
Demo-Klausur Informatik I
(Fassung vom 14.3.2001, 14:00)

Die Bearbeitungszeit für die Klausur beträgt 60 Minuten; es gibt insgesamt 61 Punkte. Als Hilfsmittel ist lediglich der *Revised⁵ Report on the Algorithmic Language Scheme* zugelassen.

Wir haben uns bemüht, die Aufgaben so zu stellen, daß sie in Themenbreite und Schwierigkeitsgrad denen der richtigen Klausur so gut wie möglich entsprechen. Das bedeutet insbesondere nicht, daß in der richtigen Klausur nur Themen vorkommen, die auch in der Demo-Klausur vorkommen. *Für die Ähnlichkeit dieser Klausur mit der richtigen Klausur gibt es somit keine Gewähr.*

1. [8 Punkte] Gegeben sei der folgende abstrakte Datentyp über dem Grundtyp A:

```

datatype Folge(A)
sorts Folge, A
constructors
  empty: () → Folge
  cons: A × Folge → Folge
operations append: Folge × Folge → Folge
  reverse: Folge × Folge → Folge
equations
  append(empty, R) = R
  append(cons(a, F), R) = cons(a, append(F, R))
  reverse(empty, R) = R
  reverse(cons(a, F), R) = reverse(F, cons(a, R))
end

```

Beweisen Sie, daß für alle $L_1, L_2, L_3 \in \text{Folge}$ gilt

$$\text{append}(\text{reverse}(L_1, L_2), L_3) = \text{reverse}(L_1, \text{append}(L_2, L_3))$$

2. [10 Punkte] Betrachten Sie folgende EBNF-Definition:

```

⟨formula⟩ ::= ⟨atomic formula⟩
           | not(⟨formula⟩)
           | (⟨formula⟩ and ⟨formula⟩)
           | (∀ ⟨var⟩ ⟨rest arg⟩* . ⟨formula⟩ )
⟨rest arg⟩ ::= , ⟨var⟩
⟨atomic formula⟩ ::= ⟨predicate⟩ ⟨arguments⟩?
⟨term⟩ ::= ⟨var⟩ | ⟨fun⟩ ⟨arguments⟩?
⟨arguments⟩ ::= (⟨term⟩ ⟨rest term⟩*)
⟨rest term⟩ ::= , ⟨term⟩
⟨var⟩ ::= x | y | z
⟨fun⟩ ::= f | g | h
⟨predicate⟩ ::= p | q | r

```

- (a) Leiten Sie aus dem Startsymbol ⟨formula⟩ das folgende Wort ab:

$((\forall x, y. (r(x,y) \text{ and not}(p(x)))) \text{ and } q(x,z))$

- (b) Formen Sie die EBNF in eine BNF-Definition um, indem Sie die EBNF-Konstrukte eliminieren.
3. [6 Punkte] Programmieren Sie eine Datenrepräsentation für Matrizen mit Hilfe von Listen! Eine Zeile der Matrix ist dabei eine Liste von Zahlen, die Matrix selbst eine Liste von Zeilen.
- (a) Geben Sie einen Konstruktor `make-matrix` an, der die Zeilen der Matrix als Argument nimmt. Der Konstruktor muß nicht überprüfen, daß die Zeilen alle die gleiche Länge haben.
- (b) Geben Sie eine Prozedur `matrix-scalar-multiply` an, die eine Matrix mit einem Skalar multipliziert
- (c) Geben Sie eine Prozedur `matrix-multiply` an, die eine $M \times N$ -Matrix mit einer $N \times K$ -Matrix multipliziert.
4. [4 Punkte] Was ist der nächste Reduktionsschritt für den λ -Term

$(\lambda x.y (\lambda y.x)) ((\lambda z.(\lambda x.z x)) (\lambda x.x))$

- (a) ... nach der Call-by-Value-Reduktion?
- (b) ... nach der Call-by-Name-Reduktion?
5. [6 Punkte] Was ist die Definition des Begriffs *Liste* in Scheme?
Sind die Werte folgender Scheme-Ausdrücke Listen? Begründen Sie Ihre Antworten!

- (a) `(cons 3 4)`
- (b) `(cons (list) (list))`
- (c) `(cons 2 (cons '() (cons 1 '())))`
- (d) `(list 1)`
- (e) `(cdr (cons '() (cons 1 (list 2 3))))`

6. [3 Punkte] Formulieren Sie das Prinzip der *strukturellen Induktion*!
7. [6 Punkte] Betrachten Sie die folgenden λ -Terme:

$(\lambda x.\lambda y.y) ((\lambda z.x z) y)$

$x (\lambda x.x) ((\lambda y.x.y) y)$

$\lambda xyz.x z (y z)$

- (a) Unterstreichen Sie in den λ -Termen alle β -Redexe.
- (b) Führen Sie für jeden der Terme, der einen β -Redex enthält, jeweils eine β -Reduktion durch!
- (c) Welches sind jeweils die freien und gebundenen Variablen der Terme?
8. [7 Punkte] Betrachten Sie das folgende Scheme-Programm:

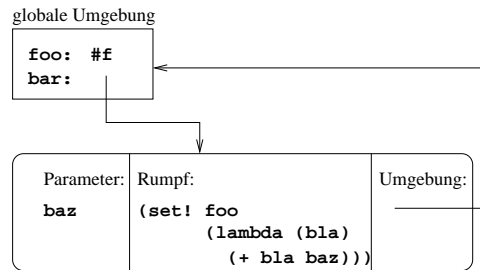
```

(define foo #f)

(define bar
  (lambda (baz)
    (set! foo
      (lambda (bla)
        (+ bla baz))))))

```

Die beiden Definition erzeugen folgendes Umgebungsdiagramm:



Ändern und ergänzen Sie das Diagramm so, daß es die Situation nach der Auswertung von `(bar 5)` reflektiert. Welchen Wert hat nach der Auswertung von `(bar 5)` der Ausdruck `(foo 7)` und warum?

9. [6 Punkte] Beweisen Sie, daß die folgende Scheme-Prozedur `euclid`, wenn sie auf zwei nichtnegative ganze Zahlen angewendet wird, ihren größten gemeinsamen Teiler berechnet:

```

(define euclid
  (lambda (m n)
    (if (= n 0)
        m
        (euclid n (remainder m n)))))

```

10. [5 Punkte] Übersetzen Sie folgende Aussage in \mathcal{L}_1 und konstruieren Sie dafür einen Beweisbaum in SC_1 :

Wenn es regnet und es nicht regnet, dann scheint die Sonne.