TDT4205 - Øving 2

Daniel Romanich

February 2018

1 LL(1) parsing table construction

Grammar:
S ⇒ c s a U O
U ⇒ n
O ⇒ L | ϵ
L ⇒ E | L E
E ⇒ b | v | p

The problem with this grammar when using an LL parser is that the L production is left recursive. In order to fix this we need to introduce a new production L'.
L ⇒ EL'
L' ⇒ EL' | ϵ

By now making L ⇒ EL' the production is no longer left recursive, but still yields the same results.

With our new grammar we can now construct a table for the First and Follow sets of the grammar:

Non-terminals	First	Follow
S	c	$
U	n	$, b, v, p
O	ϵ, b, v, p	$
L	b, v, p	$
E	b, v, p	$, b, v, p
L'	ϵ, b, v, p	$

We now have the first and follow sets of the grammar, so now we can create the LL(1) parsing table for it:

Non-terminal	c	s	a	n	b	v	p	$
S	S $\Rightarrow$ c s a U O							
U				U $\Rightarrow$ n				
O					O $\Rightarrow$ L	O $\Rightarrow$ L	O $\Rightarrow$ L	O $\Rightarrow \epsilon$
L					L $\Rightarrow$ EL'	L $\Rightarrow$ EL'	L $\Rightarrow$ EL'	
E					E $\Rightarrow$ b	E $\Rightarrow$ v	E $\Rightarrow$ p	
L'					L $\Rightarrow$ EL'	L $\Rightarrow$ EL'	L $\Rightarrow$ EL'	L' $\Rightarrow \epsilon$

2 VSL specification

The code is provided together with this PDF. This part of the exercise took a really long time. Figuring out how everything worked was not too easy. I spent a lot of time figuring out how to create the productions and how they were tied to the tokens created by the scanner. Once I finally understood how it worked (the powerpoint about the exercise helped A LOT) creating the productions and linking them up to the nodes was relatively easy. As commented in the code I spent a lot of time figuring out a bug causing a segfault. By debugging the printing of the tree I was able to pinpoint which node type that caused it. I had made a small mistake where I was just putting the parsed integer directly into the pointer instead of allocating memory for the int, then updating the data pointer to that memory location. After fixing the bug everything started working.