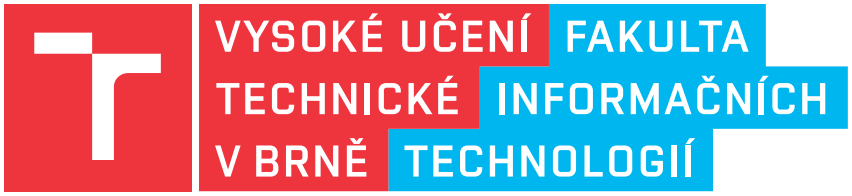


Projektová dokumentácia

Implementácia prekladača imperatívneho jazyka IFJ21

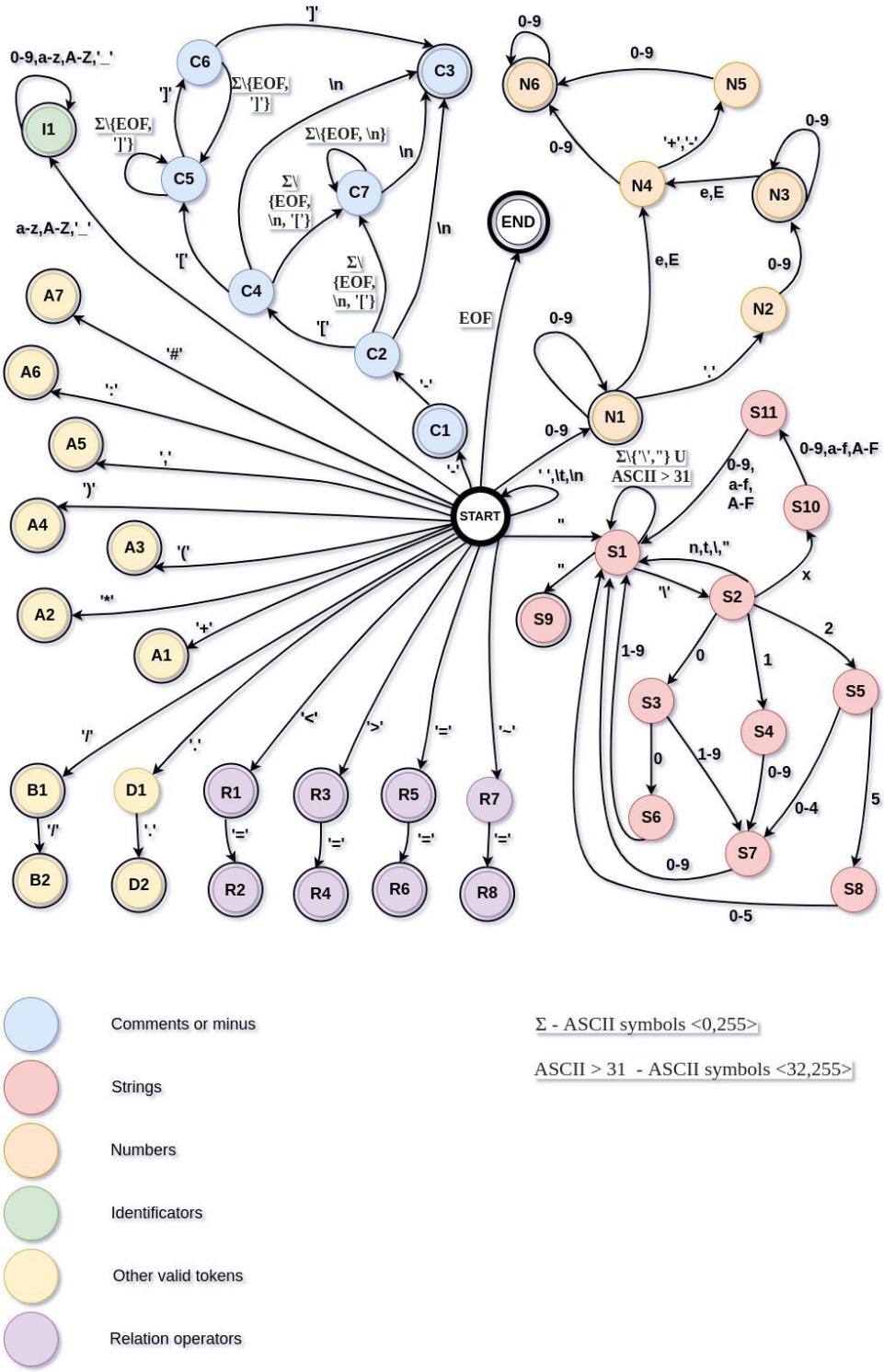
Tým 082, varianta II

Andrei Shchapaniak	**(xshcha00)**	25 %
Andrej Binovsky	(xbinov00)	25 %
Zdenek Lapes	(xlapes02)	25 %
Richard Gajdosik	(xgajdo33)	25 %

7. decembra 2021

Obsah

Diagram konečného automatu specifikujúceho lexikálny analyzátor



Obr. 1: Diagram konečného automatu specifikující lexikální analyzátor

LL – gramatika

1. <**prolog**> → `require t_string` <**prog**>
2. <**prog**> → `global id :  function (` <**arg_T**> `)` <**ret_T**> <**prog**>
3. <**prog**> → `function id (` <**arg**> `)` <**ret_T**> <**stmt**> `end` <**prog**>
4. <**prog**> → `id (` <**param**> `)` <**prog**>
5. <**prog**> → `EOF`
6. <**arg_T**> → <**type**> <**next_arg_T**>
7. <**arg_T**> → ε
8. <**next_arg_T**> → `,` <**type**> <**next_arg_T**>
9. <**next_arg_T**> → ε
10. <**ret_T**> → `:` <**type**> <**next_ret_T**>
11. <**ret_T**> → ε
12. <**next_ret_T**> → `,` <**type**> <**next_ret_T**>
13. <**next_ret_T**> → ε
14. <**arg**> → `id :` <**type**> <**next_arg**>
15. <**arg**> → ε
16. <**next_arg**> → `, id :` <**type**> <**next_arg**>
17. <**next_arg**> → ε
18. <**type**> → `integer`
19. <**type**> → `number`
20. <**type**> → `string`
21. <**type**> → `nil`
22. <**stmt**> → `if` <**expr**> `then` <**stmt**> `else` <**stmt**> `end` <**stmt**>
23. <**stmt**> → `while` <**expr**> `do` <**stmt**> `end` <**stmt**>
24. <**stmt**> → `local id :` <**type**> <**def_var**> <**stmt**>
25. <**stmt**> → `return` <**expr**> <**next_expr**> <**stmt**>
26. <**stmt**> → `id` <**fork_id**> <**stmt**>
27. <**stmt**> → ε
28. <**def_var**> → `=` <**one_assign**>
29. <**def_var**> → ε
30. <**one_assign**> → `id (` <**param**> `)`

31. <one_assign> → <expr>
32. <param> → <param_val> <next_param>
33. <param> → ε
34. <param_val> → id
35. <param_val> → <term>
36. <term> → t_string
37. <term> → t_integer
38. <term> → t_number
39. <term> → nil
40. <next_param> → , <param_val> <next_param>
41. <next_param> → ε
42. <next_expr> → , <expr> <next_expr>
43. <next_expr> → ε
44. <fork_id> → (<param>)
45. <fork_id> → <next_id>
46. <next_id> → , id <next_id>
47. <next_id> → = <mult_assign>
48. <mult_assign> → id (<param>)
49. <mult_assign> → <expr> <next_expr>

LL – tabulka

	require	global	function	id	integer	string	number	nil	t_integer	t_number	t_string	if	while	local	return	:	=	(	,	EOF	$
<prolog>	1																				
<prog>		2	3	4																5	
<arg_T>					6	6	6	6													7
<next_arg_T>																			8		9
<ret_T>																10					11
<next_ret_T>																			12		13
<arg>				14																	15
<next_arg>																			16		17
<type>					18	20	19	21													
<stmt>				26								22	23	24	25						27
<def_var>																	28				29
<one_assign>				30																	31
<param>				32				32	32	32	32										33
<param_val>				34				35	35	35	35										
<term>								39	37	38	36										
<next_param>																			40		41
<next_expr>																			42		43
<fork_id>																	45	44	45		
<next_id>																	47		46		
<mult_assign>				48																	49

Precedenčná tabuľka

1. $\mathbf{E} \rightarrow \mathrm{i}$
2. $\mathbf{E} \rightarrow (\ \mathbf{E}\ )$
3. $\mathbf{E} \rightarrow \#\,\mathbf{E}$
4. $\mathbf{E} \rightarrow \mathbf{E} + \mathbf{E}$
5. $\mathbf{E} \rightarrow \mathbf{E} - \mathbf{E}$
6. $\mathbf{E} \rightarrow \mathbf{E} * \mathbf{E}$
7. $\mathbf{E} \rightarrow \mathbf{E} \,/\, \mathbf{E}$
8. $\mathbf{E} \rightarrow \mathbf{E} \,//\, \mathbf{E}$
9. $\mathbf{E} \rightarrow \mathbf{E} \,..\, \mathbf{E}$
10. $\mathbf{E} \rightarrow \mathbf{E} > \mathbf{E}$
11. $\mathbf{E} \rightarrow \mathbf{E} < \mathbf{E}$
12. $\mathbf{E} \rightarrow \mathbf{E} >= \mathbf{E}$
13. $\mathbf{E} \rightarrow \mathbf{E} <= \mathbf{E}$
14. $\mathbf{E} \rightarrow \mathbf{E} == \mathbf{E}$
15. $\mathbf{E} \rightarrow \mathbf{E} \sim= \mathbf{E}$

	#	*	/	//	+	-	..	<	<=	>	>=	==	~=	(	)	i	$
#	<	>	>	>	>	>	>	>	>	>	>	>	>	<	>	<	>
*	<	>	>	>	>	>	>	>	>	>	>	>	>	<	>	<	>
/	<	>	>	>	>	>	>	>	>	>	>	>	>	<	>	<	>
//	<	>	>	>	>	>	>	>	>	>	>	>	>	<	>	<	>
+	<	<	<	<	>	>	>	>	>	>	>	>	>	<	>	<	>
-	<	<	<	<	>	>	>	>	>	>	>	>	>	<	>	<	>
..	<	<	<	<	<	<	<	>	>	>	>	>	>	<	>	<	>
<	<	<	<	<	<	<	<	>	>	>	>	>	>	<	>	<	>
<=	<	<	<	<	<	<	<	>	>	>	>	>	>	<	>	<	>
>	<	<	<	<	<	<	<	>	>	>	>	>	>	<	>	<	>
>=	<	<	<	<	<	<	<	>	>	>	>	>	>	<	>	<	>
==	<	<	<	<	<	<	<	>	>	>	>	>	>	<	>	<	>
~=	<	<	<	<	<	<	<	>	>	>	>	>	>	<	>	<	>
(	<	<	<	<	<	<	<	<	<	<	<	<	<	<	=	<	e
)	>	>	>	>	>	>	>	>	>	>	>	>	>	e	>	s	>
i	e	>	>	>	>	>	>	>	>	>	>	>	>	e	>	s	>
$	<	<	<	<	<	<	<	<	<	<	<	<	<	<	e	<	e

LEGENDA:
< - insert to stack with shift
> - reduction
= - insert to stack
e - error
s - special case (end of expression)