

Compiler Construction

Under 12 parsers

Flip van Spaendonck & Lars Kuijpers

March 14, 2018



- Bottom-up
- Separate lexer and parser
 - With mid-level expression parser
- Written in Java



```
FArgs = id [',' FArgs]
Field = [('.hd' | '.tl' | '.fst' | '.snd') Field]
Exp == BoolExp2
| NumRng
| '[]'
| '(' Exp, Exp)'
```



Grammar Transformation

BoolExp2 = BoolExp1 && BoolExp2

| BoolExp1 || BoolExp2

| NumRng == NumRng

| NumRng != NumRng

| NumRng < NumRng

| NumRng > NumRng

| NumRng <= NumRng

| NumRng >= NumRng

BoolExp1 = ['!'] BoolExp0

BoolExp0 = 'True'

| 'False'

| id Field

| FunCall

| '(' BoolExp2 ')'



```
NumRng = NumFld '%' NumRng  
| NumFld '/' NumRng  
| NumFld '*' NumRng  
| NumFld
```

```
NumFld = NumSng '+' NumFld  
| NumSng '-' NumFld  
| NumSng
```

```
NumSng = int  
| ''' char '''  
| id Field  
| FunCall  
| '(' NumRng ')'
```



- Input "4 -5" is recognized as two digits, resulting in error
- Bottom-up



- Higher level tokens for expressions
 - Reduce runtime/memory complexity
- High memory usage
 - Optimize garbage collection



Questions ?

