

# Verkettete Vergleichsoperatoren und Prozeduraufrufe ohne Schlüsselwort

## 1 Abstract

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

## 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  $x < y$  and  $y < z$
- $x = y = z$  anstatt  $x = y$  and  $y = z$
- $a > b > c \geq d$  anstatt  $a > b$  and  $b > c$  and  $c \geq d$

### 2.3 Aufrufe ohne Schlüsselwort

#### 2.3.1 Ziel

Das Ziel ist es, IML so zu erweitern, dass Prozeduraufrufe ohne ein Schlüsselwort `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 v4 nur dahingehend, dass das Schlüsselwort `call` gestrichen wird.

## 4 Grammatikalische Syntax

Die grammatikalische Syntax entspricht grösstenteils der von IML v4 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 dürfen weiterhin nur Typen verglichen werden, welche auch vergleichbar miteinander sind. Der Kontext wird so eingeschränkt, 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. Da der Gleichheitsoperator die Richtung nicht ändert, darf er verwendet werden, die Richtung muss aber auf beiden Seiten die selbe sein, dadurch wird sichergestellt das die Werte von links nach rechts nur kleiner bzw. nur grösser werden.

Das Weglassen des call Schlüsselwortes führt dazu, dass sichergestellt werden muss, dass nur Prozeduraufrufe ohne Zuweisungsoperator dastehen können und dass Prozeduraufrufe nicht in Expressions verwendet werden dürfen. Zudem darf die globale Init-Liste nicht hinter Funktionsaufrufen verwendet werden.

## 6 Andere Programmiersprachen

Vergleichsoperatoren lassen sich in den meisten anderen Programmiersprachen grundsätzlich nicht verketteten; Grund dafür ist, dass, wenn Expressions und Commands nicht sauber getrennt sind, der Befehl 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

### 7.1 Vergleichsoperatoren

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

## 7.2 Prozeduraufruf

Für das Weglassen des call Schlüsselwortes muss man bei der globalen Init-Liste Klammern hinzufügen, bei anderen Umsetzungen entstehen folgende Probleme:

- Werden die Klammern weggelassen, kann der Parser nicht zwischen globInitList und exprList unterscheiden, da beides kommagetrennte Listen sind.
- Wird globInitList als Teil des auxExprCmd aufgefasst entsteht ein Konflikt mit factor, da an beiden Orten optional ein INIT verwendet wird.

## 8 Codeerzeugung

Die Vergleichsoperatoren werden bei der Umwandlung vom konkreten zum abstrakten Syntaxbaum wieder mit einem logischen und verknüpft das diese in der VM nicht verkettet werden können. Die Prozeduraufrufe beeinflussen die Codeerzeugung nicht.

## 9 Beispielprogramme

Die folgenden IML Beispielprogramme werden im Anhang mitgesendet:

lauffähig:

- workingRelop.iml
- workingProc.iml

nicht lauffähig:

- directionRelop.iml (verschiedene Richtungen)
- typeRelop.iml (verschiedene Typen)
- assignProc.iml (Zuweisung an Prozeduraufruf)
- errorFun.iml (alleinstehender Funktionsaufruf)
- rvaueProc.iml (Prozeduraufruf in RValue)

## 10 Ehrlichkeitserklärung

Der eingereichte Schlussbericht ist das Resultat meiner persönlichen, selbstständigen Beschäftigung mit dem Thema. Der Compiler wurde in Zusammenarbeit mit Team06 (Hediger/Kolmann), Team04(Fässler/Oesch) und Team07(Künzli/Zeman) entwickelt.