Syntax vs. Semantics

Equality:

<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 Language Specification

Exceptions: ~3200 words

The exception mechanism of the Java SE platform is integrated with its synchronization model (§17.1), so that monitors are unlocked as synchronized statements (§14.19) and invocations of synchronized methods (§14.3.6, §15.12) complete abruptly.

Inference Rules in Sequent Calculus

\[ \rho_1, \rho_2, \ldots, \rho_n \vdash \Gamma \quad \text{(rule name)} \]

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

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

Soundness and Completeness

- **Soundness**: If \( \vdash K \) 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 \( \vdash K \models P \) 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**:
  - \( [\text{TRUE}] = \text{true} \)
  - \( [\text{FALSE}] = \text{false} \)
  - \( [\text{AND}(e_1, e_2)] = [e_1] \land [e_2] \)
Semantics Revisited

- **Small-Step Operational Semantics**

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

**Example**

\[
\begin{align*}
\mathcal{F} &= \{0, \text{succ}(t)\}, \mathcal{X} = \{x\} \\
0 &\in \mathcal{T}(\mathcal{F}, \mathcal{X}) \\
1 &\not\in \mathcal{T}(\mathcal{F}, \mathcal{X}) \\
\text{succ}(0) &\in \mathcal{T}(\mathcal{F}, \mathcal{X}) \\
\text{succ}(\text{succ}(0)) &\in \mathcal{T}(\mathcal{F}, \mathcal{X}) \\
\text{succ}(x) &\in \mathcal{T}(\mathcal{F}, \mathcal{X}) \\
\text{succ}(y) &\not\in \mathcal{T}(\mathcal{F}, \mathcal{X})
\end{align*}
\]

eval(AND(e1,e2)) = eval(e1) ? eval(e2) : false;
Ground Terms
- A term $t \in \mathcal{I}(\mathcal{F}, \mathcal{X})$ containing no variables is called a
  *Ground Term* and we can write $t \in \mathcal{I}(\mathcal{F})$.
- $0$
- $\text{succ}(0)$
- $\text{succ}(\text{succ}(0))$
- $\text{succ}(x)$

Matching
- Given two terms $t_1 \in \mathcal{I}(\mathcal{F}, \mathcal{X})$ and $t_2 \in \mathcal{I}(\mathcal{F})$. The goal of matching is to decide whether the free variables in $t_1$
  can be bound in such a way that the new term $t'_1 = t_2$
  (syntactical equivalence).
- Example
  - $t_1$
  - $t_2$
  - matching binding
  - $x$ 0 $x = 0$
  - $x$ min(1, 2) $x = \text{min}(1, 2)$
  - $\text{min}(x, 4)$ $\text{min}(1, 4)$ $x = 1$
  - $\text{min}(x, y)$ $\text{min}(\text{succ}(0), 5)$ $x = \text{succ}(0)$ $y = 5$

Substitution
- $\sigma$ is the substitution of the form
  $$\{\text{var}_1 \rightarrow t_1 \} ... \{\text{var}_n \rightarrow t_n \}$$
  $$\sigma(f(t_1, t_2, ..., t_n)) = f(\sigma(t_1), \sigma(t_2), ..., \sigma(t_n))$$
  $$\sigma(\text{var}) = \sigma(t) \text{ if } \exists t : \{\text{var} \rightarrow t\} \in \sigma$$
  $$\sigma(\text{var}) = \text{var} \text{ if } \nexists t : \{\text{var} \rightarrow t\} \in \sigma$$
- Example:
  - $\sigma := \{x \rightarrow \text{succ}(0), y \rightarrow \text{min}(1, 2)\}$
    - $\text{min}(x, 1)$ $\text{min}(\text{succ}(0), 1)$
    - $\text{min}(x, y)$ $\text{min}(\text{succ}(0), \text{min}(1, 2))$
    - $\text{min}(z)$ $\text{min}(z)$

Equational Reasoning / Term Rewriting
- $$(s = t) \in E$$
  $$E \vdash s = t$$

- $$(E \vdash t = s)$$
  $$E \vdash s = t$$
  $$E \vdash t = u$$

- $$(E \vdash s = t)$$
  $$E \vdash \sigma(s) = \sigma(t)$$

- $$(E \vdash s_1 = t_1, ..., E \vdash s_n = t_n)$$
  $$E \vdash f(s_1, ..., s_n) = f(t_1, ..., t_n)$$
The Matching Problem for Ground Terms

Let

\[
\text{\text{match}}(t_{\text{vars}}, t_{\text{ground}}) = \begin{cases} \sigma & \text{if } \exists \sigma : \sigma(t_{\text{vars}}) = t_{\text{ground}} \\ \text{fail} & \text{if } \nexists \sigma \ldots \end{cases}
\]

be match expressions.

We can now perform equational reasoning on (simple) terms as a matching problem of the form

\[
\Theta = \{ \text{match}(t_1, s_1), \text{match}(t_2, s_2), \ldots, \text{match}(t_n, s_n) \}
\]

What about

\[
\min(x, y) = \min(succ(z), y) = \min(succ(0), y)
\]

Subterms!

Unification

Unification is a generalization of matching where both terms can contain variables

> Given two terms \( t_1 \in T(F, X) \) and \( t_2 \in T(F, X) \). The goal of unification is to discover a (most general) substitution \( \sigma \) such that \( \sigma(t_1) = \sigma(t_2) \)

> Most general unifier:

Let \( \Theta \) be a MGU for \( t_1 = t_2 \), then every other valid unifier \( \Theta' \) can be expressed as \( \Theta' = \Theta \sigma' \).

Example

\[
\min(x, y) \not\equiv \min(succ(0), y)
\]

\[
\{ \min(x, y) \not\equiv \min(succ(0), y) \}\]

\[
\{ x \equiv succ(0), y \equiv y \} \quad (1)
\]

\[
\{ x \equiv succ(0) \} \quad (3)
\]

\[
\text{done: } \Theta = \{ x \rightarrow succ(0) \}
\]

Martelli-Montanari Unification Algorithm

Let \( E \) be a set of pairs of terms. Repeatedly pick a pair and apply the applicable rule:

1. \( f(s_1, \ldots, s_n) \not\equiv f(t_1, \ldots, t_m) \)
   > replace by \( s_1 = t_1, \ldots, s_n = t_n \)
2. \( f(s_1, \ldots, s_n) \not\equiv g(t_1, \ldots, t_m) \land f \not\equiv g \)
   > fail
3. \( x \not\equiv x \)
   > delete the pair
4. \( t \not\equiv x, t \in X \)
   > replace by \( x = t \)
5. \( x \not\equiv t, x \in \text{Var}(t) \) and \( x \) occurs in some other pair
   > perform \( \sigma = \{ x \rightarrow t \} \)
6. \( x \not\equiv t, x \in \text{Var}(t) \) and \( x \neq t \)
   > fail
Example

- \( \text{foo}(f(x,1), g(y)) \Rightarrow \text{foo}(f(g(2),1), x) \)
  - \( \{\text{foo}(f(x,1), g(y)) \Rightarrow \text{foo}(f(g(2),1), x)\} \)
  - \( \{f(x,1) \equiv f(g(2),1), g(y) \equiv x\} \) (1)
  - \( \{x \equiv g(2), 1 \equiv 1, g(y) \equiv x\} \) (1)
  - \( \{x \equiv g(2), g(y) \equiv x\} \) (2)
  - \( \{g(y) \equiv g(2)\} \) (5) \( \sigma = \{x \rightarrow g(2)\} \)
  - \( \{y \equiv 2\} \) (1)

- done \( \Theta = \{x \rightarrow g(2), y \rightarrow 2\} \)

- Unification: \( \text{foo}(f(g(2),1), g(2)) \)