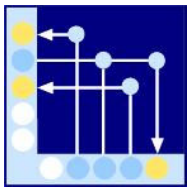

Programmieren in MOSTflexiPL

Vorlesung im Wintersemester 2025/2026
Prof. Dr. habil. Christian Heinlein

6. Aufgabenblatt (15. Dezember 2025)

Aufgabe 11: Weiterleitung von Operatoranwendungen

Teilaufgabe 11.a)

Definieren Sie einen Operator, der genau die Schnittmenge der Operatoren aus Aufgabe 7 und 8.b abdeckt, damit Ausdrücke wie `1 = 2 /= 3` auch dann eindeutig sind, wenn diese Operatoren beide sichtbar sind.

Implementieren Sie diesen neuen Operator, indem sie seine Anwendungen an einen der beiden anderen Operatoren weiterleiten.

Wie bei Aufgabe 9.a, soll auch hier jeder Operand nur ausgewertet werden, wenn sein Wert zur Ermittlung des Resultatwerts des gesamten Vergleichs benötigt wird.

Auch dieser Operator soll die gleichen Bindungseigenschaften wie der vordefinierte Gleichheitstest •=• haben.

Hinweis: Bei der Weiterleitung von Operatoranwendungen darf ein Lambda-Parameter nur an einen anderen Lambda-Parameter und ein gewöhnlicher Parameter nur an einen anderen gewöhnlichen Parameter weitergeleitet werden, obwohl der Übersetzer fälschlicherweise auch andere Konstellationen erlaubt, die dann zur Laufzeit jedoch nicht korrekt funktionieren.

Teilaufgabe 11.b)

Eliminieren Sie, sofern möglich, Codeverdopplungen in den Implementierungen Ihrer in den bisherigen Aufgaben definierten Operatoren durch vollständige oder partielle Weiterleitungen an geeignete lokale Hilfsoperatoren.

Da partielle Weiterleitungen manchmal zu Laufzeitfehlern führen, verwenden Sie, wenn möglich, lieber vollständige Weiterleitungen. Zum Beispiel:

```
(while|until|do) [(X:type)] (\ x -> (X))
{ (while|until|do) [(Y:type)] (\ y -> (Y)) }
end -> (int =
    ......

    (?* (
        $$ Direkte Implementierung:
        ......;                 $$ Verarbeitung von x.
        { ...... };             $$ Wiederholte Verarbeitung von y
                                $$ auf die gleiche Weise wie x.
```

```
            $$ Alternativ mit lokalem Hilfsoperator:
            aux { (while|until|do) [(Z:type)] (\ z -> (Z)) } -> (int =
                { ..... }          $$ Wiederholte Verarbeitung von z.
            );
            <>(aux);

            ......
        )) - 1
    )
```

Aufgabe 12: Flexible Ausgabe von `int`-, `char`- und `bool`-Werten

Definieren Sie einen Operator `print`, der beliebig viele `int`-, `char`- und `bool`-Werte nacheinander ausgibt.

Wie beim vordefinierten Operator `print`, wird nach der Ausgabe aller Werte ein abschließender Zeilentrenner ausgegeben, sofern `only` nicht angegeben ist.

Nach `print` und optional `only` können beliebig viele Operanden der Typen `int`, `char` und `bool` sowie beliebig viele der folgenden Wörter und Phrasen in beliebiger Reihenfolge angegeben werden:

- `bin`: Binäre Ausgabe von `int`-Werten.
- `oct`: Oktale Ausgabe von `int`-Werten.
- `dec`: Dezimale Ausgabe von `int`-Werten (Voreinstellung).
- `hex`: Hexadezimale Ausgabe von `int`-Werten.
- `lower`: Verwendung von Kleinbuchstaben bei der hexadezimalen Ausgabe von `int`-Werten und bei der Ausgabe von `bool`-Werten mit Buchstaben (Voreinstellung).
- `upper`: Verwendung von Großbuchstaben bei der hexadezimalen Ausgabe von `int`-Werten und bei der Ausgabe von `bool`-Werten mit Buchstaben.
- `letter`: Ausgabe der `bool`-Werte `true` und `false` mit den Buchstaben `t` und `f` bzw. `T` und `F`, abhängig von `lower` oder `upper`.
- `digit`: Ausgabe der `bool`-Werte `true` und `false` mit den Ziffern `1` und `0` (Voreinstellung).
- `sign`: Ausgabe der `bool`-Werte `true` und `false` mit den Zeichen `+` und `-`.
- `chars t f` mit zwei beliebigen `char`-Werten `t` und `f`:
  Ausgabe der `bool`-Werte `true` und `false` mit den Zeichen `t` und `f`.
- `sep s`: Zwischen je zwei Werten wird das Trennzeichen `s` ausgegeben (Voreinstellung ist ein Leerzeichen).
- `tight`: Zwischen je zwei Werten wird kein Trennzeichen ausgegeben.
- `reset`: Alle Einstellungen werden auf die o. g. Standardwerte zurückgesetzt.

Jede dieser Angaben gilt so lange für alle nachfolgenden Operanden der aktuellen und aller nachfolgenden Anwendungen von `print`, bis sie durch eine andere Angabe aus derselben Kategorie (`bin`/`oct`/`dec`/`hex`, `lower`/`upper`, `letter`/`digit`/`sign`/`chars`, `sep`/`tight`) überschrieben oder mittels `reset` auf die Standardeinstellung dieser Kategorie zurückgesetzt wird.

Nach jedem Wert kann optional mit `width` ein `int`-Wert als Mindestbreite für die Ausgabe dieses Werts angegeben werden. Wenn die Breite des Werts (d. h. die Anzahl der Zeichen für seine Ausgabe) dann kleiner als die Mindestbreite ist, werden vor dem Wert entsprechend viele Leerzeichen ausgegeben. (Das heißt umgekehrt: Wenn die Breite des Werts größer oder gleich der Mindestbreite ist oder wenn die Mindestbreite ein unnatürlicher Wert ist, werden keine zusätzlichen Leerzeichen ausgegeben.)

Wie beim vordefinierten Operator `print`, soll die Ausgabe unnatürlicher `int`- und `char`-Werte an sich leer sein. Jeder `bool`-Wert außer `false` soll wie `true` ausgegeben werden.

Zum Beispiel:

```
$$ Anwendung von print:                $$ Zugehörige Ausgabe:

print 10 20 30;                        $$ 10 20 30
print 10 hex 20 30;                    $$ 10 14 1e
print true false;                      $$ 1 0
print sep '|' letter upper true false; $$ T|F
print 0 tight chars 'W' 'F' true;      $$ 0W
print dec sep '|' 0 1:0 width 2 -123 width 5 456 width 1:0;
                                       $$ 0|  | -123|456
print;                                 $$ (Nur eine Leerzeile)
print only reset                       $$ (Zurücksetzen aller Einstellungen
                                       $$ ohne irgendeine Ausgabe)
```