### Applicative / Declarative / Functional Programming Language Semantics

**Example Programs in Standard ML:**

<table>
<thead>
<tr>
<th>Program</th>
<th>Evaluation Strategy</th>
<th>Result</th>
</tr>
</thead>
</table>
| `fun fac(x) = if x=0 then 1 else x*fac(x-1);` | call-by-value | x
| `fun fib(x) = if x=0 then 1 else if x=1 then 1 else fib(x-1)+fib(x-2);` | call-by-value | fib(x-1)+fib(x-2)
| `fun gcd(x,y) = if y=0 then x else gcd(y,modulo(x,y));` | call-by-name | gcd(y,modulo(x,y))
| `fun f(x) = f(x)+1;` | call-by-name | f(x)+1

**Question:** What does \( g(f(0)) \) give?

**Answer:** It depends...

- **call-by-value** evaluation strategy: no answer.
- **call-by-name** evaluation strategy: result is 1.

---

### The Syntax of the Language Rec

#### Syntactic Domains

- \( n, m \in \mathbb{N} = \{ \ldots, -2, -1, 0, 1, 2, \ldots \} \) (numbers, integers)
- \( x, y \in \text{Var} = \{ x, y, z, \ldots \} \) (variables)
- \( f_1 \in \text{FVar} = \{ f_1, f_2, \ldots, f_k \} \) (function variables)
- \( t \in \text{Rec} \) (terms)

#### The Language

<table>
<thead>
<tr>
<th>Expression</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>t ::= n</code></td>
<td>test ( t = n ) for zero</td>
</tr>
<tr>
<td>`</td>
<td>`</td>
</tr>
</tbody>
</table>

Function variables are defined by a **declaration** \( d \):

\[
\begin{align*}
    f_1(x_1, \ldots, x_{a_1}) &= d_1 \\
    \vdots \\
    f_k(x_1, \ldots, x_{a_k}) &= d_k
\end{align*}
\]

**Note:** Despite its simplicity, \( \text{Rec} \) is fully expressive: any computable function can be written as a \( \text{Rec} \) program (i.e., a term \( t \) and an associated declaration \( d \)).

---

### Call-by-Value Operational Semantics

**Evaluation Rules:**

- \( t \rightarrow_{av} d \)
- \( t_1 \rightarrow_{av} d_1, t_2 \rightarrow_{av} d_2 \)
- \( t_1 \rightarrow_{av} d_1, t_2 \rightarrow_{av} d_2, t_3 \rightarrow_{av} d_3 \)
- \( t \rightarrow_{av} d_0 \)
- \( t_1 \rightarrow_{av} d_1, t_2 \rightarrow_{av} d_2, n \neq 0 \)
- \( t_1 \rightarrow_{av} d_1, \ldots, t_n \rightarrow_{av} d_n, d_{n+1} \rightarrow_{av} d_{n+1} \)

**Theorem:** If \( t \rightarrow_{av} d_1 \) and \( t \rightarrow_{av} d_2 \) then \( n_1 = n_2 \).  

**Proof:** By induction on the depth of inference of \( t \rightarrow_{av} d_1 \).
Denotation of Declarations

**Theorem:** $\llbracket t \rrbracket_{va}$ is continuous.

**Proof:** $t$ is expressed in our Metalanguage for Continuous Functions.

The function environment $\delta = \{\delta_1, \ldots, \delta_k\}$ determined by the declaration

\[
\begin{align*}
\delta_1(x_1, \ldots, x_n) &= d_1 \\
\vdots \\
\delta_k(x_1, \ldots, x_n) &= d_k
\end{align*}
\]

is defined as

\[
\delta = \text{fix } F = \bigsqcup_{\eta \in \omega} F^n(\bot),
\]

where $F : \text{Fenv}_{va} \to \text{Fenv}_{va}$ is given by

\[
F(\varphi)(n_1, \ldots, n_m) = [d_1, \varphi, \rho_{n_1 + n_2, \ldots, n_m + n_2}]
\]

**Theorem:** For closed $t$, $t \vdash_{va}^d n$ iff $[t]_{va} \delta \rho = [n]$.

---

Call-by-Name Operational Semantics

**Evaluation Rules:**

\[
\begin{align*}
|n \vdash_{na}^d n| \\
|t_1 \vdash_{na}^d n_1, t_2 \vdash_{na}^d n_2| \\
|t_1 \vdash_{na}^d n_1, t_2 \vdash_{na}^d n_2| \\
|t_1 \vdash_{na}^d n_1, t_2 \vdash_{na}^d n_2|
\end{align*}
\]

**Theorem:** If $t \vdash_{na}^d n_1$ and $t \vdash_{na}^d n_2$ then $n_1 = n_2$.

**Proof:** By induction on the depth of inference of $t \vdash_{na}^d n_1$.

**Theorem:** If $t \vdash_{na}^d n$ then $t \vdash_{na}^d n$.

(The reverse implication is not true in general.)

---

Call-by-Name Denotational Semantics

**Variable Environment** $\text{Env}_{na}$ and

**Function Variable Environment** $\text{Fenv}_{na}$

\[
\begin{align*}
\rho &\in \text{Env}_{na} = \{\text{Var} \to Z_\bot\} \\
\varphi &\in \text{Fenv}_{na} = [Z_\bot \to Z_\bot] \times \cdots \times [Z_\bot \to Z_\bot]
\end{align*}
\]

\[
[t]_{na} \in \text{Fenv}_{na} \to \text{Env}_{na} \to Z_\bot
\]

**Theorem:** $\llbracket t \rrbracket_{na}$ is continuous.

**Proof:** $t$ is expressed in our Metalanguage for Continuous Functions.

The function environment $\delta = \{\delta_1, \ldots, \delta_k\}$ determined by the declaration

\[
\begin{align*}
\delta_1(x_1, \ldots, x_n) &= d_1 \\
\vdots \\
\delta_k(x_1, \ldots, x_n) &= d_k
\end{align*}
\]

is defined as

\[
\delta = \text{fix } F = \bigsqcup_{\eta \in \omega} F^n(\bot),
\]

where $F : \text{Fenv}_{na} \to \text{Fenv}_{na}$ is given by

\[
F(\varphi)(n_1, \ldots, n_m) = [d_1, \varphi, \rho_{n_1 + n_2, \ldots, n_m + n_2}]
\]

**Theorem:** For closed $t$, $t \vdash_{na}^d n$ iff $[t]_{na} \delta \rho = [n]$.