Typesystem cookbook

Type equations

Simple case: inference rule for a constant; type of concept instance always equals some concrete type

typeOf_IntegerLiteral

```
rule typeOf_IntegerLiteral {
  applicable for concept = IntegerLiteral as integerLiteral
  overrides false

  do {
    typeof(integerLiteral) :==: <int>;
  }
}
```

Example 1: type of instance of integer literal is always instance of IntegerType (int)

Another simple case: inference rule for variable reference; type of concept instance equals to the type of some other node

typeof_BaseVariableReference

```
rule typeof_BaseVariableReference {
  applicable for concept = BaseVariableReference as reference
  overrides false

  do {
    typeof(reference) :==: typeof(reference.baseVariableDeclaration);
  }
}
```

Example 2

Subtyping

checking, inference,

typeOf_TernaryOperator

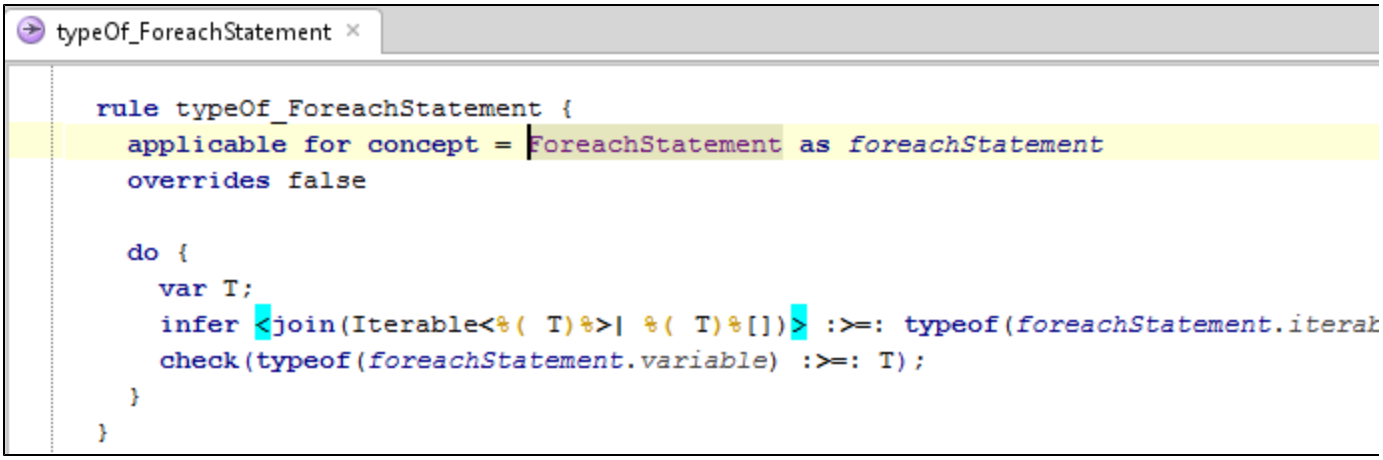
```
rule typeOf_TernaryOperator {
  applicable for concept = TernaryOperatorExpression as toe
  overrides false

  do {
    infer typeof(toe.condition) :<=: <boolean>;
    infer typeof(toe) :>=: typeof(toe.ifTrue);
    infer typeof(toe) :>=: typeof(toe.ifFalse);
  }
}
```

Example. conditional ternary operator. Type of condition should be boolean.

Inequality is used here, because in MPS baseLanguage there is a BooleanType (boolean) and ClassifierType Boolean from java and they are weak subtypes of each other.

If condition evaluates to "true" this operator will be substituted with "toe.ifTrue", so the type of "ifTrue" should be a subtype of the type of operator. The same for "ifFalse".

Another inference example using inequalities:



Example for each loop

Subtyping

defining