Semantic Analyzer

xpojez00, xbudin05

November 22, 2021

1 Introduction

Whenever a rule is applied we need to check if semantic attributes match accordingly. Semantic analyzer should be called only upon expressions and operations involving them, i.e. assignment, return values, function calls etc. With creation of semantic action upon a rule applied we are closely tied to generating the final code, which can be done from Abstract Semantic Tree. Tree is generated from expressions currently.

2 Structure

```
NodeType = {OPERATION, ID, FUNCTION, VALUE, VOID}

SemanticType = {INTEGER, NUMBER, BOOLEAN, STRING, VOID}

struct Node {
    NodeType nodeType
    void *data
    vector *sons
    SemanticType semanticType
}
```

3 Code generation

When we want to generate code we just iterate in post-order to know, that all our sons are generated and they will return the variable they are saved in.

4 Possible Operators

Considering all operands must have the same semantic value, i.e. for example

$$INTEGER < NUMBER$$

is not valid.

Set of operators	Set of possible semantic values
$\{\#\}$	$\{STRING\}$
$\{\textbf{not}\}$	$\{BOOLEAN\}$
$\{*, /, +, -\}$	$\{INTEGER, NUMBER\}$
$\{//\}$	$\{INTEGER\}$
$\{..\}$	$\{STRING\}$
$\{>, <, >=, <=\}$	$\{INTEGER, NUMBER\}$
$\{==, \sim=\}$	$\{INTEGER, NUMBER, BOOLEAN\}$
$\{\textbf{and}, \textbf{or}\}$	$\{BOOLEAN\}$