Parsergenerator CUP

- **Bottom up Strategie (LALR)**
- **Scott Hudson, Andrew Appel an Georgia Institute of Technology, USA**
- **In Appels Buch wird die Verwendung von CUP beschrieben**

CUP – Format der Parserspezifikation

User Code

Terminale und Nonterminale

Operatorpräzedenzen

Gammatik

CUP - Teil der expr-Grammatik

```
ausdr  ::=   ausdr ADDOP term
        |          term
        ;

term   ::=   term MULOP fact
        |    fact
        ;

fact   ::=    INTCONST
        |     IDENT
        |     LPAR ausdr RPAR
        ;
```

Terminalsymbole groß geschrieben

JFlex – Lexikalische Regeln

```
Ident        =  [a-zA-Z] [a-zA-Z0-9_]*

IntLiteral   =  0 | [1-9] [0-9]*

%%

 " + "          { return new Symbol(sym.ADDOP,yyline+1 , yycolumn+1, new Integer(1) ); }

{Ident}          { return new Symbol(sym.IDENT, yytext()); }

{IntLiteral}    { return new Symbol(sym.INTCONST, new Integer(Integer.parseInt(yytext()))); }
```

Deklarationen

Lexikalische Regeln

Parser – Literatur

- Andrew W. Appel: *Modern Compiler Implementation in Java,*

  Basic Techniques, Cambridge University Press 1997.
  (Chapter 3: Parsing, S. 40 – 77, leicht lesbar)

- Wilhelm W., Maurer D.: *Übersetzerbau: Theorie, Konstruktion, Generierung*, Springer-Verlag 1992.

  (Kapitel 7: Syntaktische Analyse, S. 217 – 349, sehr ausführlich)

classgen

- Am Lehrstuhl entwickelt (Gerwin Klein)
- Erzeugt Java-Klassen aus einer Grammatik (wie JavaCC und Sable)
- Sehr nützlich für abstrakte Syntaxbäume (Konstruktion,Baumdurchläufe)

- Reguläre rechte Seiten in der Grammatik
- Attribute
- Visitor-Schnittstelle (für Baumdurchläufe)
- Aspekte – aus Attributregeln werden visit-Methoden generiert

classgen – Format der Eingabe

- Attributdefinitionen
- Grammatik
- Methodendefinitionen
- Aspektdefinition

Abstrakte Syntax für Ausdrücke

```
Ausdr   ::= {BinAusdr}   Ausdr   "int":op   Ausdr
          |  {IntConst}   "int":value
          |  {Var}        "String":ident
```

Generierte Klasse Ausdr

```
public abstract class Ausdr implements SyntaxNode {

  private SyntaxNode parent;

  public SyntaxNode getParent() {
    return parent;
  }
   public void setParent(SyntaxNode parent) {
    this.parent = parent;
  }
  public abstract void accept(Visitor visitor);
}
```

Generierte Klasse BinAusdr

```
public class BinAusdr extends Ausdr {

  public Ausdr ausdr;
  public int op;
  public Ausdr ausdr2;

  public BinAusdr (Ausdr ausdr, int op, Ausdr ausdr2) {
    this.ausdr = ausdr;
    if (ausdr != null) ausdr.setParent(this);
    this.op = op;
    this.ausdr2 = ausdr2;
    if (ausdr2 != null) ausdr2.setParent(this);
  }
  public void accept(Visitor visitor) { visitor.visit(this); }
```

Generierte Klasse IntConst

```
public class IntConst extends Ausdr {

  public int value;

  public IntConst (int value) {
    this.value = value;
  }

  public void accept(Visitor visitor) { visitor.visit
(this); }
```

Baumaufbau durch Parseraktionen

```
ausdr  ::= ausdr:a  ADDOP:op  term:t
               {: RESULT = new BinAusdr(a, op.intValue(), t); :}

         |     term:t
               {: RESULT = t; :}
         ;


fact     ::=  INTCONST:i
               {: RESULT = new IntConst(i.intValue()); :}
```