A small language

Expressions

\[ a ::= n \mid X \mid a + a \]

Environments are functions from variables to integers, they are denoted by \( \sigma \).

We want to evaluate an expression \( a \) w.r.t. an environment \( \sigma \).

Defining an Operational Semantics

- Granularity
- Order of evaluation

Big-step Semantics

Each rule completely evaluates an expression w.r.t. an environment to a value.

\[
\begin{align*}
\langle n, \sigma \rangle &\Downarrow n \\
\langle X, \sigma \rangle &\Downarrow \sigma(X)
\end{align*}
\]

\[
\begin{align*}
\langle a_1, \sigma \rangle &\Downarrow n_1 \\
\langle a_2, \sigma \rangle &\Downarrow n_2
\Rightarrow \\
\langle a_1 + a_2, \sigma \rangle &\Downarrow n
\end{align*}
\]

where \( n \) is the sum of \( n_1 \) and \( n_2 \)
Properties

- Abstract
- Allows to avoid details.
- No specification of evaluation order (e.g. \((1 + 3) + (5 - 3)\)).
- No specification of control of errors.
- No specification of interleaving.

Small-step Semantics

Describes evaluation as a sequence of state changes of an abstract machine. Evaluation terminates when the state cannot be reduced further.

\[
\langle X, \sigma \rangle \leadsto \langle \sigma(X), \sigma \rangle \quad n \text{ is the sum of } n_1 \text{ and } n_2
\]

\[
\langle a_1, \sigma \rangle \leadsto \langle a'_1, \sigma' \rangle \quad \langle n_1 + n_2, \sigma \rangle \leadsto \langle n, \sigma \rangle
\]

\[
\langle a_1 + a_2, \sigma \rangle \leadsto \langle a'_1 + a_2, \sigma' \rangle \quad \langle a_2, \sigma \rangle \leadsto \langle a'_2, \sigma' \rangle
\]

\[
\langle a, \sigma \rangle \leadsto \langle \sigma', \sigma \rangle
\]

Properties

- Less abstract
- Specification of order of evaluation
- Control of errors: \(\frac{n_2 \neq 0}{n_1/n_2 \leadsto n}\), where \(n\) is \(n_1\) divided by \(n_2\).
- Interleaving: \(\langle c_1, \sigma \rangle \leadsto \langle c'_1, \sigma' \rangle \quad \langle c_1 \| c_2, \sigma \rangle \leadsto \langle c'_1 \| c_2, \sigma' \rangle\)

From Small-step to Multi-step Semantics

Notation: \(t \leadsto^* t'\)

- \(t \leadsto^* t\) for every \(t\)
- \(t \leadsto t'\) implies \(t \leadsto^* t'\)
- \(t \leadsto^* t'\) and \(t' \leadsto^* t''\) implies \(t \leadsto^* t''\)
Normal Forms

- A normal form is a term that cannot be evaluated any further.
- A normal form is a state where the abstract machine is halted (result of the evaluation).
- The meaning of a term $t$ in a small-step semantics is a term $t'$ such that $t \leadsto^* t'$ and $t'$ is a normal form.

Big-step versus Small-step Semantics

- In small-step semantics evaluation stops at errors. In big-step semantics errors occur deeply inside derivation trees.
- The order of evaluation is explicit in small-step semantics but implicit in big-step semantics.
- Big-step semantics is more abstract, but less precise.
- Small-step semantics allows to make difference between non-termination and "getting stuck".

Properties of the Small-step Semantics

- $t \Downarrow v$ iff $t \leadsto^* v$
- The relation $\leadsto$ is deterministic.

A functional language

Expressions/programs are closed terms.
Values are closed terms of the form $\lambda x.M$.

Reduction Strategies are deterministic subrelations of $\rightarrow_\beta$.

We study two different reduction strategies: call-by-name and call-by-value.

$M \Downarrow_v V$ : big-step semantics for call-by-value
$M \Downarrow_n V$ : big-step semantics for call-by-name
$M \leadsto_v N$ : small-step semantics for call-by-value
$M \leadsto_n N$ : small-step semantics for call-by-name
Call-by-value lambda-calculus (big-step semantics)

\[ \lambda x. M \Downarrow_v \lambda x. M \]

\[ M \Downarrow_v \lambda x. L \quad N \Downarrow_v V_2 \quad L\{x/V_2\} \Downarrow_v V_1 \]

\[ M N \Downarrow_v V_1 \]

Call-by-value lambda calculus (small-step semantics)

\[ (\lambda x. M) V \rightsquigarrow_v M\{x/V\} \]

\[ M \rightsquigarrow_v M' \]

\[ N \rightsquigarrow_v N' \]

\[ M N \rightsquigarrow_v M' N \]

\[ V N \rightsquigarrow_v V N' \]

Examples

Let \( \Delta = \lambda x.xx \). Then,
- \((\lambda x.x) (I I) \rightsquigarrow_v (\lambda x.x x) I \rightsquigarrow_v I I \rightsquigarrow_v I\).
- \((\lambda x.I) (\Delta \Delta) \rightsquigarrow_v (\lambda x.I) (\Delta \Delta) \rightsquigarrow_v \ldots\)
- \(\Delta \Delta \rightsquigarrow_v \Delta \Delta \rightsquigarrow_v \Delta \Delta \ldots\)

Call-by-name lambda-calculus (big-step semantics)

\[ \lambda x. M \Downarrow_n \lambda x. M \]

\[ M \Downarrow_n \lambda x. L \quad L\{x/N\} \Downarrow_n V \]

\[ M N \Downarrow_n V \]
Call-by-name lambda calculus (small-step semantics)

\[ (\lambda x. M) N \leadsto_n M \{ x/N \} \]

\[ M \leadsto_n M' \]

\[ M N \leadsto_n M' N \]

Examples

- \((\lambda x. x) (I I) \leadsto_n (I I) (I I) \leadsto_n I I (I I) \leadsto_n I I \leadsto_n I.\)
- \((\lambda x. I) (\Delta \Delta) \leadsto_n I.\)
- \(\Delta \Delta \leadsto_n \Delta \Delta \leadsto_n \Delta \Delta \Delta \ldots.\)

Deterministic properties

- If \( M \downarrow_v N \) and \( M \downarrow_v N' \), then \( V = V' \).
- If \( M \downarrow_n P \) and \( M \downarrow_n P' \), then \( P = P' \).
- If \( M \leadsto_v N \) and \( M \leadsto_v N' \), then \( N = N' \).
- If \( M \leadsto_n N \) and \( M \leadsto_n N' \), then \( N = N' \).
Relating big and small-steps semantics (i)

**Lemma:** If \( M \downarrow^* V \), then \( M \sim^*_v V \).

**Proof.** By induction on \( M \downarrow^* V \).
- If \( M = \lambda x.K \downarrow^* \lambda x.K = V \), then \( M \sim^*_v V \) trivially holds.
- If \( M = M_1 M_2 \downarrow^* V \) comes from \( M_1 \downarrow^* \lambda x.K, M_2 \downarrow^* W \) and \( K \{x/W\} \downarrow^* V \), then \( M_1 \sim^*_v \lambda x.K, M_2 \sim^*_v W \) and \( K \{x/W\} \sim^*_v V \) hold by the inductive hypothesis so that we construct the following small-steps reduction sequence:

\[
M = M_1 M_2 \sim^*_v (\lambda x.K) M_2 \sim^*_v (\lambda x.K) W \sim^*_v K \{x/W\} \sim^*_v V
\]

Taking \( n_1 = k_1 + 1 \), \( n_2 = k_2 \) and \( n_3 = k_3 \) we conclude

\[
M = TU \sim^*_v (\lambda x.K) U \sim^*_v (\lambda x.K) W \sim^*_v K \{x/W\} \sim^*_v N
\]

2. \( M = (\lambda x.K) U \sim^*_v (\lambda x.K) U' \sim^*_v U' \sim^* \sim^* N \), where \( U \sim^* U' \).

Since \( (\lambda x.K) U' \) is not a value we can apply the i.h. Thus

\[
(\lambda x.K) U' \sim^*_v (\lambda x.K) U' \sim^*_v (\lambda x.K) W \sim^*_v K \{x/W\} \sim^*_v N
\]

Taking \( n_1 = 0 \), \( n_2 = k_2 + 1 \) and \( n_3 = k_3 \) we conclude

\[
M = (\lambda x.K) U \sim^*_v (\lambda x.K) U \sim^*_v (\lambda x.K) W \sim^*_v K \{x/W\} \sim^*_v N
\]

3. \( M = (\lambda x.K) W \sim^*_v K \{x/W\} \sim^*_v N \).

We conclude with \( T = \lambda x.K, U = W, n_1 = n_2 = 0 \) and \( n_3 = n - 1 \).

Relating big and small-steps semantics (ii)

**Lemma:** If \( M \sim^*_v N \) in \( n \) steps, \( M \) is not a value and \( N \) is a value, then \( M \) is an application \( TU \) and \( \exists n_1, n_2, n_3 < n \) such that

\[
M \sim^*_v (\lambda x.K) U \sim^*_v (\lambda x.K) W \sim^*_v K \{x/W\} \sim^*_v N
\]

**Proof.** Suppose \( M \sim^*_v N \) in \( n \) steps. We reason by induction on \( n \).
- If \( n = 0 \), then \( M = N \) and thus \( M \) is a value. The property holds because the hypothesis is false.
- If \( n > 0 \), then there are three cases.

1. \( M = TU \sim^*_v T' U \sim^*_v N \), where \( T \sim^*_v T' \).

Since \( T' U \) is not a value we can apply the i.h. Thus

\[
T' U \sim^*_v (\lambda x.K) U \sim^*_v (\lambda x.K) W \sim^*_v K \{x/W\} \sim^*_v N
\]

Relating big and small-steps semantics (iii)

**Lemma:** If \( M \sim^*_v N \) and \( N \) is a value, then \( M \downarrow^* N \).

**Proof.** Suppose \( M \sim^*_v N \) in \( n \) steps. We reason by induction on \( n \).
- If \( n = 0 \), then \( M = N \). But \( N = \lambda x.K \) since \( N \) is a value so that \( M = \lambda x.K \downarrow^* \lambda x.K = N \).
- If \( n > 0 \), then \( M \) is not a value, so that by previous Lemma

\[
M = TU \sim^*_v (\lambda x.K) U \sim^*_v (\lambda x.K) W \sim^*_v K \{x/W\} \sim^*_v N
\]

for \( n_1, n_2, n_3 < n \). By the i.h. \( T \downarrow^* \lambda x.K \) and \( U \downarrow^* W \) and \( K \{x/W\} \downarrow^* N \), so that we conclude \( M \downarrow^* N \).
Relating big and small-steps semantics (iv)

- If $M \Downarrow_n P$, then $M \prec_n^* P$.
- If $M \prec_n^* N$ and $N$ is a value, then $M \Downarrow_n N$.

Progress properties

Let $M$ be a closed term which is not still a value. Then,

- There exist $N$ such that $M \prec_v N$.
- There exist $N$ such that $M \prec_n N$.

Standardisation

A standard reduction sequence never reduces right redexes before left redexes.

The following reduction sequences are not standard

$$(\lambda x. II)(II) \rightarrow (\lambda x. II)I \rightarrow II \rightarrow I$$

$$(\lambda x. II)(II) \rightarrow (\lambda x.II) \rightarrow I$$

The following reduction sequence is standard

$$(\lambda x.II)(II) \rightarrow II \rightarrow I$$

The Standardisation Theorem

Theorem: If $t \rightarrow_\beta^* t'$, then there is a standard reduction sequence from $t$ to $t'$.

Subtle point:

$$(\lambda x.(II))(II) \rightarrow (\lambda x.I)(II) \rightarrow (\lambda x.I)I$$

is standard (there is no violation of the left-to-right order)

$$(\lambda x.(II))(II) \rightarrow (\lambda x.I)(II) \rightarrow (\lambda x.I)I \rightarrow I$$

is not standard (there is violation of the left-to-right order).
Perpetual Strategies

Given any reduction step \( t \rightarrow_\beta t' \):
- If \( t \in SN_\beta \), then \( t' \in SN_\beta \)
- If \( t \notin SN_\beta \), then \( t' \) is not necessarily in \( SN_\beta \)

A reduction strategy \( \rightsquigarrow \) for the \( \lambda \)-calculus is perpetual iff \( t \rightsquigarrow t' \) and \( t \notin SN_\beta \) implies \( t' \notin SN_\beta \) (equiv. \( t' \in SN_\beta \) implies \( t \in SN_\beta \)).

Call-by-name is not perpetual: \((\lambda x. I)(\Delta \Delta) \rightsquigarrow_n I \) and \((\lambda x. I)(\Delta \Delta) \notin SN_\beta \) but \( I \in SN_\beta \).

A Perpetual Strategy for the \( \lambda \)-calculus

\[
\begin{align*}
  t_i \rightsquigarrow t'_i & \& t_1 \ldots t_{i-1} \in NF_\beta \\
  xt_1 \ldots t_i \ldots t_n \rightsquigarrow xt_1 \ldots t'_i \ldots t_n & \\
  \lambda x. u \rightsquigarrow \lambda x. u' & \\
  u \rightsquigarrow u' \text{ and } x \notinfv (t) & \\
  u \in NF_\beta \text{ or } x \infv (t) & \\
  (\lambda x. t) u t_1 \ldots t_n \rightsquigarrow (\lambda x. t) u' t_1 \ldots t_n & \\
  (\lambda x. t) u t_1 \ldots t_n \rightsquigarrow t \{ x/u \} t_1 .
\end{align*}
\]

Example: \((\lambda x. I)(\Delta \Delta) \rightsquigarrow (\lambda x. I)(\Delta \Delta) \).

Perpetuality Theorem

The reduction strategy of the previous slide is perpetual, i.e. if \( t \rightsquigarrow t' \) and \( t \notin SN_\beta \), then \( t' \notin SN_\beta \).