Generování syntaktických analyzátorů nejen regulovaných gramatik

#18

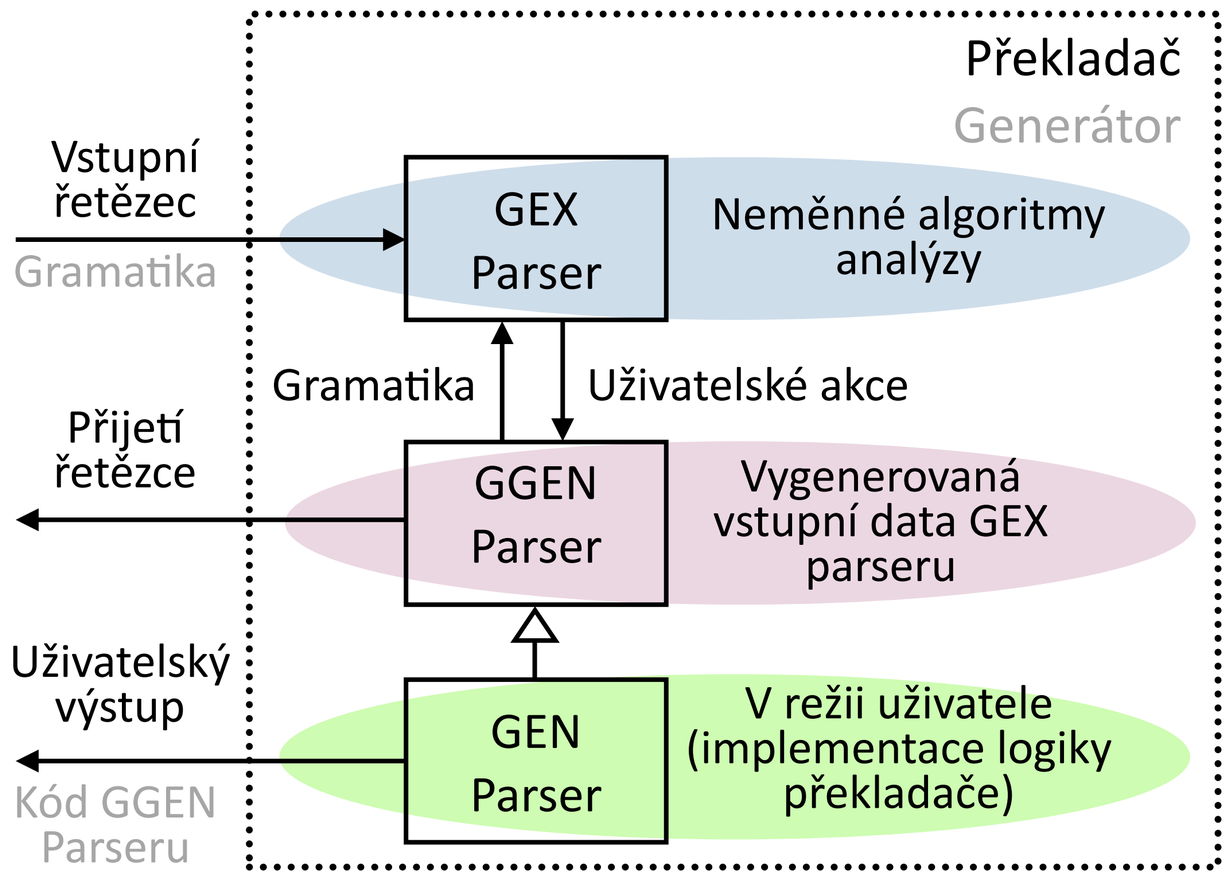
Motivace

- Nenáročná a intuitivní konstrukce překladačů
- Nezávislost popisu gramatiky na výstupním programovacím jazyce
- Podpora regulovaných gramatik
- Rychlý vývoj a prototypování

Řešení

- Jednoduchý definiční metajazyk pro popis podporovaných gramatik
- Popis gramatiky definičního metajazyka definičním metajazykem
- Překladač definičního metajazyka na uživatelské akce (*GEX Parser*)
- Překladač uživatelských akcí na kód vstupu GEX Parseru (*GEN Parser*)
- Zachování levého rozboru i při použití hlubokého zásobníku

Architektura



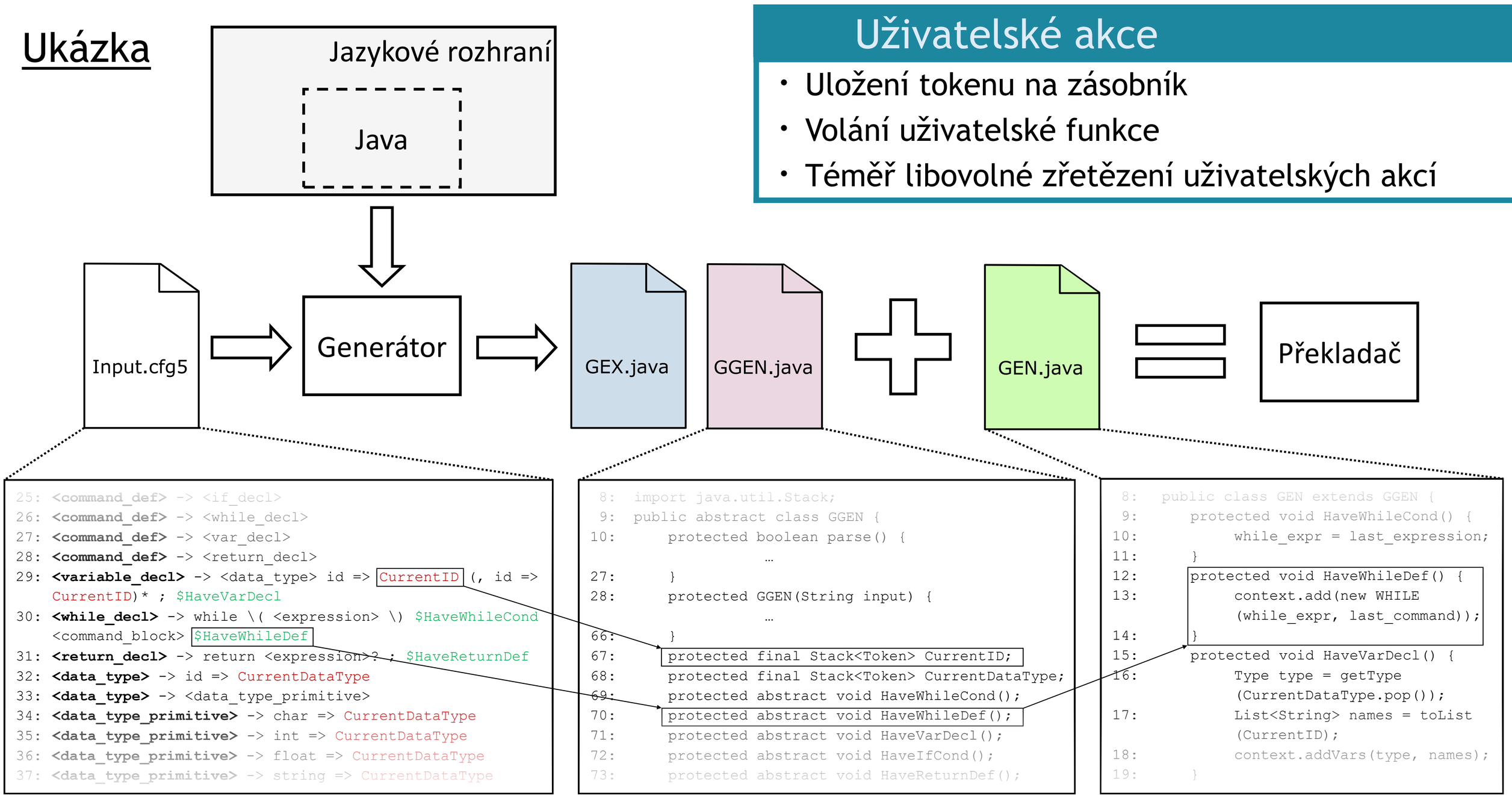
Podporované gramatiky

- LL(1) bezkontextové gramatiky
- Deterministické maticové gramatiky
- Deterministické programované gramatiky
- Deterministické gramatiky s nahodilým kontextem

Uživatelské akce

- Uložení tokenu na zásobník
- Volání uživatelské funkce
- Téměř libovolné zřetězení uživatelských akcí

Ukázka




```
25: <command_def> -> <if_decl>
26: <command_def> -> <while_decl>
27: <command_def> -> <var_decl>
28: <command_def> -> <return_decl>
29: <variable_decl> -> <data_type> id => CurrentID (, id =>
    CurrentID)* ; $HaveVarDecl
30: <while_decl> -> while \( <expression> \) $HaveWhileCond
    <command_block> $HaveWhileDef
31: <return_decl> -> return <expression>? ; $HaveReturnDef
32: <data_type> -> id => CurrentDataType
33: <data_type> -> <data_type_primitive>
34: <data_type_primitive> -> char => CurrentDataType
35: <data_type_primitive> -> int => CurrentDataType
36: <data_type_primitive> -> float => CurrentDataType
37: <data_type_primitive> -> string => CurrentDataType
```

```
 8: import java.util.Stack;
 9: public abstract class GGEN {
10:     protected boolean parse() {
                …
27:     }
28:     protected GGEN(String input) {
                …
66:     }
67:     protected final Stack<Token> CurrentID;
68:     protected final Stack<Token> CurrentDataType;
69:     protected abstract void HaveWhileCond();
70:     protected abstract void HaveWhileDef();
71:     protected abstract void HaveVarDecl();
72:     protected abstract void HaveIfCond();
73:     protected abstract void HaveReturnDef();
```

```
 8: public class GEN extends GGEN {
 9:     protected void HaveWhileCond() {
10:         while_expr = last_expression;
11:     }
12:     protected void HaveWhileDef() {
13:         context.add(new WHILE
            (while_expr, last_command));
14:     }
15:     protected void HaveVarDecl() {
16:         Type type = getType
            (CurrentDataType.pop());
17:         List<String> names = toList
            (CurrentID);
18:         context.addVars(type, names);
19:     }
```


Tomáš Dvořák
xznebe00@stud.fit.vutbr.cz

Excel@FIT 2019