Compiler Practical 2013

Syntax Analysis of Expressions

Berthold Hoffmann (B. Gersdorf, T. Röfer)
hof@informatik.uni-bremen.de
Cartesium 2.48

Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

Universität Bremen

Structure

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH
Universität Bremen

1. Syntax Analysis of Expressions
2. Tasks: Adding *AND*, *OR, a*nd *NOT*
3. Task: Adding the Class *Boolean*
4. Bonus Task: Adding *AND THEN* and *OR ELSE*

Precedence of Operators

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH
Universität Bremen

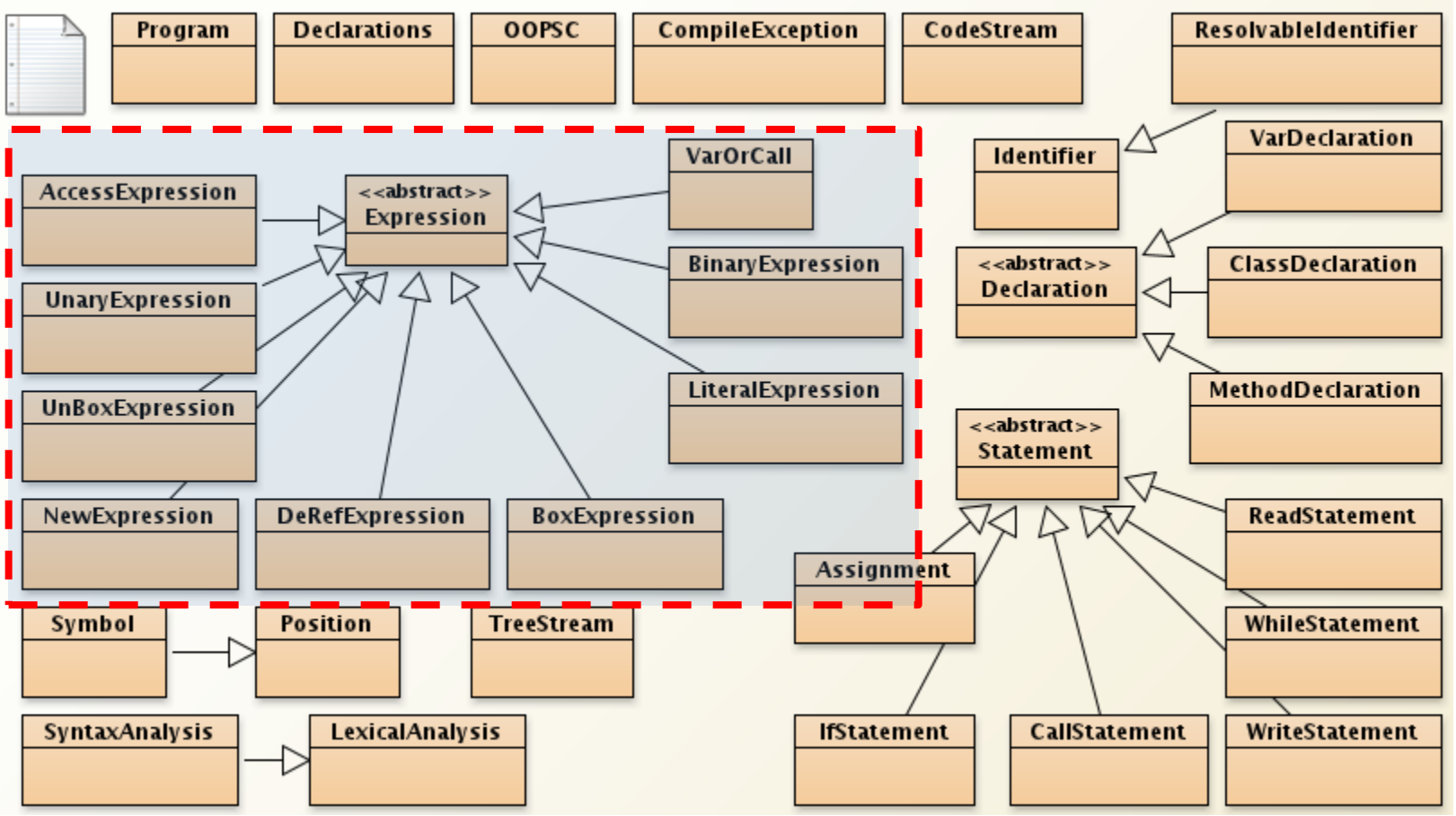
Precedence	Operators	Assoziativity	Description
	NEW	(prefix) right	object generation
	.	left	object access
	-	(prefix) right	sign
	*, /, MOD	left	multiplication, division, modulus
	+, -	left	addition, subtraction
	<, <=, >, >=, =, #	Not associative	Less, less-or-equal, greater, greater-or-equal, equality, inequality

Expressions: Grammar

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH
Universität Bremen

relation	::=	*expression*
		[('=' | '#' | '<' | '>' | '<=' | '>=') *expression*]
expression	::=	*term* { ('+' | '-') *term* }
term	::=	*factor* { ('*' | '/' | MOD) *factor* }
factor	::=	'-' factor
	|	memberaccess
memberaccess	::=	*literal* { '.' *varorcall* }
Literal	::=	*number*
	|	NULL
	|	SELF
	|	NEW *identifier*
	|	'(' *expression* ')'
	|	*varorcall*

Expressions: Abstract Syntax



Task: AND, OR, NOT

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH
Universität Bremen

- Lexical analysis
  - Add keywords AND, OR, NOT
- Extend the grammar
  - Respect precedence of operators
  - In analogy with expressions / term / factor
- Bracketed expressions can be more general now

```
IF NOT (a < b AND b < c)
        AND (a < b) # (b < c)
        OR c > a
THEN
   …
```

Task: AND, OR, and NOT

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH
Universität Bremen

- Extend syntax analysis
  - New methods for new productions
  - Extending *factor*
- Extend syntax tree
  - *UnaryExpression*, *BinaryExpression*
  - and *contextAnalysis*(…), *generateCode*(…), resp.

5%

Aufgabe: AND, OR, NOT Bindung

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH
Universität Bremen

precedence	Operators	Assoziativity	Decription
	NEW	(prefix) right	object generation
	.	left	object access
	-, **NOT**	(prefix) right	sign
	*, /, MOD	left	multiplication, division, modulus
	+, -	left	addition, subtraction
	<, <=, >, >=, =, #	not associative	Less, less-or-equal, greater, greater-or-equal, equality, inequality
	AND	left	conjunction
	OR	left	Disjunction

Task: Adding the class *Boolean*

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH
Universität Bremen

- New type
  - Class attribute for new predefined type in *ClassDeclaration*
  - Initializing type, and introducing it in *Program.contextAnalysis*()
- [Un]Boxing
  - *Expression.box/unBox*(…)
  - [*Un*]*BoxExpression*.[*Un*]*BoxExpression*(…)
  - Analoguously to *ClassDeclaration.intClass*

10%

```
METHOD main IS
   a, b : Boolean;
BEGIN
   a := 5 > 7;
   b := 1 < 2 OR a;
   a := a # b;
   IF b THEN
   END IF
END METHOD
```

Bonus: AND THEN and OR ELSE

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH
Universität Bremen

- Both operators stop evaluation as soon as the result is clear
  - *a AND THEN b AND THEN c*
  - *a OR ELSE b OR ELSE c*
- Problems
  - How should the grammar be defined?
  - How should the code be generated?
  - What happens with *a AND THEN b AND c*?

5%