Zeyd uysal

zeyd.uysal@gmail.com

Abstract

Concepts to understand about Program Semantics to fully understand the working of ANTLP

**Algebraic Approaches to Program Semantics applied to ANTLP**

ANTLP Semantics research

Contents

[1. Introduction to denotational semantics 2](#_Toc70484851)

[1.1 Syntax and semantics 2](#_Toc70484852)

[Operational semantics 2](#_Toc70484853)

[Denotational semantics 2](#_Toc70484854)

[Assertion semantics 2](#_Toc70484855)

[partially additive semantics 3](#_Toc70484856)

# Introduction to denotational semantics

## Syntax and semantics

semantics: prescribe how a program is to be

interpreted as a sequence of instructions

### Operational semantics

1. "abstract computer" and describe how the programs run
2. traces all intermediate states in a computation
3. information on how to implement a programming language

### Denotational semantics

1. focuses on input/output behavior and ignores intermediate states
2. avoid worry about details of implementation
3. hard to invent a mathematical framework to describe repetitive constructs(loops) without intermediate states

### Assertion semantics

What is an assertion:

1. statement about the program state which is either true or false.
2. assertions shown enclosed by braces.
3. A program is correct if indeed the satisfaction of all initial assertions about the input data guarantees the truth of all assertions encountered later on.
4. Possible to create a programming language with assertions in mind

Example:

**INPUTS: X**

**OUTPUTS: Y**

**{X>= 0}**

**BEGIN**

**(a block of code representing an algorithm for Y := Sqrt(x)**

**END**

**{X = Y\*Y}**

benefits:

assertion documents the program and it would be possible to write software that could automatically scan the assertions to detect bugs and check for correctness.

### partially additive semantics

1. power-series representation for computed functions

expresses programming constructs in terms of operations that manipulate power series.