Introduction to language theory and compiling Project – Part 2

Antoine Passemiers Alexis Reynouard

November 14, 2017

Contents

1	Trai	nsformi	ng Imp grammar	1
	1.1	Removing useless rules		2
		1.1.1	Unreachable variables	2
		1.1.2	unproductive variables	2
	1.2	Remov	ving left-recursion and applying factorization	2
		1.2.1	Left-recursion	2
		1.2.2	Factorization	2
	1.3	Remov	ving ambiguity	2
		1.3.1	Operator priority	2
		1.3.2	Operator associativity	2
	1.4	Result	ing grammar	2

1. Transforming Imp grammar

This part of the project consists in implementing a LL(k) parser for the Imp programming language. A LL(k) parser is a recursive descent parser composed of:

• An input buffer, containing k input tokens. Since we are considering a LL(1) parser, the latter only considers one token at a time to decide how to grow the syntactic tree.

•

1.1 Removing useless rules

- 1.1.1 Unreachable variables
- 1.1.2 unproductive variables
- 1.2 Removing left-recursion and applying factorization
- 1.2.1 Left-recursion
- 1.2.2 Factorization
- 1.3 Removing ambiguity
- 1.3.1 Operator priority
- 1.3.2 Operator associativity
- 1.4 Resulting grammar

```
[1] <Program> \rightarrow begin <Code> end
```

[2] <Code> $\rightarrow \epsilon$

 $[3] \quad <> \qquad \quad \rightarrow < InstList>$