CS-M43
Algebraic Specification of Software and Hardware
(Answer 2 questions out of 3)

Question 1

a) Give the definitions of

i) a signature Σ.

ii) a Σ-algebra A.

[8 marks]

b) `byte, int, long` are among Java's integral types. Consider the following (slighlty shortened) description of operations on integral types in Java:

```
The Java programming language provides a number of operators
that act on integral values:
  * The comparison operators, which result in a value of
    type boolean:
    o The numerical comparison operators <, <=, >, and >=
    o The numerical equality operators == and !=
  * The numerical operators, which result in a value of
    type int or long:
    o The unary plus and minus operators + and -
    o The multiplicative operators *, /, %
```

(From J.Gosling, B.Joy, G.Steele, G.Bracha: The Java Language Specification)

Write a CASL specification that makes the above described signature precise for the integral type `byte`. Give short explanations for your choice of partial or total function symbols.

Hint: It might be useful to consider the ranges of the different integral types as defined below.

```
The values of the integral types are integers in the following ranges:
  * For byte, from -128 to 127, inclusive
  * For short, from -32768 to 32767, inclusive
  * For int, from -2147483648 to 2147483647, inclusive
  * For long, from -9223372036854775808 to 9223372036854775807, inclusive
  * For char, from '\u0000' to '\uffff' inclusive, that is, from 0 to 65535
```

[10 marks]

Question 1 continues on the next page

c) Specify a XOR-gate in CASL.

A XOR-gate is a hardware element, that takes two incoming bit streams A and B and produces one outgoing bit stream O, where O is 'true' if and only if A and B were different one clock signal earlier.

You may use the following specification fragment in order to import the specifications of the natural numbers and the specification of the booleans from the CASL standard library:

```
spec XOR =
  Nat
then
  Boolean
then
  ...
```

Specification fragments of `Nat` and `Boolean` can be found at the end of this paper.

[7 marks]

Question 2

a) i) What is a specification?

ii) What does it mean for a specification to be precise?

iii) Give an example of an inconsistent specification in natural language. Explain the inconsistency.

[2 + 2 + 4 = 8 marks]

b) Consider the following loose specification of a file system in CASL:

library FILESYSTEM **version** 1.0

spec DATA =
  sorts *File*, *Attribute*, *FileAndAttribute*
  ops *first* : *FileAndAttribute* → *File*;
    snd : *FileAndAttribute* → *Attribute*;
    combine : *File* × *Attribute* → *FileAndAttribute*;
end

spec STATE = DATA
then
  sort *State*
  op *setAttr* : *State* × *FileAndAttribute* → *State*;
    getAttr : *State* × *File* →? *Attribute*;
    initial : *State*;
end

A *File* is something that can have attributes. An *Attribute* could, for instance, be the content of the file, or the name of the file, or information on access rights.*FileAndAttribute* combines a *File* with its *Attribute*s into one entity. *first* and *snd* are projections on the components, *combine* glues two components together into one entity.

i) Give three algebras A, B, and C out of the model-class of the specification DATA, where

- A shall be an intended model,
- B shall be a model with confusion in the carrier set of *FileAndAttribute*, and
- C shall be a model with junk in the carrier set of *FileAndAttribute*.

Hint: You can work with arbitrary carrier sets for *File* and *Attribute*s.

[9 marks]

Question 2 continues on the next page

ii) Express the properties

- 'no confusion' in the carrier of *FileAndAttribute*, and
- 'no junk' in the carrier of *FileAndAttribute*

by first-order formulae in CASL.

[4 marks]

A file system can store a *FileAndAttribute* in its *State*. From there, the attributes of a file can be retrieved by *getAttr*. This is a partial function, as one might ask for information on a non-existing file. The function *setAttr* allows to change the state of the file system by adding a file or changing the attributes of a file. There is also an *initialState*, in which no file is known to the file system.

iii) Formulate in CASL the following properties using the signature provided by the specification STATE:

- In the initial state the function *getAttr* is undefined.
- The function *getAttr* yields for a file the last attribute that was set for it.

[4 marks]

Question 3

a) Give a sketch of the Waterfall-model of software-development and indicate, what kind of languages are used in its different phases. **[4 marks]**

b) Let Σ be the signature extracted from the CASL specification NAT below. Find four Σ-equations, which hold in the Σ-algebra A but not in the Σ-algebra B.

spec NAT =
  sort *Nat*;
  ops *0* : *Nat*;
    suc : *Nat* → *Nat*;
    __ + __, __ * __ : *Nat* × *Nat* → *Nat*;
end

Algebra A : $A(\text{Nat}) = \mathbf{N}$; (natural numbers)
$A(0) = 0$;
$A(\text{suc})(\text{N}) = \text{N}+1$;
$A(\_\_+\_\_)(\text{M},\text{N}) = \text{M}+\text{N}$;
$A(\_\_*\_\_)(\text{M},\text{N}) = \text{M}*\text{N}$;

Algebra B : $B(\text{Nat}) = \mathbf{N}$;
$B(0) = 0$;
$B(\text{suc})(\text{N}) = \text{N}+1$;
$B(\_\_+\_\_)(\text{M},\text{N}) = \text{M}*\text{N}$;
$B(\_\_*\_\_)(\text{M},\text{N}) = \text{M}+\text{N}$; **[8 marks]**

c) Specify the Mergesort-Algorithm for natural numbers in CASL.

A short reminder of Mergesort: Mergesort is based on the divide-and-conquer paradigm. Here is the three step divide-and-conquer process for sorting a list L:

Divide: Divide the n element list L into two sublists L_1 and L_2 of $n/2$ (or $(n+1)/2$ and $(n-1)/2$, if n is odd) elements each.

Conquer: The two lists L_1 and L_2 are sorted by recursive calls to mergesort.

Combine: Merge the sorted version of the list L_1 and the sorted version of the list L_2.

You may use the following specification fragment in order to import the specifications of lists from the CASL standard library:

```
spec MergeSort =
  List[Nat fit sort Elem |-> Nat] with sort List[Nat] |-> List
then
  ...
```

Specification fragments of `Nat` and `Boolean` can be found at the end of this paper.

[13 marks]

Useful specifications from the CASL standard library

```
spec Nat =
  free type Nat ::= 0 | suc(pre:? Nat)
  preds   __ <= __, __ >= __,
          __ <  __, __ > __:   Nat * Nat;
          even, odd:           Nat
  ops   __! :  Nat -> Nat;
        __ + __, __ * __, __ ^ __,
        min, max, __ -!__ :         Nat * Nat ->  Nat;
        __ -?__, __ /? __,
        __ div __, __ mod __:       Nat * Nat ->? Nat;
  ...
end

spec Boolean =
  free type Boolean ::= True | False
  ops   Not__ :           Boolean -> Boolean;
        __And__, __Or__ : Boolean * Boolean -> Boolean
  forall x,y:Boolean
  . Not(False) = True                %(Not_False)%
  . Not(True) = False                %(Not_True)%
  . False And False = False          %(And_def1)%
  . False And True = False           %(And_def2)%
  . True  And False = False          %(And_def3)%
  . True  And True = True            %(And_def4)%
  . x Or y = Not(Not(x) And Not(y))  %(Or_def)%
end

spec GenerateList [sort Elem] =
  free type List[Elem] ::=  [] |  __ :: __ (first:? Elem; rest:? List[Elem])
end
spec List [sort Elem] given Nat = GenerateList[sort Elem] then
     pred  isEmpty: List[Elem];
           __eps__: Elem * List[Elem]
     ops   __ + __ : List[Elem] * Elem -> List[Elem];
           first, last: List[Elem] ->? Elem;
           front, rest: List[Elem] ->? List[Elem];
           #__: List[Elem] -> Nat;
           __ ++ __ : List[Elem] * List[Elem] -> List[Elem];
           reverse:     List[Elem] -> List[Elem];
           __ ! __:     List[Elem] * Nat ->? Elem;
           take,drop:   Nat * List[Elem] ->? List[Elem];
           freq: List[Elem] * Elem -> Nat
     ...
end
```