Prof. Dr. Hans-Jörg Kreowski, Renate Klempien-Hinrichs
Studiengang Informatik
MZH 3260, Tel.: 2956, 3697 (Sekr.), Fax: 4322
E-Mail: {kreo, rena}@informatik.uni-bremen.de

20. November 2001
Aufgaben 2

Algebraische Spezifikation

2. Übungsblatt

Geübt werden Gleichungen, ihre Gültigkeit und der Zusammenhang zwischen Spezifikationen und zugehörigen Algebren (vgl. die Definitionen vom Ergänzungsblatt und 4.1, 4.5 im Skript).

1. Betrachte die folgenden Gleichungen zur Signatur SET vom 1. Übungsblatt:

$$insert(x, insert(x, S)) = insert(x, S) \quad \text{(i)}$$
$$insert(x, insert(y, S)) = insert(y, insert(x, S)) \quad \text{(ii)}$$
$$union(\emptyset, S') = S' \quad \text{(iii)}$$
$$union(insert(x, S), S') = insert(x, union(S, S')) \quad \text{(iv)}$$

   (a) Zeige, dass diese Gleichungen in der SET-Algebra $2^A = (A, 2^A, a_1, \ldots, a_n, \emptyset, \mathsf{ins}, \cup)$ gelten.

   (b) Welche der Gleichungen gelten in der SET-Algebra $A^* = (A, A^*, a_1, \ldots, a_n, \lambda, \mathsf{insert}, \mathsf{concat})$? Gib für jede Gleichung, die nicht gilt, eine Wertzuweisung als Gegenbeispiel an.

   Für die Bearbeitung der Aufgabe dürfen die Ergebnisse des 1. Übungsblatts verwendet werden.

2. Unbeschränkte Felder mit ganzen Zahlen als Indizes und Zeichenketten als Einträge können formalisiert werden als Abbildungen $arr\colon \mathbb{Z} \to A^*$ mit endlichem Träger, d.h. $support(arr) = \{x \in \mathbb{Z} \mid arr(x) \neq \lambda\}$ ist eine endliche Menge. Typische Operationen auf Feldern sind assign, das einem Feld arr, einem Index x und einem Eintrag w das Feld arr' zuordnet mit $arr'(x) = w$ und $arr'(y) = arr(y)$ sonst, und read, das für ein Feld arr und einen Index x den Eintrag $arr(x)$ liefert. Eine typische Konstante ist das initiale Feld init mit $\mathsf{init}(x) = \lambda$ für alle Indizes x. Das definiert einen Datentyp zu der folgenden Spezifikation.

spec ARRAY = INT + STRING
sorts *Array*
opns *init*: $\to$ *Array*
assign: *Array* $\times$ *Int* $\times$ *String* $\to$ *Array*
read: *Array* $\times$ *Int* $\to$ *String*
cond: *Bool* $\times$ *Array* $\times$ *Array* $\to$ *Array*
eqns *cond*(*true*, *A*, *A'*) = *A*
cond(*false*, *A*, *A'*) = *A'*

   Dabei ist *cond* gerade das allseits bekannte *if-then-else*-Konstrukt (für Felder). Dessen erstes Argument hat die Sorte *Bool*. Sie ist wie die beiden Booleschen Konstanten *true* und *false* mit INT importiert. In INT ist außerdem ein Gleichheitstest der Form *eq*: *Int* $\times$ *Int* $\to$ *Bool* verfügbar.

   Erweitere die Spezifikation ARRAY um Gleichungen, die möglichst genau den eingangs beschriebenen Datentyp fassen.

3. Betrachte die folgende Spezifikation der natürlichen Zahlen.

spec NAT1 =
sorts *Nat*
opns 0: $\to$ *Nat*
succ: *Nat* $\to$ *Nat*
+: *Nat* $\times$ *Nat* $\to$ *Nat*
eqns +(0, *n*) = *n*
+(*succ*(*m*), *n*) = *succ*(+(*m*, *n*))

   (a) Zeige, dass die Gleichung $succ(n) = +(succ(0), n)$ in jeder NAT1-Algebra gilt.

   (b) Die Gleichung $succ(n) = +(n, succ(0))$ gilt *nicht* in jeder NAT1-Algebra. Konstruiere eine NAT1-Algebra (mit möglichst kleiner Datenmenge für *Nat*), in der diese Gleichung nicht gilt.

Abgabe bis zum 4. Dezember 2001.