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
try {
    synchronized(x) {
        throw new Exception("boom");
    }
} catch (Exception e) {
    ...
}
```

Java-Style Try/Catch

```
(def x : volatile)
(decl : t : object)
(constr : t : object) (class Test
    (definit (defn add : (class Test) : test
        (arg x : int) (return x + 1)))
    (definit (defn TestException : throwable)
        (class TestException : throwable)
    )
    (class Test
        (definit (defn main : (class Test) : void)
            (arg x : int) (return x + 1)))
    )

Definition of semantics is difficult!
```

Inference Rules in Sequent Calculus

\[
\frac{\rho_1 \quad \rho_2 \quad \cdots \quad \rho_n}{\rho_1 \land \rho_2 \land \cdots} \rightarrow C
\]

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

\[
\frac{A \land B}{A} \quad \frac{A \land B}{B} \quad \frac{A}{A \rightarrow B} (\lor e) \quad \frac{A}{A} (\lor i) \quad (MP)
\]

\[
\frac{A \lor B}{A} \quad \frac{B}{A \lor B} (\lor i) \quad \frac{A \lor B}{A \rightarrow C \rightarrow C} (\lor e)
\]

\[
\frac{A 
\rightarrow B}{A} \quad \frac{B \rightarrow A}{A} (\rightarrow e) \quad \frac{A B}{A} \quad \frac{B \rightarrow A}{A} (\rightarrow i) \quad (T \lor)
\]

\[
\frac{\neg A}{A \rightarrow 1} \quad \frac{\neg A}{A} (\neg e) \quad \frac{\neg A}{\neg A} (\neg i) \quad \frac{A \lor \neg A}{\top} \quad (\land e)
\]

\[
\frac{A \land y}{x \land y} (\land i) \quad \frac{x \land y}{x \lor z} (\lor i) \quad \frac{y \land (x \lor z)}{y \land (x \lor z)} (\land e)
\]

Semantics Revisited

- **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

- **Small-Step Operational Semantics**
  
  \[ \text{AND}(\text{TRUE}, e) \rightarrow e \]  
  \[ \text{(AND1)} \]
  
  \[ \text{AND}(\text{FALSE}, e) \rightarrow \text{FALSE} \]  
  \[ \text{(AND2)} \]
  
  \[ \frac{e_1 \rightarrow e_1'}{\text{AND}(e_1, e_2) \rightarrow \text{AND}(e_1', e_2)} \]  
  \[ \text{(AND3)} \]

Example: Fractran

- **Program:**
  
  \[
  \begin{align*}
  17 & \quad 78 & \quad 19 & \quad 23 & \quad 29 & \quad 77 & \quad 95 & \quad 77 & \quad 1 & \quad 11 & \quad 15 & \quad 15 & \quad 55 \\
  (91 & \quad 85 & \quad 51 \cdot 38 & \quad 33 \cdot 29 & \quad 23 & \quad 19 & \quad 17 & \quad 13 & \quad 14 & \quad 2 & \quad 1)
  \end{align*}
  \]

- **Semantics:**
  
  Given the input number \( n \):
  
  - Find the first fraction \( f \) for which \( nf \) is an integer, replace \( n \) by \( nf \)
  - Repeat until no such \( f \) can be found, then halt.

- **Example: \( n=2 \)**
  
  \[
  \begin{align*}
  2, & \quad 15, & \quad 825, & \quad 725, & \quad 1925, & \quad 2275, & \quad 425, & \quad 390, & \quad 330, & \quad 290, & \quad 770, & \quad 910, & \quad 170, & \quad 156, & \quad 132, & \quad 116, & \quad 308, & \quad 364, & \quad 68, & \quad 4, & \quad 30, & \quad 825, & \quad 12275, & \quad 10875, & \quad 28875, & \quad 25375, & \quad 67375, & \quad 79625, & \quad 14875, & \quad 13650, & \quad 2550, & \quad 2340, & \quad 1980, & \quad 1740, & \quad 4620, & \quad 4060, & \quad 10780, & \quad 12740, & \quad 2380, & \quad 2184, & \quad 408, & \quad 152, & \quad ...
  \end{align*}
  \]
  
  (John Conway)

Example: Logic Circuits

**Model of Computation Semantics?**

\[
\begin{align*}
  d & := a \land b \\
  o & := d \lor c \\
\end{align*}
\]

(isomorph to straight line programs)

(isomorph to systems of equations)
Example: Term Rewriting

- \( x + 10 = 15 \)
  - \( x + 10 - 10 = 15 - 10 \)
  - \( x = 5 \)

- \( x + y = 10 \)
  - \( x \times 2 = 8 \)
  - \( x + y = 10 \)
  - \( x = 4 \)
  - \( 4 + y = 10 \)
    - \( x = 4 \)
    - \( y = 6 \)
    - \( x = 4 \)

Example

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

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

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)

Ground Terms

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

> Given two terms \( t_1 \in T(\mathcal{F}, X) \) and \( t_2 \in T(\mathcal{F}) \). 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**

\[
\begin{align*}
& t_1 \quad t_2 \\
& x \quad 0 \\
& x \quad \min(1, 2) \\
& \min(x, 4) \quad \min(1, 4) \\
& \min(x, y) \quad \min(\text{succ}(0), 5)
\end{align*}
\]

Equational Reasoning / Term Rewriting

\[
\frac{(s = t) \in E \quad E \vdash s \equiv t}{E \vdash t \equiv t}
\]

\[
\frac{E \vdash t = s \quad E \vdash s = t \quad E \vdash t = u}{E \vdash t = u}
\]

\[
\frac{E \vdash s = t}{E \vdash \sigma(s) = \sigma(t)}
\]

\[
\frac{E \vdash t_1 = t_n \quad E \vdash s_1 = t_1 \ldots E \vdash s_n = t_n}{E \vdash \sigma(t_1, \ldots, t_n) = \sigma(s_1, \ldots, s_n)}
\]

Substitution

> \( \sigma \) is the substitution of the form 

\[
\{ \text{var}_1 \mapsto t_1 \} \ldots \{ \text{var}_n \mapsto t_n \}
\]

\[
\sigma(f(t_1, t_2, \ldots, t_n)) = f(\sigma(t_1), \sigma(t_2), \ldots, \sigma(t_n))
\]

\[
\sigma(\text{var}) = \sigma(t) \quad \text{if} \quad \exists t : \{ \text{var} \mapsto t \} \in \sigma
\]

\[
\sigma(\text{var}) = \text{var} \quad \text{if} \quad \nexists t : \{ \text{var} \mapsto t \} \in \sigma
\]

**Example:**

\[
\begin{align*}
& \sigma := \{ x \mapsto \text{succ}(0) \}, \{ y \mapsto \min(1, 2) \} \\
& \min(x, 1) \quad \min(\text{succ}(0), \min(1, 2)) \\
& \min(x, y) \quad \min(\text{succ}(0), \min(1, 2)) \\
& \min(z) \quad \min(z)
\end{align*}
\]

The Matching Problem for Ground Terms

> Let \( \text{match}(t_{\text{vars}}, t_{\text{ground}}) = \)

\[
\begin{cases}
\{ \sigma \quad \text{if} \quad \exists \sigma : \sigma(t_{\text{vars}}) = t_{\text{ground}} \\
\text{fail} \quad \text{if} \quad \nexists \sigma \quad \text{be match expressions.}
\end{cases}
\]

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 \( \min(x, y) = \min(\text{succ}(z), y) \approx \min(\text{succ}(0), y) \) 

> Subterms!
Unification

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

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

Most general unifier:

1. $\theta = \{\text{unifies}(t_1, s_1), \text{unifies}(t_2, s_2), \ldots \}$
2. $\theta$ be a MGU for $t_1 = t_2$, then every other valid unifier $\theta'$ can be expressed as $\theta' = \theta \sigma'$.

Example

1. $\text{min}(x, y) \equiv \text{min}(\text{succ}(0), y)\{\text{min}(x, y) \equiv \text{min}(\text{succ}(0), y)\}$
2. $\{x \equiv \text{succ}(0), y \equiv y\}$
3. $\{x \equiv \text{succ}(0)\}$
4. $\text{done: } \theta = \{x \mapsto \text{succ}(0)\}$

1. $\text{min}(x, y) \equiv \text{min}(\text{succ}(x), y)\{\text{min}(x, y) \equiv \text{min}(\text{succ}(x), y)\}$
2. $\{x \equiv \text{succ}(x), y \equiv y\}$
3. $\{x \equiv \text{succ}(x), y \equiv y\}$
4. $\{x \equiv \text{succ}(x)\}$
5. $\text{fail}$

Martelli-Montanari Unification Algorithm

1. Let $E$ be a set of pairs of terms. Repeatedly pick a pair and apply the applicable rule:
2. $f(s_1, \ldots, s_n) \equiv f(t_1, \ldots, t_m)$
   - => replace by $s_1 = t_1, \ldots, s_n = t_n$
3. $f(s_1, \ldots, s_n) \equiv g(t_1, \ldots, t_m)$, $f \neq g$
   - => fail
4. $x \equiv x$
   - => delete the pair
5. $t \equiv x$, $x \notin \mathcal{X}$
   - => replace by $x = t$
5. $x \equiv t, x \notin \mathcal{Var}(t)$ and $x$ occurs in some other pair
   - => perform $\sigma = (x \mapsto t)$
6. $x \equiv t$, $x \notin \mathcal{Var}(t)$ and $x \neq t$
   - => fail

Example

1. $\text{foo}(f(x, 1), g(y)) \equiv \text{foo}(f(\text{g}(2), 1), x)$
2. $\{\text{foo}(f(x, 1), g(y)) \equiv \text{foo}(f(\text{g}(2), 1), x)\}$
3. $\{f(x, 1) \equiv f(\text{g}(2), 1), g(y) \equiv x\}$
4. $\{x \equiv \text{g}(2), 1 \equiv 1, g(y) \equiv x\}$
5. $\{x \equiv \text{g}(2), g(y) \equiv x\}$
6. $\{g(y) \equiv \text{g}(2)\}$
7. $\{\{y \equiv 2\}$
8. $\text{done } \theta = \{x \mapsto \text{g}(2), y \mapsto 2\}$

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