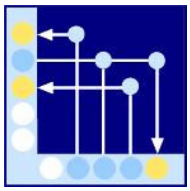
Hochschule Aalen
Fakultät Elektronik und Informatik
Studienbereich Informatik



Compilerbau-Praktikum

Lehrveranstaltung im Wintersemester 2022/2023
Prof. Dr. habil. Christian Heinlein

8. Aufgabenblatt (6. Dezember 2022)

Aufgabe 8: Benutzerdefinierte Operatoren

Teilaufgabe 8.a)

Erweitern Sie die bis jetzt entwickelte Minisprache wie folgt um benutzerdefinierte Operatoren:

- Die Deklaration eines benutzerdefinierten Operators hat die Form `sig -> (impl)`

  Die Signatur `sig` besteht aus einem oder mehreren Teilen, bei denen es sich entweder um einen Namen (wie bei Konstantendeklarationen) oder um eine Parameterdeklaration der Form `(names)` handelt, wobei `names` für einen oder mehrere Namen steht.

  Die Implementierung `impl` ist ein beliebiger Ausdruck, in dem die in der Signatur deklarierten Parameter als Konstanten zur Verfügung stehen. Außerdem kann der gerade deklarierte Operator bereits in seiner eigenen Implementierung verwendet werden.

- Damit sowohl die Parameter als auch der deklarierte Operator selbst in der Implementierung verfügbar sind, müssen sie durch eine passende Prolog- oder Epilogfunktion erzeugt und im Exportkatalog des Operatordeklarationsausdrucks gespeichert werden, bevor der Parser die Implementierung verarbeitet.
  Weil die Implementierung des Operators zu diesem Zeitpunkt noch nicht verfügbar ist, muss sie später in einer weiteren Prolog- oder Epilogfunktion hinzugefügt werden.

- Um unerwünschte Mehrdeutigkeiten zu reduzieren, soll die Nacheinanderausführung im vorderen und hinteren Operanden eines benutzerdefinierten Operators immer automatisch ausgeschlossen sein, sofern der Operator entsprechende Operanden besitzt.
  Anschließend müssen die Funktionen `setback` und `setfollow` des Parsers mit der Signatur des Operators aufgerufen werden.

- Außerdem muss der Parameter des Operatordeklarationsoperators, der die Implementierung `impl` bezeichnet, die Pseudo-Operatoren `All` und `Self` importieren. `Self` bewirkt hierbei, dass alle Operatoren aus dem Exportkatalog des Operatordeklarationsausdrucks importiert werden.

- Damit der Operatordeklarationsausdruck abschließend nur den deklarierten Operator und nicht auch seine Parameter exportiert, müssen diese in der zweiten o. g. Prolog- oder Epilogfunktion wieder aus dem Exportkatalog entfernt werden.

  Hinweis: Wenn man einen neuen Operatorkatalog `c` mit Hilfe der Funktion `cat` aus einer Sequenz von Operatoren `os` erzeugt, enthält `c(opers_)` die Operatoren anschließend in der gleichen Reihenfolge wie `os`.

- Das Laufzeitsystem benötigt zusätzlich folgende Attribute:

```
// Umschließender Kontext eines Kontexts (nil beim globalen Kontext).
ATTR1(encl_, Context, Context)

// Siehe unten.
ATTR1(oper_, Value, Oper)
ATTR1(encl_, Value, Context)
```

- Die Auswertungsfunktion des Konstantendeklarationsoperators speichert den Wert der Konstanten nicht mehr im globalen Kontext, sondern im jeweils aktuellen Kontext.
- Die Auswertungsfunktion eines Konstantenoperators sucht den Eintrag ihres Operators mit dem zugehörigen Wert zunächst im aktuellen Kontext und anschließend ggf. sukzessive in jedem umschließenden Kontext, bis der passende Eintrag gefunden wurde.
- Die Auswertungsfunktion des Operatordeklarationsoperators liefert einen synthetischen Wert, der im Attribut `oper` den deklarierten Operator und im Attribut `encl` den aktuellen Kontext enthält. Außerdem speichert sie den deklarierten Operator zusammen mit diesem Wert im aktuellen Kontext. (Momentan kann man mit diesem Resultatwert einer Operatordeklaration allerdings noch nichts Sinnvolles anfangen.)
- Die Auswertungsfunktion eines benutzerdefinierten Operators sucht den Eintrag ihres Operators mit dem zugehörigen Wert auf die gleiche Weise wie die Auswertungsfunktion eines Konstantenoperators. Dann erzeugt sie einen neuen Kontext, der als umschließenden Kontext den im Attribut `encl` des Werts gespeicherten Kontext besitzt.

  Anschließend werden alle Operanden des Ausdrucks (mit unverändertem aktuellem Kontext) von links nach rechts ausgewertet und ihre Werte zusammen mit den korrespondierenden Parametern des Operators in diesem neuen Kontext gespeichert.

  Schließlich wird die Implementierung des Operators mit dem neuen Kontext als aktuellem Kontext ausgewertet und der resultierende Wert zurückgegeben, nachdem der vorige aktuelle Kontext wiederhergestellt wurde.

Teilaufgabe 8.b)

Implementieren und testen Sie dann u. a. folgende benutzerdefinierte Operatoren:

- $|\bullet|$ liefert den Absolutbetrag seines Operanden.
- $\bullet^2$ liefert das Quadrat seines Operanden.
- `gcd of` • `and` • liefert den größten gemeinsamen Teiler seiner Operanden, der entweder iterativ oder rekursiv berechnet wird.
- `ack` • • liefert den Wert der Ackermann-Funktion, die für $n, m \in \mathbb{N}_0$ wie folgt definiert ist (vgl. Wikipedia):

$$\mathrm{ack}(n, m) = \begin{cases} m + 1 & \text{für } n = 0 \\ \mathrm{ack}(n - 1, 1) & \text{für } n > 0 \text{ und } m = 0 \\ \mathrm{ack}(n - 1, \mathrm{ack}(n, m - 1)) & \text{sonst} \end{cases}$$

  Für andere Werte von n und/oder m soll das Ergebnis nil sein.

Denken Sie bei allen Operatoren auch an „ungewöhnliche“ Operandenwerte, z. B. negative Werte, Null (0), nil und synthetische Werte, und versuchen Sie, in jedem Fall ein möglichst sinnvolles Ergebnis zu liefern, das ggf. auch nil sein kann.

Da für benutzerdefinierte Operatoren momentan keine Vorrangregeln (außer dem automatischen Ausschluss der Nacheinanderausführung im vorderen und hinteren Operanden) definiert werden können, können bei ihrer Verwendung leicht Mehrdeutigkeiten entstehen, die dann durch explizite Klammerung aufgelöst werden müssen.