Compilers

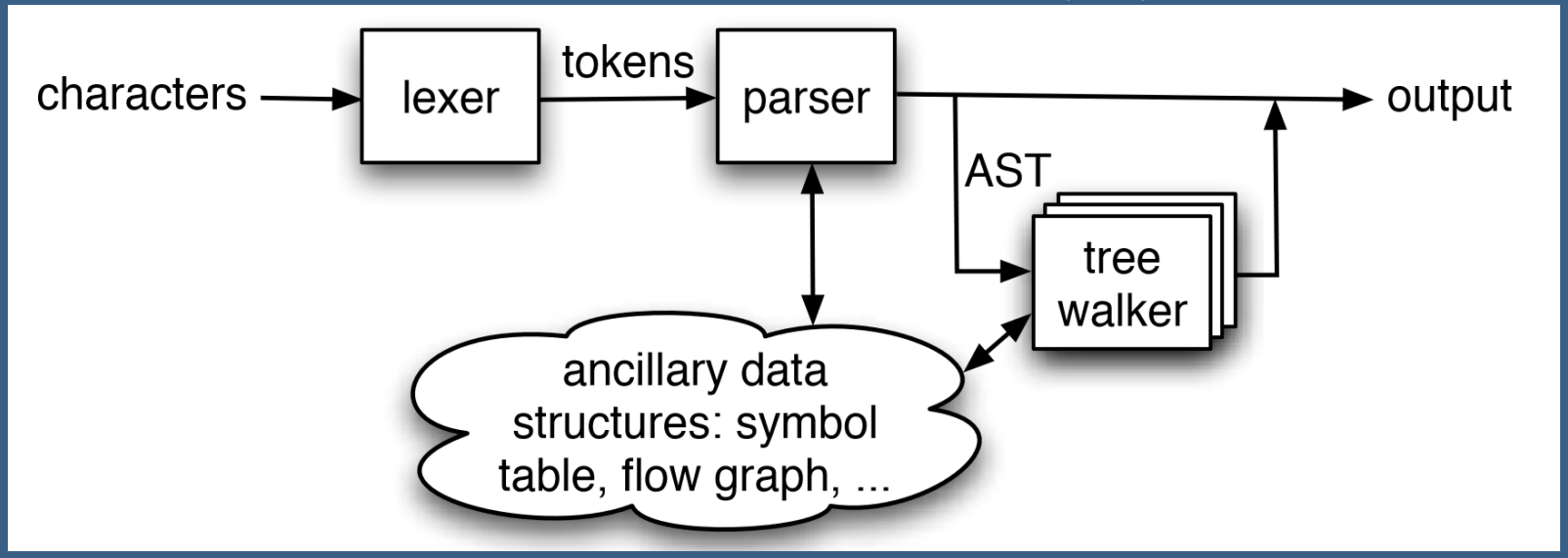
c2p: mvn-driven ANTLR

Naomi Christis

Universiteit Antwerpen

Introduction

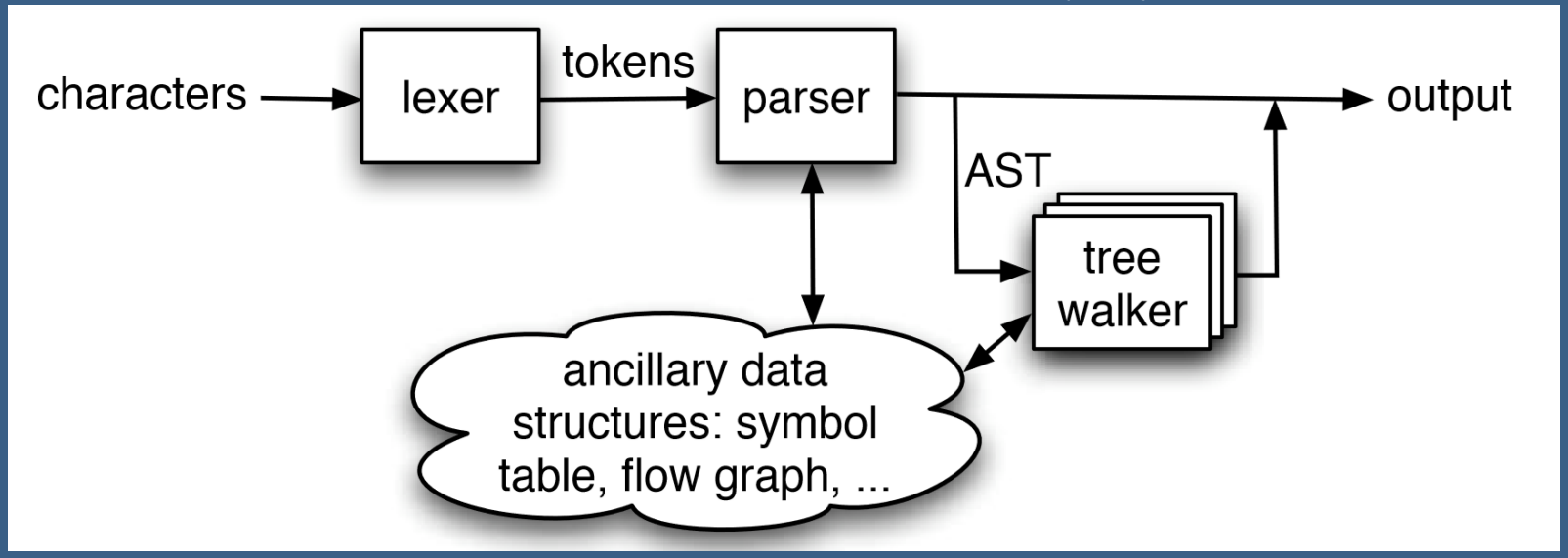
Source: T. Parr, The Definitive ANTLR Reference, Pragmatic Bookshelf (2007)



Plain text: a=b; //assign b to a

Introduction

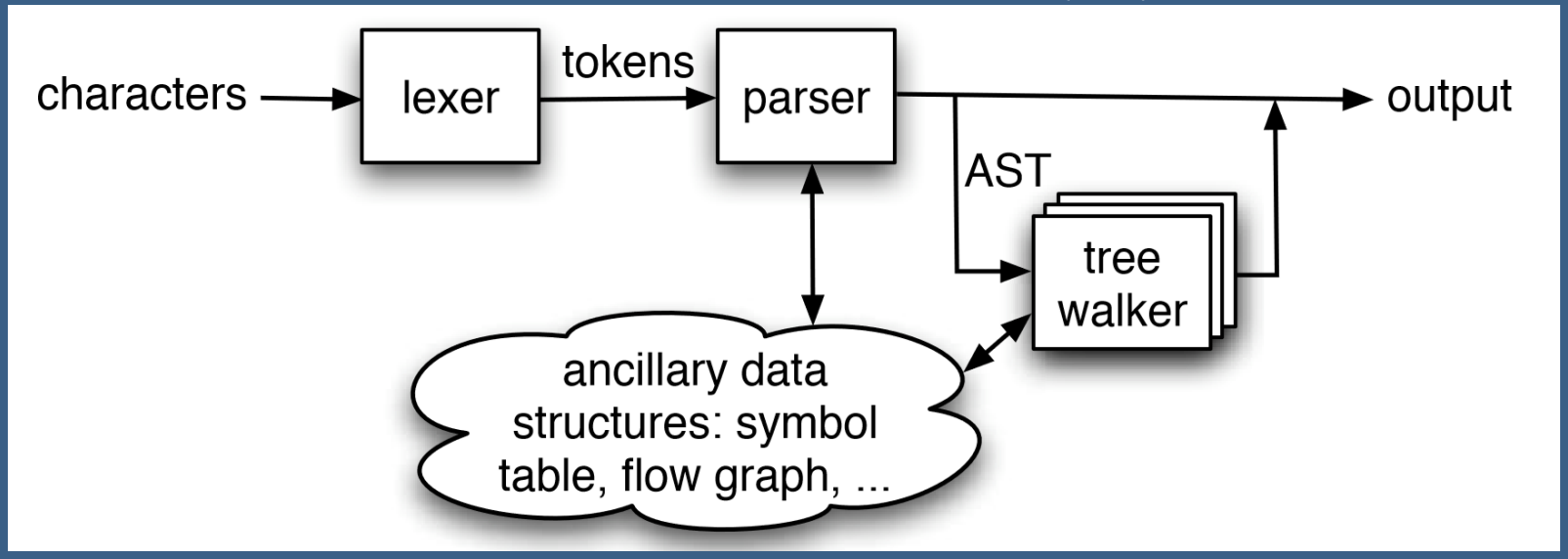
Source: T. Parr, The Definitive ANTLR Reference, Pragmatic Bookshelf (2007)



Tokens: ID EQ ID COL COMMENT

Introduction

Source: T. Parr, The Definitive ANTLR Reference, Pragmatic Bookshelf (2007)



Introducing ANTRL

- Another Tool for Language Recognition
- Reads a grammar .g file
  - specifies a language
  - fixed structure
- Generates a recognizer for that language
  - lexer: tokenising input stream (regexp)
  - parser: parse token stream (rewrite rules) $\rightarrow$ AST

Installing ANTRL

`http://www.antlr.org/`

- ANTLR IDE requires:
  - Eclipse 4.4-Luna
  - XText 2.5.x/2.6
  - Java 5.0+
- Install via Update Manager:
  - `http://download.eclipse.org/modeling/tmf/xtext/updates/composite/releases/`
  - `http://dl.bintray.com/jknack/antlr4ide`

Combined Grammar

Expression Language to support:

- Operators: $+$, $-$ and $*$
- Parenthesizes expressions to alter the order of operator evaluation
- Variable assignments and references

Recognizing Language Syntax

```
grammar Expr;

prog: stat+;

stat: expr NEWLINE
    | ID EQ expr NEWLINE
    | NEWLINE
    ;

expr: multExpr (
        (MINUS | PLUS) multExpr )*;

multExpr: atom (STAR atom )*;

atom:
    INT
  | ID
  | LPAREN expr RPAREN
  ;
```

```
PLUS: '+';
MINUS: '-';
STAR: '*';
LPAREN: '(';
RPAREN: ')';
EQ: '=';
ID: ('a'..'z'|'A'..'Z')+;
INT: '0'..'9'+;
NEWLINE: '\r'?'\n';
WS:(' '|'\t'|'\n'|'\r')+{skip();};
```


Testing the Recognizer

```
public class Main {

   public static void main(String[] args) throws Exception{
      ANTLRInputStream input = new ANTLRInputStream(System.in);

      ExprLexer lexer = new ExprLexer(input);
      CommonTokenStream tokens = new CommonTokenStream(lexer);

      ExprParser parser = new ExprParser(tokens);

      parser.prog();
   }
}
```


Testing the Recognizer

```
public class Main {

   public static void main(String[] args) throws Exception{
      ANTLRInputStream input = new ANTLRInputStream(System.in);

      ExprLexer lexer = new ExprLexer(input);
      CommonTokenStream tokens = new CommonTokenStream(lexer);

      ExprParser parser = new ExprParser(tokens);

      parser.prog();
   }
}
```

Problems @ Javadoc Declaration Console

<terminated> Main [Java Application] /usr/lib/jvm/java-7-openjdk-amd64/bin/java (Nov 24, 2014, 12:00:15 PM)

3+3

Testing the Recognizer

```
public class Main {

   public static void main(String[] args) throws Exception{
      ANTLRInputStream input = new ANTLRInputStream(System.in);

      ExprLexer lexer = new ExprLexer(input);
      CommonTokenStream tokens = new CommonTokenStream(lexer);

      ExprParser parser = new ExprParser(tokens);

      parser.prog();
   }
}
```

Problems @ Javadoc Declaration Console

```
<terminated> Main [Java Application] /usr/lib/jvm/java-7-openjdk-amd64/bin/java (Nov 24, 2014, 12:00:15 PM)
3+(4
line 1:4 mismatched input '\n' expecting {'+', '-', '*', ')'}
```


Testing the Recognizer

```
public class Main {

   public static void main(String[] args) throws Exception{
      ANTLRInputStream input = new ANTLRInputStream(System.in);

      ExprLexer lexer = new ExprLexer(input);
      CommonTokenStream tokens = new CommonTokenStream(lexer);

      ExprParser parser = new ExprParser(tokens);

      parser.prog();
   }
}
```

Problems @ Javadoc Declaration Console

```
<terminated> Main [Java Application] /usr/lib/jvm/java-7-openjdk-amd64/bin/java (Nov 24, 2014, 11:57:00 AM)
3++
line 1:2 no viable alternative at input '+'
```


Attach actions to grammar elements

```
grammar Expr;

@header {
  import java.util.HashMap;
}

@members {
  HashMap memory = new HashMap();
}

prog: stat+;

stat:   expr NEWLINE
{System.out.println($expr.value);}
    | ID '=' expr NEWLINE
      {memory.put($ID.text, new
Integer($expr.value));}
    | NEWLINE
    ;
```

```
Problems @ Javadoc Declaration Cons
<terminated> Main [Java Application] /usr/lib/jvm/ja
3+(4*5)
23
```

```
expr returns [int value]
  :   e=multExpr {$value=$e.value;}
    (
      '-' e=multExpr {$value -= $e.value;}
      |
      '+' e=multExpr {$value += $e.value;}
    )*
  ;
multExpr returns [int value]
  : e=atom {$value= $e.value;} ('*' e=atom {$value *= $e.value;} )*
  ;

atom returns [int value]
  : INT {$value = Integer.parseInt($INT.text);}
  | ID
    {
      Integer v = (Integer)memory.get($ID.text);
      if(v!=null) $value = v.intValue();
      else System.err.println("undefined variable " + $ID.text);
    }
  | '(' expr ')' {$value = $expr.value;}
  ;
```


Attach actions to grammar elements

```
grammar Expr;

@header {
  import java.util.HashMap;
}

@members {
  HashMap memory = new HashMap();
}

prog: stat+;

stat:   expr NEWLINE
{System.out.println($expr.value);}
    | ID '=' expr NEWLINE
      {memory.put($ID.text, new
Integer($expr.value));}
    | NEWLINE
    ;
```

```
Problems @ Javadoc Declaration Cons
<terminated> Main [Java Application] /usr/lib/jvm/ja
a=4
3+a
7
```

```
expr returns [int value]
  :   e=multExpr {$value=$e.value;}
    (
      '-' e=multExpr {$value -= $e.value;}
      |
      '+' e=multExpr {$value += $e.value;}
    )*
  ;
multExpr returns [int value]
  : e=atom {$value= $e.value;} ('*' e=atom {$value *= $e.value;} )*
  ;

atom returns [int value]
  : INT {$value = Integer.parseInt($INT.text);}
  | ID
    {
      Integer v = (Integer)memory.get($ID.text);
      if(v!=null) $value = v.intValue();
      else System.err.println("undefined variable " + $ID.text);
    }
  | '(' expr ')' {$value = $expr.value;}
  ;
```


Exercise

Let a list be defined as follows:

- (a, b) is a list of atoms
- if $L1, L2, \ldots Ln$ are lists, then $(L1, L2, \ldots Ln)$ is a list

Example: $((a), ((b), (c, d)))$ is a list
Develop a recognizer for such a list using ANTLR.

Exercise Solution

```
grammar Lists;

prog: item;

item: ATOM
    | list;

list: '(' item (',' item)* ')';

ATOM:('a'..'z')+;
WS:(' '|'\t'|'\n'|'\r')+{skip();};
```