Section: Master 1 SIGL

Faculty of Mathematics and Computer Science
Department of Computer Science

Formal Verification and Specification
Lab Session (TP) 02

1. **Ticket Dispenser:**
   Introduce In Method B the ticket dispenser in a queuing system. You have to define two operations *take_ Ticket* and *serve*. In atleir B, use the Proof System to check the consistency of the machine.

2. Suppose that we have the following problem: The computer science department has a number of available *Datashows*, and each one has possible *options* such as HDMI, VGA, ... A teacher may perform three main operations of the requested DataShow:

   - **Add:** add an option to the requested datashow
   - **Remove:** Remove an option from the requested datashow
   - **HasOption:** Asks whether the requested datashow is allowed to include a specific option

   We consider the following:

   - Three main sets: *Datashows*, *OPTION* and *Allowed* =OK, KO.
   - One relation *options* from *Datashows* to *OPTION*.

   Where the relation is denoted by $< - >$ and a pair or a Maplet is denoted by $|- >$.

3. **Dictionnary:**
   Implement in Atelier B a dictionary of words, where we define a total function between each two words. The dictionary defines a maximum number of words $maxMots$, which is of type $MaxInt$. In this Machine we can define the following operations:

   - *AddWord:* Add a word to the dictionary
   - *RetrieveWord:* Delete a word from the dictionary
   - *ExistWord:* Find a word

4. **SIGL students:**
   Define the abstract machine SIGL_Student as follows:

   - Define two variables SIGL_Student and year, where year refers to the level (1 or 2)
   - A SIGL_student must be a student
   - A year is a total function that maps to each student in SIGL_Student a year
   - A year is a natural (1 or 2)
   - As operations we have the following operations:
     - *initialization:* which initializes the defined variables
     - *inscription:* which allows the addition of a new student to SIGL_Students (NB: the year is affected automatically)
     - *graduate:* the graduation function means that a student is no longer in SIGL_Students
     - *get_ year:* which returns the current year for a given student

– *admis:* the admission function, which allows a student to pass from a year to another, where the maximum value is 2

<table>
<tr><th>Désignation</th><th>Notation</th><th>ASCII</th></tr>
<tr><td>Uniion</td><td>E U F</td><td>E \/F</td></tr>
<tr><td>Total fonction</td><td>S → T</td><td>S --> T</td></tr>
<tr><td>appartenance</td><td>$x \in E$</td><td>$x: E$</td></tr>
<tr><td>Difference</td><td>E/F</td><td>$E - F$</td></tr>
<tr><td>Restriction codomaine</td><td></td><td>r |> T</td></tr>
<tr><td>Anti-restriction domaine</td><td></td><td>S <<| r</td></tr>
<tr><td>For all</td><td>∀</td><td>!</td></tr>
</table>

Figure 1: Symbols to use for describing the machine

Answers

1. **Ticket Dispenser:**
   Introduce In Method B the ticket dispenser in a queuing system. You have to define two operations *take_ Ticket* and *serve*. In atleir B, use the Proof System to check the consistency of the machine.

```
    MACHINE Ticket

    VARIABLES
        serve, next

    INVARIANT
3/3     serve : NATURAL & next : NATURAL
3/3         & serve <= next

    INITIALISATION
2/2     serve, next := 0,0
```

```
    OPERATIONS

    ss <-- serve_next =
        PRE serve < next
2/2     THEN ss, serve := serve + 1, serve + 1
        END;

    tt <-- take_ticket =
2/2     tt, next := next, next+1

    END
```

Figure 2: Abstract machine for Ticket Dispenser

Answer:

2. Suppose that we have the following problem: The computer science department has a number of available *Datashows*, and each one has possible *options* such as HDMI, VGA, ... A teacher may perform three main operations of the requested DataShow:

   - **Add:** add an option to the requested datashow
   - **Remove:** Remove an option from the requested datashow
   - **HasOption:** Asks whether the requested datashow is allowed to include a specific option

We consider the following:

- Three main sets: *Datashows*, *OPTION* and *Allowed* =OK, KO.
- One relation *options* from *Datashows* to *OPTION*.

Where the relation is denoted by $< - >$ and a pair or a Maplet is denoted by $|- >$.

```
MACHINE
    Use_dataShow
SETS
    DataShows;
    OPTION;
    Allowed = {ok, ko}
VARIABLES
    options
INVARIANT
    options : DataShows <-> OPTION
INITIALISATION    options := {}

OPERATIONS
```

```
OPERATIONS
  add(ds,oo) =
  PRE ds : DataShows & oo:OPTION
  THEN options := options \/ {ds |-> oo}
  END;
  remove(ds,oo) =
  PRE ds : DataShows & oo:OPTION
  THEN options := options - {ds |-> oo}
  END;
  ans <-- hasOption(ds,oo) =
  PRE ds : DataShows & oo:OPTION
  THEN IF ds |-> oo : options
       THEN ans := ok
       ELSE ans := ko
       END
  END
END
```

Answer:

3. **Dictionnary:**
Implement in Atelier B a dictionary of words, where we define a total function between each two words. The dictionary defines a maximum number of words $maxMots$, which is of type $MaxInt$. In this Machine we can define the following operations:

   - *AddWord:* Add a word to the dictionary
   - *RetrieveWord:* Delete a word from the dictionary
   - *ExistWord:* Find a word

```
MACHINE dicoMot
SETS MOT /* abstract set of words */
; SIGNIFIK = {s0,s1,s2}/* abstract set of significations */
; OKKO = {ok, ko} /* a word used or not */
CONSTANTS maxMots /* limit */
PROPERTIES maxMots : 1..MAXINT
VARIABLES mots /* subset of words */
, dico /* the dictionary */
INVARIANT mots <: MOT /* subset of used words */
 & card(mots) <= maxMots & dico : mots --> SIGNIFIK
 INITIALISATION mots := {} ||
 dico := {} /* : mots --> SIGNIFIK */
 OPERATIONS
 ajoutMot(mm, signif) =
 PRE mm : MOT & mm /: mots & signif : SIGNIFIK & (mm,signif) /:
 dico
 & card(mots) < maxMots
 THEN
 mots := mots \/ {mm} || dico(mm) := signif
 END ;
 RetraitMot(mm) =
 PRE mm : MOT & mm : dom(dico) & card(mots) > 1
 THEN
 mots := mots - {mm} || dico := {mm} <<| dico
 END ;
 bb <-- existeMot(mm) =
 PRE mm : MOT
 THEN
 bb := bool(mm : dom(dico))
 END ;
 res <-- rechercheSignifMot(mm) = /* find the signification
 of a word */
 PRE mm : MOT & mm : dom(dico)
 THEN
 res := dico(mm)
 END
 END
```

Figure 3: Abstract machine for SIGL_Student

```
IMPLEMENTATION dicoMot_i REFINES dicoMot
DEFINITIONS PLAGE_MOT == 0..20 /* a range for implimenting the set of words */
VALUES MOT = PLAGE_MOT /* iplmimentation of abstract MOT */
; maxMots = 22 /* some value */
CONCRETE_VARIABLES c_mots, /* new variables */
c_dico
INVARIANT c_mots : PLAGE_MOT --> OKKO /* used or not */
  & mots = c_mots~[{ok}] /*  link abstract/concret */
  & c_dico : PLAGE_MOT --> SIGNIFIK
  & dico = (mots <| c_dico) /* link abstract concret */
INITIALISATION c_mots := (PLAGE_MOT)*{ko}; /* no word is already used */
c_dico := (PLAGE_MOT)*{s0} /* empty */
OPERATIONS
ajoutMot ( mm , signif ) =
BEGIN
c_mots(mm) := ok ; /* mots := mots \/ { mm } */
c_dico (mm) := signif
END ;
RetraitMot ( mm ) =
BEGIN
c_mots(mm) := ko /* mots := mots - { mm } */
END ;
bb <-- existeMot ( mm ) =
BEGIN /* bb := bool ( mm : dom (dico) )*/
VAR okko IN
okko := c_mots(mm);
IF okko = ok THEN
bb := TRUE
ELSE
bb := FALSE
END
END
END ;
res <-- rechercheSignifMot ( mm ) =
BEGIN
res := c_dico(mm) /* res := dico ( mm ) */
END
END
```

Figure 4: Abstract machine for SIGL_Student

Answer:

4. **SIGL students:**
   Define the abstract machine SIGL_Student as follows:

   - Define two variables SIGL_Student and year, where year refers to the level (1 or 2)
   - A SIGL_student must be a student
   - A year is a total function that maps to each student in SIGL_Student a year
   - A year is a natural (1 or 2)
   - As operations we have the following operations:
     - *initialization:* which initializes the defined variables
     - *inscription:* which allows the addition of a new student to SIGL_Students (NB: the year is affected automatically)
     - *graduate:* the graduation function means that a student is no longer in SIGL_Students
     - *get_year:* which returns the current year for a given student
     - *admis:* the admission function, which allows a student to pass from a year to another, where the maximum value is 2

Désignation	Notation	ASCII
Uniion	E U F	E $\backslash/F$
Total fonction	S → T	S --> T
appartenance	$x \in E$	$x: E$
Difference	E/F	$E - F$
Restriction codomaine		r |> T
Anti-restriction domaine		S <<| r
For all	∀	!

Figure 5: Symbols to use for describing the machine

```
MACHINE
    SIGL_Student (STUDENT)
VARIABLES
    SIGL_students, year
INVARIANT
    SIGL_students <: STUDENT &
    year : SIGL_students --> NATURAL &
    card(year) = 2
INITIALISATION
    SIGL_students := {} ||
    year := {}
OPERATIONS
    inscription (ss) =
        PRE
            ss : STUDENT - SIGL_students
        THEN
            SIGL_students := SIGL_students \/ {ss} ||
            year := year \/ {ss |-> 0}
        END ;

    graduate (ss) =
        PRE
            ss : SIGL_students
        THEN
            SIGL_students := SIGL_students - {ss} ||
            year := {ss} <<| year
        END ;

    yy <-- get_year (ss) =
        PRE
            ss : SIGL_students
        THEN
            yy := year(ss)
        END ;
  admis =
      ANY
          new_year
      WHERE
          new_year : SIGL_students --> NATURAL &
          !ss . (ss : SIGL_students => (new_year(ss) = year(ss) + 1) &(year(ss)<=1) )
      THEN
          year := new_year
      END
END
```

Figure 6: Abstract machine for SIGL_Student

Answer: