Syntax vs. Semantics

<table>
<thead>
<tr>
<th>Syntactic</th>
<th>Semantic</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 = 1</td>
<td>1 = 1</td>
</tr>
<tr>
<td>x = 1</td>
<td>x = 1</td>
</tr>
<tr>
<td>2 + 2 = 4</td>
<td>2 + 2 = 4</td>
</tr>
<tr>
<td>1 + 3 = 2 + 2</td>
<td>1 + 3 = 2 + 2</td>
</tr>
</tbody>
</table>

Semantics

- **Semantics** = “Assigning Meaning to Programs” (Robert W. Floyd)
- **Operational Semantics**
  - Defining behavior of a programming language based on a simple abstract machine for it.
    - (E.g., state of the machine is a term, machine operates on a transition function)
- **Denotational Semantics**
  - Instead of describing a term as a sequence of machine states, the meaning of a term is taken to be some mathematical object
    - E.g., number or functions as semantic domains
- **Axiomatic Semantics**
  - Defining behavior of a language directly in terms of laws.

It was originally believed that operational semantics were inferior (< 1980)
- “quick and dirty”
- They have become more accepted due to concepts that showed to be difficult to express in the other forms.
  - Axiomatic semantics struggled with procedures
  - Denotational semantics struggled with non-determinism and concurrency
- More recent work (1980-2000) introduces more elegant formalism and managed to match many of the mathematical techniques developed in the context of denotational in operational semantics.
Java Try/Catch

- Assume you wanted to check if a monitor held within a try/catch block is automatically released if an exception occurs within the block...

Java-Style Try/Catch

```
Defining semantics is difficult!
```

Inference Rules in Sequent Calculus

- Inference: Act of drawing conclusions
- Notation: Gentzen
Example: Natural Deduction

\[
\begin{align*}
&\frac{A \land B}{A} \quad \frac{A \land B}{B} \quad \frac{A \land B}{(\land e)} \quad \frac{A \quad B}{A \to B} \quad \frac{A}{A \to B} \quad \frac{A}{(MP)} \\
&\frac{A \lor B}{A} \quad \frac{B \lor B}{B} \quad \frac{A \lor B}{A \to C \quad B \to C \quad (\lor e)} \quad \frac{A \to B \quad B \to A}{A \land \neg A \to B} \quad \frac{A \land \neg A \to B}{(\lor i)} \quad \frac{A \to \bot}{\bot} \quad \frac{A \to \bot}{(T \ i)} \\
&\frac{A \to \bot}{A \to \bot} \quad \frac{\neg A \land B}{A \quad (EFQ)} \quad \frac{\bot}{A \lor \neg A} \quad \frac{A \lor \neg A}{(TND)}
\end{align*}
\]

Soundness and Completeness

- **Soundness**: If \( K \vdash P \) then \( K \models P \) in \( T \).
  - If the conclusion is provable from the premise, then the premise logically entails the conclusion.

- **Completeness**: If \( K \models P \) then \( K \vdash P \) in \( T \).
  - If the premise logically entails the conclusion, then the conclusion is provable from the premise.

- A premise logically entails the conclusion iff every interpretation that satisfies the premise also satisfies the conclusion.

Semantics Revisited

- **Denotational semantics**:
  - \([TRUE]\) = true
  - \([FALSE]\) = false
  - \([AND(e_1, e_2)] = [e_1] \land [e_2]\)
Semantics Revisited

Small-Step Operational Semantics

\[
\begin{align*}
\text{AND}(\text{TRUE}, e) & \rightarrow e \quad (\text{AND1}) \\
\text{AND}(\text{FALSE}, e) & \rightarrow \text{FALSE} \quad (\text{AND2}) \\
\text{AND}(e_1, e_2) & \rightarrow \text{AND}(e_1', e_2) \quad (\text{AND3})
\end{align*}
\]

Big-Step Operational Semantics

\[
\begin{align*}
\text{eval}(\text{AND}(e_1, e_2)) \{ & \\
\text{if} \ (\text{eval}(e_1)) \ { & \text{return} \ \text{eval}(e_2); } \\
\text{else} \ { & \text{return} \ \text{false}; } \\
\}
\end{align*}
\]

eval(\text{AND}(e_1,e_2)) := \text{eval}(e_1) \ ? \ \text{eval}(e_2) : \text{false};

Term Algebra

Let \( \mathcal{F}^i \) be a set of function symbols with arity \( i \).
- We call functions with 0 arguments (nullary functions) constants.
- Let \( \mathcal{F} \) be the set of all function symbols: \( \mathcal{F} = \bigcup_{i=0}^\infty \mathcal{F}^i \)
- Let \( \mathcal{X} \) denote a set of (free) variable symbols.
- Given a set of function symbols \( \mathcal{F} \) and disjoint set of variable symbols \( \mathcal{X} \), the tuple \( \mathcal{T}(\mathcal{F}, \mathcal{X}) \) denotes a Term Algebra (or Term Language)

Example

\( \mathcal{F} = \{0, \text{succ}(t)\}, \mathcal{X} = \{x\} \)

\( 0 \in \mathcal{T}(\mathcal{F}, \mathcal{X}) \)
\( 1 \notin \mathcal{T}(\mathcal{F}, \mathcal{X}) \)
\( \text{succ}(0) \in \mathcal{T}(\mathcal{F}, \mathcal{X}) \)
\( \text{succ} (\text{succ}(0)) \in \mathcal{T}(\mathcal{F}, \mathcal{X}) \)
\( \text{succ}(x) \in \mathcal{T}(\mathcal{F}, \mathcal{X}) \)
\( \text{succ}(y) \notin \mathcal{T}(\mathcal{F}, \mathcal{X}) \)
Ground Terms

- A term $t \in T(F, \mathcal{X})$ containing no variables is called a *Ground Term* and we can write $t \in T(F)$.

- $0$
- $\text{succ}(0)$
- $\text{succ}(\text{succ}(0))$
- $\text{succ}(x)$