lists of elements of a given type
  - An element may appear more than once
- A sequence over S is a total function from $1..n \to S$ for some $n \in \mathbb{N}$
  - Operations applicable on sets, relations, functions also apply to sequences
- Notations
  - $seq(S)$ - the set of finite sequences with elements from S
  - $seq1(S)$ - the set of finite non-empty sequences with elements from S
  - $iseq(S)$ - the set of injective sequences with elements from S (no repetitions)
  - $perm(S)$ - permutations of elements from a finite set S
  - $[e_1, e_2, \ldots, e_n]$ - sequence with elements $e_1, e_2, \ldots, e_n$, the same as $\{(1, e_1), (2, e_2), \ldots, (n, e_n)\}$
  - $[]$ - empty sequence

Operations on Sequences

- $size(s)$ - size of sequence s
- $rev(s)$ - the reverse of s
- $first(s)$ - the first element of s, if s is non-empty
- $last(s)$ - the last element of s, if s is non-empty
- $tail(s)$ - s with the first element removed (if non-empty)
- $front(s)$ - s with the last element removed (if non-empty)
- $s/|\backslash n$ - s with the first n elements retained (head restriction, if $size(s) \geq n$)
- $s\backslash|/n$ - s with the first n elements removed (tail restriction)
- $s\hat{}t$ - concatenation of sequences s and t
- $e -> s$ - the sequence obtained by prepending e to s (head insertion)
- $s <- e$ - the sequence obtained by appending e to s (tail insertion)