CS_376 PROGRAMMING WITH ABSTRACT DATA TYPES

(Attempt 2 questions out of 3)

Question 1

(a) Let Σ be the signature with one sort, one constant, 0, and one binary operation, $+$. Let A be the Σ-algebra of natural numbers, $\{0, 1, 2, 3, \ldots\}$, where the constant 0 is interpreted by the number 0 and the operation $+$ is interpreted by addition.

For each of the following formulas decide whether it is logically valid and whether it holds in the Σ-algebra A. Justify your answers.

(i) $\forall x \, \forall y \, (x + y = y + x)$

(ii) $\forall x \, (\forall y \, (x + y = y) \vee \forall y \, (x + y \neq y))$

(iii) $\forall x \, (\forall y \, (x + y = y) \vee \exists y \, (x + y \neq y))$

[6 marks]

(b) Let Σ be a signature. Let Δ be the set of all Σ-formulas that are either of the form $t = s$ (t, s arbitrary Σ-terms of the same sort), or of the form $P_1 \vee P_2$, or of the form $\exists x \, P$ (P_1, P_2, P arbitrary Σ-formulas). Set

$$\Gamma := \{\neg\neg P \to P \mid P \in \Delta\}$$

Show that

$$\Gamma \vdash_{\mathrm{m}} \neg\neg P \to P$$

for *all* Σ-formulas P, by induction on Σ-formulas.

[9 marks]

(c) **(i)** Produce an initial specification for the algebra of *double ended queues of natural numbers* (*deques*, for short) that has a sort deque, a constant nil for the empty queue and the following constants and operations:

- cons: adds a number at the front of a deque;
- snoc: adds a number at the end of a deque;
- head: computes the front element of a nonempty deque (if the deque is empty the number 0 shall be returned);
- last: computes the element at the end of a nonempty deque (if the deque is empty the number 0 shall be returned);

You may import a suitable data type of natural numbers.

(ii) Determine a minimal set of generators in the specification you have given in (i) above. Are these generators free?

[10 marks]

Question 2

(a) Let Σ be the signature with one sort, two constants, 0 and 1, and two binary operations, $+$ and $*$. Let $\mathbf{R}$ respectively $\mathbf{C}$ be the Σ-algebras of real respectively complex numbers with 0, 1, addition and multiplication. Show that $\mathbf{R}$ and $\mathbf{C}$ are not isomorphic.

[6 marks]

(b) Suppose an initial specification of an abstract data type of finite sets (of elements of an unspecified sort element of elements) with a constant for the empty set and operations for inserting an element into a set and selecting an element from a set is given. Suppose the specification contains the following equations:

$$\begin{aligned} \mathsf{insert}(\mathsf{insert}(s,x),x) &= \mathsf{insert}(s,x) \\ \mathsf{insert}(\mathsf{insert}(s,x),y) &= \mathsf{insert}(\mathsf{insert}(s,y),x) \\ \mathsf{select}(\mathsf{emptyset}) &= \mathsf{error} \\ \mathsf{select}(\mathsf{insert}(s,x)) &= x \end{aligned}$$

Explain why this specification is flawed. **[6 marks]**

(c) Consider the following Haskell implementation of an abstract data type of finite binary trees.

```
module Tree (Tree,leaf,branch,isleaf,root,left,right) where

data Tree  = Leaf Int | Branch (Tree Int) Int (Tree Int)

leaf :: Int -> Tree
leaf x = Leaf x

branch :: Tree -> Int -> Tree -> Tree
branch s x t = Branch s x t

root :: Tree -> Int
root (Leaf x)       = x
root (Branch s x t) = x

left   :: Tree -> Tree
left (Leaf x)       = error "left of Leaf"
left (Branch s x t) = s

right  :: Tree -> Tree
right (Leaf x)       = error "right of Leaf"
right (Branch s x t) = t
```

(i) Extend the implementation above by an observer sumLabels that computes the sum of all labels occurring in a tree.

(ii) Give a different implementation of the same abstract data type that allows for a more efficient implementation of the operation sumLabels.

(iii) Why is it important that the Haskell constructors for trees, Leaf and Branch, are not exported directly, but only via their synonyms, leaf and branch?

[13 marks]

Question 3

(a) What does it mean for an implementation of an abstract data type to be *persistent*? Explain, by means of an example, why imperative implementations of abstract data types may fail to be persistent.

[7 marks]

(b) Explain, what it means for a term rewriting system to be

(i) terminating,

(ii) locally confluent,

(iii) confluent.

[6 marks]

(c) Consider the signature $\Sigma := (\{s\}, \{1\colon s,\ *\colon s \times s \to s\})$ and the following three term rewriting systems:

$$R_1 := \{1 * x \mapsto x\ ,\ (x * y) * z \mapsto x * (y * z)\},$$
$$R_2 := \{x * 1 \mapsto x\ ,\ (x * y) * z \mapsto x * (y * z)\},$$
$$R_3 := \{1 * x \mapsto x\ , x * 1 \mapsto x\ ,\ (x * y) * z \mapsto x * (y * z)\}$$

(i) Prove that R_3 is terminating.

(ii) Decide for each of the term rewriting systems, R_1, R_2, R_3, whether it is confluent. Justify your answers.

[12 marks]