VII. Übungsblatt "Informatik I" WS 95/96

Christian Schulte, Prof. Gert Smolka

Ausgabe: 1. Dezember 1995 **Abgabe:** 8. Dezember 1995 vor der Vorlesung

Aufgabe VII.1 (Mengen von Symbolen (3)) Mengen von Symbolen lassen sich in Scheme als Listen von Symbolen darstellen, wobei jedes Element einer Menge nur einmal in der Liste vorkommen darf.

1. Binden Sie den Namen `menge-leer` an die leere Liste und definieren Sie die Prozedur `(menge-leer? m)`, die testet, ob die Menge `m` leer ist.
2. Definieren Sie eine Prozedur `(menge-adjunktion m s)`, die eine Menge zurückliefert, die alle Elemente der Menge `m` und das Symbol `s` enthält. Die Gleichheit von zwei Symbolen kann in Scheme mit `eq?` getestet werden.
3. Implementieren Sie eine Prozedur `menge-vereinigung`, die die Vereinigung zweier Mengen berechnet.
4. Implementieren Sie eine Prozedur `(menge-ohne m s)`, die die Menge berechnet, die alle Elemente von `m` ohne `s` enthält.

Aufgabe VII.2 (Arithmetische Ausdrücke (4)) Ein *arithmetischer Ausdruck* A ist definiert als:

A	::=	Z	*Zahl*
	|	V	*Variable*
	|	(+ A_1 A_2)	*Summe*
	|	(* A_1 A_2)	*Produkt*

Ein Ausdruck kann in Scheme zum Beispiel als `'(* (+ 1 x) (* 2 y))` eingegeben werden. Das heißt Zahlen sind Zahlen in Scheme (kann mit `number?` getestet werden), Variablen sind Symbole in Scheme (kann mit `symbol?` getestet werden), eine Summe ist eine dreielementige Liste, deren erstes Element das Symbol + ist, und ein Produkt ist eine dreielementige Liste, deren erstes Element das Symbol `*` ist. Die restlichen Elemente von Summen und Produkten sind gerade wieder arithmetische Ausdrücke. Im folgenden meinen wir mit Ausdrücken stets arithmetische Ausdrücke.

1. Definieren Sie Prozeduren `zahl?`, `variable?`, `summe?` und `produkt?`, die testen, ob ein Ausdruck eine Zahl, eine Variable, eine Summe oder ein Produkt ist.
2. Definieren Sie Selektoren `summe-1`, `summe-2`, `produkt-1` und `produkt-2`, die die jeweiligen Teilausdrücke der Ausdrücke zurückliefern. Zum Beispiel sollen `(summe-1 '(+ 1 2))` zu `1` und `(produkt-2 '(* 1 2))` zu `2` evaluieren.
3. Definieren Sie eine Prozedur `(ausdruck? a)`, die testet, ob ihr Argument ein Ausdruck ist.
4. Definieren Sie eine Prozedur `(variablen a)`, die zu dem arithmetischen Ausdruck `a` die Menge der darin auftretenden Variablen berechnet. Benutzen Sie dabei die Mengenoperationen aus Aufgabe VII.1.

Aufgabe VII.3 (Umgebungen (2)) Eine Umgebung ist eine Abbildung, die (endlich viele) Variablen auf Zahlen abbildet. Eine Umgebung kann in Scheme als Liste von Paaren dargestellt werden, wobei das erste Element eines Paares eine Variable, und das zweite Element eines Paares eine Zahl ist. Zusätzlich verlangen wir, daß jede Variable in einer Umgebung nur einmal vorkommt. Eine Umgebung, die `x` auf `4`, `y` auf `5` und `z` auf `42` abbildet, ist zum Beispiel: `((x . 4) (y . 5) (z . 42))`

1. Definieren Sie eine Prozedur `(umgebung-zahl u v)`, die die Zahl zurückliefert, auf die die Umgebung `u` die Variable `v` abbildet. So soll `(umgebung-zahl '((x . 1) (y . 2)) 'y)` zu `2` evaluieren.

2. Definieren Sie eine Prozedur `(umgebung-adjunktion u v z)`, die eine Umgebung berechnet, die `u` um das Paar `(v . z)` erweitert, falls `v` noch nicht in `u` vorkommt. Kommt `v` bereits in einem Paar in `u` vor, so soll eine Umgebung berechnet werden, in der dieses Paar durch `(v . z)` ersetzt ist. So soll `(umgebung-adjunktion '((x . 2)) 'y 3)` zu `((x . 2) (y . 3))` evaluieren, und `(umgebung-adjunktion '((x . 2)) 'x 3)` zu `((x . 3))` evaluieren.

Aufgabe VII.4 (Evaluieren arithmetischer Ausdrücke (3)) Schreiben Sie eine Prozedur `(eval a u)`, die den arithmetischen Ausdruck `a` in der Umgebung `u` evaluiert.

Aufgabe VII.5 (Semantik arithmetischer Ausdrücke (5)) Geben Sie eine Semantik für arithmetische Ausdrücke an (im Stil der Semantik für Scheme 1). Die Transitionen der Semantik sollen von der Form $A, U \longrightarrow Z$ sein, wobei Z die Zahl ist, zu der der arithmetische Ausdruck A in der Umgebung U evaluiert.
Hinweis: Sie brauchen vier Regeln für die Semantik. Die Regel für Zahlen ist: $Z, U \longrightarrow Z$.

Aufgabe VII.6 (Infix-Darstellung arithmetischer Ausdrücke (3)) Wir haben arithmetische Ausdrücke in der sogenannten *Präfix*-Darstellung definiert: Das Operationssymbol kommt vor den Argumenten. Die Ihnen vertrautere Darstellung aus der Mathematik ist die sogenannte *Infix*-Darstellung, wobei das Operationssymbol zwischen die beiden Argumente geschrieben wird. Für den Ausdruck `(* (+ x 3) (* (+ 4 y) z))` ist die Infix-Darstellung `((x + 3) * ((4 + y) * z))`. Schreiben Sie eine Prozedur `(praefix->infix a)`, die einen arithmetischen Ausdruck in die zugehörige Infix-Darstellung umwandelt.

Aufgabe VII.7 (Erweiterte arithmetische Ausdrücke (5)) Im folgenden wollen wir erweiterte arithmetische Ausdrücke betrachten, in denen auch ein Let-Ausdruck erlaubt ist:

$$A ::= \textit{wie in Aufgabe VII.2} \quad | \quad \texttt{(let (}V\ A_1\texttt{)}\ A_2\texttt{)}$$

Dabei nennen wir V den *Parameter*, A_1 das *Argument* und A_2 den *Rumpf* des Let-Ausdrucks. In Scheme wollen wir einen Let-Ausdruck als dreielementige Liste darstellen, wobei das erste Element das Symbol `let` ist, das zweite Element eine zweielementige Liste, die eine Variable und einen Ausdruck enthält, und das dritte Element wieder ein Ausdruck ist.

1. Schreiben Sie ein Prädikat `let?`, und Selektoren `let-parameter`, `let-argument` und `let-rumpf`.

2. Erweitern Sie die Prozedur `ausdruck?` aus Aufgabe VII.2 auf erweiterte Ausdrücke.

3. Geben Sie eine Prozedur `(freie-variablen a)` an, die die Menge der freien Variablen des erweiterten Ausdrucks `a` berechnet. Die freien Variablen F[] eines Let-Ausdrucks sind definiert als:
$$\mathrm{F}[\texttt{(let (}V\ A_1\texttt{)}\ A_2\texttt{)}] := \mathrm{F}[A_1] \cup (\mathrm{F}[A_2] - \{V\}) .$$

4. Erweitern Sie die Prozedur `eval` aus Aufgabe VII.4, so daß erweiterte arithmetische Ausdrücke evaluiert werden.