Hochschule Aalen
Fakultät Elektronik und Informatik
Studienbereich Informatik

Advanced Programming with MOSTflexiPL

Lecture in Wintersemester 2025/2026
Prof. Dr. habil. Christian Heinlein

4. Task Sheet (November 25, 2025)

Task 8: Generic Operators

Subtask 8.a)

Define the logic operators from task 1 generically, so that they can be applied to values `x` and possibly `y` of arbitrary (even different) types `X` and `Y`:

- The negation `/ x` returns `true` if `x` is nil and `false` otherwise (i. e., the result type is still `bool`).
- The conjunction `x /\ y` returns `y` if `x` is not nil and otherwise nil of type `Y` (i. e., the result type is `y`).
- The disjunction `x \/ y` returns `y` if `x` is nil and otherwise a new synthetic value of type `Y` (i. e., the result type is `Y`, too).

Additionally define a specialization of the disjunction for values `x` and `y` of the same type `XY`, which returns `x` if it is not nil and `y` otherwise (i. e., the result type is `XY`).
According to the rules explained in §3.9.2 of the lecture slides, this specialization is preferred over the more general operator in an application of the disjunction if both operands have the same type.

In addition to the binding properties already defined in task 1, define:

- Both variants of the disjunction have the same binding properties.
- In the left operand of the disjunction and conjunction, all operators are excluded which are excluded in the left operand of the predefined assignment `•=!•` (which includes the assignment itself).
- In the right operand of the disjunction and conjunction as well as in the operand of the negation, all operators are excluded which are excluded in the right operand of the predefined assignment plus the assignment itself (which is not excluded in its own right operand).
- Disjunction and conjunction are excluded everywhere where the predefined equality test `•=•` is excluded.
- The negation is excluded in the left operand of the predefined equality test as well as in the operand of the predefined negative test `•-`. Furthermore, it is excluded everywhere where the predefined operator `char•` (conversion from `int` to `char`) is excluded.

In summary, the effect of these exclusions is to position the three logical operators in the precedence hierarchy of §2.13.1 between the assignment and the equality test.

Subtask 8.b)

Define, similar to task 7, an operator that allows chained equality/inequality tests with two or more operands of any type, for example:

```
b1 = b2 =/ b3          $$ with b1, b2, b3 of type bool
c1 =/ c2 = c3 = c4     $$ with c1, c2, c3, c4 of type char
```

Just as the operator from task 7, this operator shall also have the same binding properties as the predefined equality test `•=•`.

Subtask 8.c)

Define the operator to swap variable values from task 3 as well as the operator to rotate variable values from task 6 generically, so that they can be applied to variables of any content type.

Subtask 8.d)

Define a generic postfix operator `•?` such that `x?`, for a value `x` of any type `X`, always returns a new variable with type `X?` and initial value `x`.

Define the following exclusions for this operator:

- The predefined constant declaration is excluded in the operand of this operator, so that `v := 1?` defines a variable `v` with content type `int` and initial value `1` (and not a constant `v` with type `int` and value `1` inside of the operand of the operator `•?`).
- The operator itself is excluded in the operand of the predefined loop operator `?*•` as well as in the operand of the predefined query operator `?•`.

Further exclusions are not necessary, because many other expressions such as `1 + 2 ?` or `x =! y ?` are unambiguous simply because of the types of the involved subexpressions.