Hochschule München
Fakultät für Informatik und Mathematik
M. Ruckert

Theoretische Informatik II
Übungen

Aufgabe 22. Implementieren Sie Bindungen und Umgebungen für den Scheme Interpreter. Sowohl Bindungen als auch Umgebungen sind dabei Strukturen, die sich innerhalb des simulierten Speichers befinden.

Umgebungen sind partielle Abbildungen von Symbolen auf Ausdrücke. Eine solche partielle Abbildung ist ϵ, die leere Abbildung. Die nächst einfachere Abbildung, eine Bindung, ordnet einem einzelnen Symbol einen Wert zu. Alle anderen Abbildungen können aus diesen beiden einfachen Formen durch Komposition zusammengesetzt werden.

Bindungen werden nie verändert (nur der garbage collector sammelt sie wieder ein). Umgebungen verändern sich im Laufe der Ausführung eines Programms (z.B. durch `define`). Deshalb muss man Umgebungen und Bindungen unterscheiden.

- Bindungen.

  Genau genommen kann man auch eine einfache Bindungen als Komposition schreiben—als Komposition mit ϵ. Deshalb verwenden wir als Repräsentation im simulierten Speicher zum einen das Tag `'epsilon` und zum anderen Records mit Tag `'bind` gefolgt von der Variablen, (also dem Zeiger auf ein Symbol), gefolgt vom (Zeiger auf den) Wert, gefolgt vom (Zeiger auf die) Komposition. Mittels des Zeigers auf die Komposition werden so Listen von Bindungen gebildet, die mit ϵ enden.

  Definieren sie `i-epsilon` und schreiben Sie folgende Funktionen:

  - `(new-binding symbol value compose)`

    Diese Funktion erzeugt eine neue Bindung (Konstruktor). Die neue Bindung ordnet dem Symbol `symbol` den Wert `value` zu und verknüpft sie mit der Bindung (oder genauer Liste von Bindungen) `compose`.

  - `(binding->value symbol binding)`

    Diese Funktion wendet die gegebene Bindung (genauer Liste von Bindungen) auf das gegebene Symbol (Zeiger) an und liefert als Ergebnis den (Zeiger auf den) zugeordneten Wert (oder false `#f`). Die Funktion sucht dabei falls nötig die Liste aller verknüpften Bindungen durch bis ϵ erreicht ist.

- Umgebungen.

  Umgebungen sind Variablen in denen eine Liste von Bindungen gespeichert ist. Während des Programms nimmt man die Liste aus der Umgebung, erweitert sie durch Komposition und speichert sie wieder in der Umgebung ab. Umgebungen bestehen im Speicher aus einem Tag (`'environment`) gefolgt vom Zeiger auf eine Liste von Bindungen.

  In der globalen Variablen `i-environment` wird der Zeiger auf die globale Umgebung gespeichert. Sie wird mit `i-epsilon` initialisiert. Schreiben sie weiter folgende Funktionen:

  - `(new-environment binding)`
    Erzeugt im simulierten Speicher eine neue Umgebung (Konstruktor).
  - `(environment->binding env)`
    Extrahiert die Bindung aus einer Umgebung (Getter).
  - `(environment-set! env binding)`
    Überschreibt die Bindung einer Umgebung (Setter).
  - `(add-variable var value env)`
    Die in der Umgebung gespeicherte Liste wird um eine Bindung (eine Variable mit zugehörigem Wert) erweitert und die neue Liste wird wieder in der Umgebung gespeichert.
  - `(variable->value var env)`
    Dies ist die Funktion mit der man für eine Variable in einer Umgebung den Wert bestimmt. Nach dieser Vorbereitung sind das jetzt nur noch zwei Zeilen Code.