4. Semantic Analysis I

Syntax Directed Definitions – Syntax Directed Translation Schemes

Andrea Polini, Luca Tesei

Formal Languages and Compilers
MSc in Computer Science
University of Camerino

ToC

Where we are?

So far we were able to check:

- the program includes correct “words”
- “words” are combined in correct “sentences”

What’s next?

- We would like to perform additional checks to increase guarantees of correctness
- We would like to transform the program from the source language into the target one, and according to precisely defined semantic rules

Where we are?

So far we were able to check:

- the program includes correct “words”
- “words” are combined in correct “sentences”

What’s next?

- We would like to perform additional checks to increase guarantees of correctness
- We would like to transform the program from the source language into the target one, and according to precisely defined semantic rules

Where we are?

So far we were able to check:

- the program includes correct “words”
- “words” are combined in correct “sentences”

What’s next?

- We would like to perform additional checks to increase guarantees of correctness
- We would like to transform the program from the source language into the target one, and according to precisely defined semantic rules

Additional checks

Additional Checks

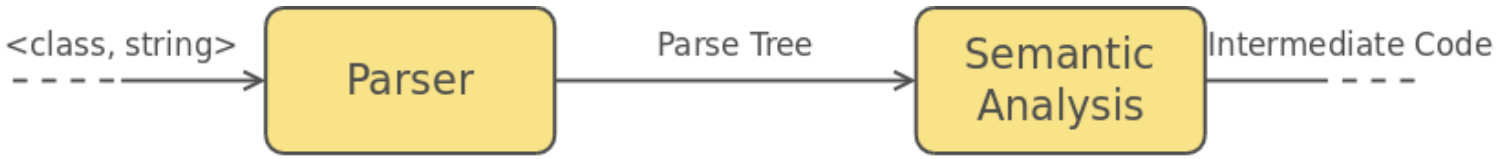
There are many additional checks that can be performed to increase correctness of code:

- Coherent usage of variables
  - definition-usage
  - type
- Existence of unreacheable code blocks
- ...

Semantic Analysis

In semantic analysis context sensitive analysis are performed without resurrecting to Context Sensitive grammar definitions. Here we focus on mechanisms for type checking and generation of intermediate code

Semantic analysis



ToC

Syntax Directed Definitions

Attributes

Attributes are used to associate characteristics and store values associated to grammar symbols.
A syntax directed definition provides the semantic rules to permit the definition of the values for the attributes

PRODUCTION	SEMANTIC RULE
$E \rightarrow E_1 + T$	$E.code = E_1.code \|\| T.code \|\| '+'$

- attributes are associated to grammar symbols and can be of any kind
- rules are associated to productions

Attributes

An SDD can be defined using two different kinds of attributes:

- Synthesized attributes: a synthesized attributes at node N is defined only in terms of attribute values at the children of N and at N itself
- Inherited attributes: an inherited attribute at node N is defined only in terms of attribute values at N's parent, N itself, and N's siblings

Attributes

Example

Consider the usual grammar and let's define a set of "reasonable" semantic rules:

$L \to E \quad E \to E + T \quad E \to T \quad T \to T * F \quad T \to F \quad F \to (E) \quad F \to id$

SDD and parse trees

An SDD with only synthesized attributes is called *S-attributed*

It is generally useful to represent attributes within parse trees. A parse tree showing the values of attributes is referred as an annotated parse tree

Order of evaluation for attributes

The order of evaluation of attributes should reflect the defined parsing strategy. In any case semantic rules impose an order of evalution that in case, inherited and synthetized attributes are present at the same time, is not guaranteed to exist.

Let's consider the expression "(3+4)*(5+6)" and let's derive its annotated parse tree from the semantic rules defined before

Inherited attributes example

Let's consider the non left recursive and factored grammar for expressions:
$E \to TE' \quad E' \to +TE'|\epsilon \quad T \to FT' \quad T' \to *FT'|\epsilon \quad F \to (E)|id$
define an SDD using as reference the parse tree for the sentence "$3 + 5 * 6$"

Evaluation Orders for SDD's

Dependency Graphs

A dependency graph represents the flow of information among the attribute instances in a particular parse tree.

- each attribute for a grammar symbol constitute a node in the graph
- syntesized attributes
- inherited attributes

Let's identify the dependency graph for the parse tree defined before, and let's compute the value of the various attributes

SDD with acyclic topological sort

S-attributed

If every attribute is synthesized the SDD is said S-attributed, in such a case an LR parser could even avoid the explicit derivation of the parse tree

L-attributed

Each attribute in the SDD satisfies one of the following conditions:

- it is synthesized
- it is inherited but it depends only from attributes on siblings on the left or inherited attributes associated to the parent symbol
- it is inherited or synthesized from attributes from the same symbol in a way that cycle are not generated

Semantic rules with controlled side effects

Side effects

A side effect consists of program fragment contained with semantic rules. It is necessary to control side effects is SDD in two possible ways:

- Permit incidental side effects
- Constraint admissible evaluation orders so to have the same translation with any admissible order.

Why to use them?

- to associate actions to carry on with specific steps of the compiler
- to print messages for the user useful during compilation
- to check correctness related aspects (e.g. types)

Semantic rules with controlled side effects

Side effects

A side effect consists of program fragment contained with semantic rules. It is necessary to control side effects is SDD in two possible ways:

- Permit incidental side effects
- Constraint admissible evaluation orders so to have the same translation with any admissible order.

Why to use them?

- to associate actions to carry on with specific steps of the compiler
- to print messages for the user useful during compilation
- to check correctness related aspects (e.g. types)