GLL parsing for embedded languages

Author: Ragozina Anastasiya

Saint Petersburg State University
JetBrains Programming Languages and Tools Lab

25/10/2015

Problem statement

- Errors are detected in runtime
- IDEs do not provide support (highlighting, brace matching and etc.)
- It is necessary to get structure which merges all parse trees — SPPF

Generalised algorithms for embedded languages

- Ambiguous grammars are parsed by generalised algorithm (GLL, GLR)
- New type of conflict — ambiguities in the input
- Regular approximation of the input is represented as deterministic FSA with tokens on edges

GLL for embedded languages

- Table-based GLL parsing
- Descriptors specify parser state and allow to handle
  - Recursions
  - Ambiguities
  - Non-linear input
    - Vertex index is used as input position in descriptors
    - Branching in the input are handled in the same manner as grammar conflicts: the set of descriptors is created
  - Cycles in input
    - Uniqueness of descriptors allows to handle cycles without parsing process changes
- No changes in the process of GSS and SPPF construction

Branching in the input

- For each outgoing edge
  - Construct the set of descriptors (as in GLL)
- Union all the constructed sets
- Example
  - Grammar: `start ::= A C | B C`
  - Input:
  - Current vertex index is "0"
  - Construct two descriptors
    - For the edge labeled with "A" and grammar rule `start ::= A C`
    - For the edge labeled with "B" and grammar rule `start ::= B C`
  - During parsing process choose the edge which correspond to rule specified in current descriptor

Static analysis of string-embedded code: the scheme

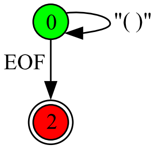
Code: hotspot is marked

```
string res = "";
for(i = 0; i < l; i++)
    res = "()" + res;
use(res);
```

Possible values `{"", "()", "()()", ..., "()"^l}`

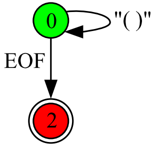
Regular approximation `("()")*`

Approximation

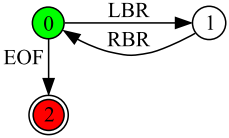


Static analysis of string-embedded code: the scheme

Approximation



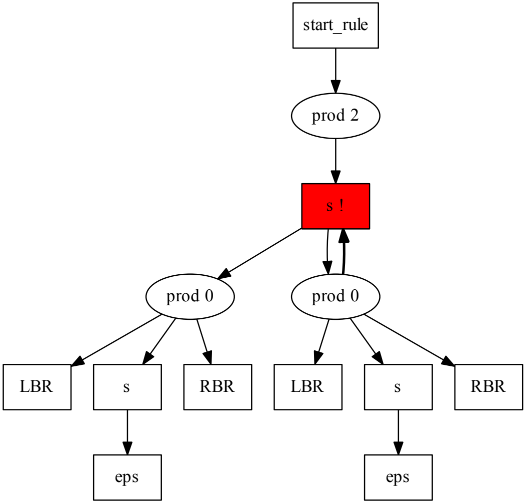
After lexing



Grammar

$$\begin{array}{rcl} \mathit{start} & ::= & s \\ s & ::= & \mathtt{LBR}\ s\ \mathtt{RBR}\ s \\ s & ::= & \epsilon \end{array}$$

Parse forest



Conclusion

- Algorithm based on GLL for parsing of regular approximation of string-embedded code is proposed
- Correctness and completeness of the algorithm are proved
- The algorithm is implemented and tested in open source project
    - `https://github.com/YaccConstructor`