Today

Big Picture - Semantics
Judgments
Evaluation Order
Operational Semantics

Questions

Big Picture

What is the purpose of a language spec?

"To reduce ambiguity" between

User = programme + compiler = language implementation
Syntax
- strings
- abstract syntax trees

What you can write?

1 ≠ 2

Semantics (meaning)
What does what you write mean? “What does it evaluate to?”

1 + 2 → 42?
Operational Semantics

- Describe "how program expressions evaluate?"
  → in terms of the language itself

- Describe an interpreter for the language

Judgements

- Inductively-defined relations

Inductive definitions seen already

1. Stack inductive types
2. Grammar
   \[ e ::= n \mid e + e \]
sealed abstract class Nat

case object Zero extends Nat

case class Succ(n : Nat) extends Nat

Values of type Nat

\{ Zero, Succ(Zero), Succ(Succ(Zero)), ... \}  

Inductively-defined set:

"Least set closed under these rules"

Closed

If \( x \in X \), then apply a rule \( \text{Succ}(x) \in X \)

Zero \( \in X \)

\( X = \{ \text{junk} \), Zero, Succ(junk), Succ(Zero), Succ(Succ(junk), Succ(Succ(Zero)), ... \} \)

\( \rightarrow \) not least

\( \rightarrow \) (past one does not have "junk"

sealed abstract class Integer

case object Zero

case class Succ (i : Integer)

case class Pred (i : Integer)

\[ x : \{ \text{Zero, Succ(Zero), Pred(Succ(Zero)), } \ldots \} \]

Relations

Some statement between a set of objects

e : e'  (as an example)

e \rightarrow e'
Judgment is a statement between a set of objects

c : c

judgement form

Define them via inference rules

"if-then" premises

\[ J_1, J_2, \ldots, J_n \]

conclusion

\[ J \]

\[ n ::= 0 \mid s(n) \]

\[ \text{Nat} \]

\[ n \in \text{Nat} \]

\[ \text{axiom} \]

\[ 0 \in \text{Nat} \]

\[ s(n) \in \text{Nat} \]

\[ n \in \text{Nat} \]
Dernations

\[ \frac{0 \in \text{Nat} \quad \text{zero}}{\text{succ}} \]
\[ \frac{S(0) \in \text{Nat}}{\text{succ}} \]
\[ \frac{S(S(0)) \in \text{Nat}}{\text{succ}} \]

\[ L_{\text{mathnat}} \overset{\text{def}}{=} S_{\text{mathnat}}(0) \]

Theorem: For all \( \text{mathnat} \in \mathbb{N} \), we can construct a derivation for this judgment:

\[ L_{\text{mathnat}} \downarrow \overset{\text{Nat}}{\in} \]
Evaluation Order

# $

\[(a \# b) \# (c \$ d)\]

Syntactic concern of precedence + associativity

Semantic

Left first? \rightarrow\n
Evaluation order \rightarrow\n
Concurrently? \rightarrow\n
\[e_1 \# e_2\]

\[n_1 \# n_2\]

\[n_1 + n_2\]

\[\text{math}\]

Does \(\#\) matter?

Yes if there are side effects

No if not

"Referential transparency" (if I replace an expr by its value, then result doesn't change)
\[ e_1 \neq e_2 \]

\[ v_1 \neq e_2 \]

\[
\begin{align*}
(\text{print}(1), 1) & \rightarrow (\text{print}(2), 2) \\
1 & \rightarrow 1 \\
\text{side-effect}
\end{align*}
\]

Operational semantics

Defining \[ e \rightarrow e' \]

Judgement form

This is a relation between two expressions