CS4022D Principles of Programming Languages
Lecture #7: Operational Semantics

Saleena N
NIT Calicut

August 11, 2021

Formal Semantics: Approaches

- Operational semantics
  - the meaning of a construct is specified by the computation it induces when it is executed on a machine.
  - interest is on how the effect of a computation is produced.
- Denotational semantics
  - meanings are modelled by mathematical objects that represent the effect of executing the constructs
  - only the effect is of interest, not how it is obtained
- Axiomatic semantics
  - specific properties of the effect of executing the constructs are expressed as assertions
  - there may be aspects of the executions that are ignored

Operational Semantics

- Structural Operational Semantics (small-step)[1] - specifies the individual steps of evaluation
- Natural Semantics (big-step)[2] - specifies the relationship between the initial configuration and the final configuration

[1] G. D. Plotkin *A Structural Approach to Operational Semantics*, Computer Science Department, Aarhus University, 1981

[2] G. Kahn *Natural Semantics*, 4^{th} Annual Symposium on Theoretical Aspects of Computer Science, 1987

Example Language 1

t::=		*terms*
	0	*constant zero*
	succ t	*successor*

Example Language 1

t::=		*terms*
	0	*constant zero*
	succ t	*successor*

List some strings in the language

Example Language 1

t::=		*terms*
	0	*constant zero*
	succ t	*successor*

- strings / sentences / terms in the language

  0, succ 0, succ succ 0, succ succ succ 0

Example Language 1

t::=		*terms*
	0	*constant zero*
	succ t	*successor*
	pred t	*predecessor*

0, succ 0, pred 0, succ pred 0, pred pred succ 0

Example Language 1

t::=		*terms*
	0	*constant zero*
	succ t	*successor*
	pred t	*predecessor*
	iszero t	*iszero*

0, succ 0, iszero pred 0, succ iszero pred 0

Example Language 2

t::=		*terms*
	true	*constant true*
	false	*constant false*
	if t then t else t	*conditional*

true
false
if true then false else true
if (if true then true else false) then false else true

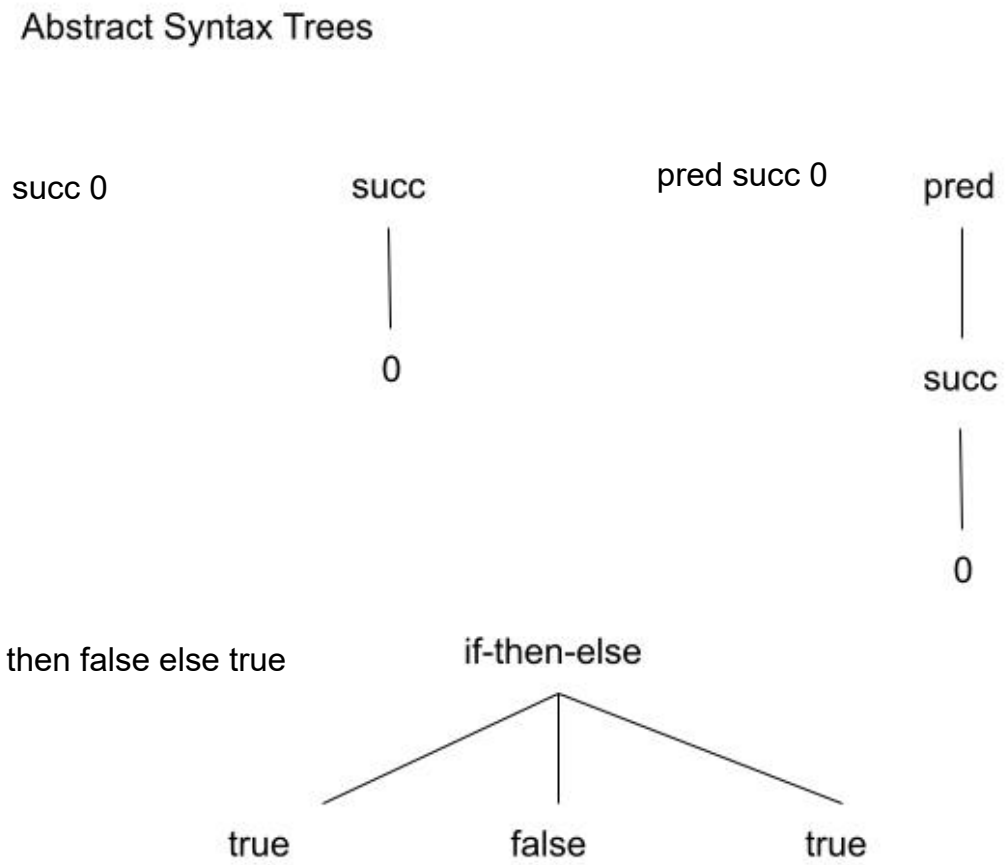
Language of Arithmetic Expressions

t::=		*terms*
	true	*constant true*
	false	*constant false*
	if t then t else t	*conditional*
	0	*constant zero*
	succ t	*successor*
	pred t	*predecessor*
	iszero t	*zero test*

Language of Arithmetic Expressions: Terms / Programs

iszero succ 0
if (*iszero* (*pred* 0)) *then true else false*
true
succ succ succ 0
pred succ pred succ 0

Terms: Abstract Syntax Tree Representation



Term: size

- $size(t)$: number of nodes in the AST representation of t
  - $size(0) = 1$
  - $size(succ\ succ\ 0) = 3$
  - $size(if\ true\ then\ 0\ else\ false) = ?$

Language of Booleans

t::=		*terms*
	true	*constant true*
	false	*constant false*
	if t then t else t	*conditional*

Language of Booleans: Values

v ::=

true

false