Zwischenbericht Daniel Gürber)

1 Abstract

Dieses Dokument zeigt auf wie die geplanten Erweiterungen aussehen und wie sie in der Syntax umgesetzt werden. Die geplanten Erweiterungen sind verkettete Vergleichsoperatoren und Prozeduraufrufe ohne call.

2 Erweiterung

2.1 Verkettete Vergleichsoperatoren

2.1.1 Ziel

Das Ziel ist es, IML so zu erweitern, dass Vergleichsoperatoren verkettet verwendet werden können. Ausdrücke sollen nicht mehrmals geschrieben werden müssen, um sie mit mehreren Werten zu vergleichen.

2.2 Beispiele

- x < y < z anstatt x < y cand y < z
- x = y = z anstatt x = y cand y = z
- a > b > c >= d anstatt a > b cand b > c cand c >= d

2.3 Aufrufe ohne call

2.3.1 Ziel

Das Ziel ist es, IML so zu erweitern, dass Prozeduraufrufe ohne das SchlÃijsselwort call möglich sind.

2.4 Beispiele

- doSomething(); anstatt call doSomething;
- calculateIt(x init); anstatt call calculateIt(x init);

3 Lexikalische Syntax

Die lexikalische Syntax ändert sich im Vergleich zu der von IML v3 nur dahingehend, dass das Schlüsselwort call gestrichen wird.

4 Grammatikalische Syntax

Die grammatikalische Syntax entspricht grösstenteils dervon IML v3 mit eingebauten Hilfskonstrukten, die folgenden Definitionen werden durch die Erweiterungen geändert:

```
term1 ::= term2 repTerm2

repTerm2 ::= RELOPR term2 repTerm2
           | epsilon

cmd ::= SKIP
      | expr auxExprCmd
      | IF LPAREN expr RPAREN blockCmd ELSE blockCmd
      | WHILE LPAREN expr RPAREN blockCmd
      | QUESTMARK expr
      | EXCLAMARK expr


auxExprCmd ::= BECOMES expr
             | auxGlobInitList


auxGlobInitList ::= INIT LPAREN globInitList RPAREN
                  | epsilon
```

Weitere Details können dem angehängten File Grammar_IML.sml entnommen werden.

5 Kontext- und Typeinschränkungen

Bei den Vergleichsoperatoren dÃijrfen weiterhin nur Typen verglichen werden, welche auch Vergleichbar miteinander sind. Der Kontext wird so eingeschrÃd'nkt, dass verkettete Vergleichsoperatoren die Richtung nicht ändern dürfen, das heisst wenn ein < oder <= Operator verwendet wurde, darf also kein > oder >= Operator verwendet werden und umgekehrt.
Das Weglassen des call Schlüsselwortes fÃijhrt zu keinen zusätzlichen Kontext- oder Typeinschränkungen.

6 Andere Programmiersprachen

Vergleichsoperatoren lassen sich in den meisten anderen Programmiersprachen grundsätzlich nicht verketten; Grund dafür ist, dass, wenn Expressions und Commands nicht sauber getrennt sind, der Command zwischen zwei Vergleichsoperatoren zweimal ausgeführt wird.
Prozeduraufrufe ohne call sind in vielen anderen Programmiersprachen möglich zum Beispiel in den von C abgeleiteten Sprachen.

7 Begründungen Entwurf

Die Einschränkung auf eine Richtung bei den Vergleichsoperatoren führt zu verbesserter Leserlichkeit, da die Werte von links nach rechts immer grösser beziehungsweise kleiner werden.