Operational Semantics

Viktor Kunčak

Amyli language

Tiny functional language that supports recursive functions.
Works only on integers and booleans.

(Initial) program is a pair (e_{top}, t_{top}) where

- e_{top} is the top-level environment mapping function names to function definitions
- t_{top} is the top-level term (expression) that starts execution

Function definition for a given function name is a tuple of: parameter list $\bar{x}$, parameter types $\bar{\tau}$, expression representing function body t, and result type τ_0.

Expressions are formed by invoking primitive functions $(+, -, \leq, \&\&)$, invocations of defined functions, or **if** expressions.
No local **val** definitions nor **match**. e will remain fixed

Amyli: abstract syntax of terms

$$t := true \mid false \mid c_I \mid f(t_1, \ldots, t_n) \mid \textbf{if } (t) \; t_1 \textbf{ else } t_2$$

where

- $c_I \in \mathbb{Z}$ denotes integer constant
- f denotes either application of a user-defined function or one of the primitive operators

Program representation as a mathematical structure

$p_{fact} = (e, fact(2))$
where environment e is defined by:

$$
\begin{array}{rlr}
e(fact) = (& n, & (parameters) \\
& Int, & (their\ types) \\
& \textbf{if } (n \leq 1)\ 1 \textbf{ else } n * fact(n-1), & (body) \\
& Int & (result\ type) \\
&) &
\end{array}
$$

Operational semantics of Amyli: **if** expression

Given a program with environment e, we specify the result of executing the program as an inductively defined binary (infix) relation "$\leadsto$" on expressions.

If the top-level expression becomes a constant after some number of steps of $\leadsto$, we have computed the result: $t \overset{*}{\leadsto} c$

Rules for **if**:

$$\frac{b \leadsto b'}{(\textbf{if}\ (b)\ t_1\ \textbf{else}\ t_2) \leadsto (\textbf{if}\ (b')\ t_1\ \textbf{else}\ t_2)}$$

$$\frac{}{(\textbf{if}\ (\mathit{true})\ t_1\ \textbf{else}\ t_2) \leadsto t_1}$$

$$\frac{}{(\textbf{if}\ (\mathit{false})\ t_1\ \textbf{else}\ t_2) \leadsto t_2}$$

b, b', t_1, t_2 range over expressions

Operational semantics of Amyli: primitives

Logical operators:

$$\frac{b_1 \rightsquigarrow b_1'}{(b_1 \ \&\& \ b_2) \rightsquigarrow (b_1' \ \&\& \ b_2)}$$

$$\frac{}{(\mathit{true} \ \&\& \ b_2) \rightsquigarrow b_2}$$

$$\frac{}{(\mathit{false} \ \&\& \ b_2) \rightsquigarrow \mathit{false}}$$

Arithmetic:

$$\frac{k_1 \rightsquigarrow k_1'}{(k_1 + k_2) \rightsquigarrow (k_1' + k_2)}$$

$$\frac{k_2 \rightsquigarrow k_2'}{(c + k_2) \rightsquigarrow (c + k_2')} \quad c \in \mathbb{Z}$$

$$\frac{}{(c_1 + c_2) \rightsquigarrow c} \quad c_1, c_2, c \in \mathbb{Z}, \ c = c_1 + c_2$$

Operational semantics: user function f

If $c_1, \ldots, c_{i-1}$ are constants, then (as expected in call-by-value)

$$\frac{t_i \rightsquigarrow t_i'}{f(c_1, \ldots, c_{i-1}, t_i, \ldots) \rightsquigarrow f(c_1, \ldots, c_{i-1}, t_i', \ldots)}$$

Let the environment e define f by $e(f) = ((x_1, \ldots, x_n), \bar{\tau}, t_f, \tau_0)$

- $(x_1, \ldots, x_n)$ is the list of formal parameters of f
- t_f is the body of the function f

Then we have a rule

$$\frac{}{f(c_1, \ldots, c_n) \rightsquigarrow t_f[x_1 := c_1, \ldots, x_n := c_n]}$$

In general, if t is term, then $t[x_1 := t_1, \ldots, x_n := t_n]$ denotes result of substituting (replacing) in t each variable x_i by term t_i.

Execution of factorial example program

$p_{fact} = (e, fact(2))$
where $e(fact) = (n, Int, \textbf{ if } (n \leq 1)\ 1 \textbf{ else } n * fact(n-1), Int)$

$$fact(2) \rightsquigarrow$$

Execution of factorial example program

$p_{fact} = (e, fact(2))$
where $e(fact) = (n, Int,\ \textbf{if}\ (n \le 1)\ 1\ \textbf{else}\ n * fact(n-1), Int)$

$$fact(2) \rightsquigarrow$$
$$\textbf{if}\ (2 \le 1)\ 1\ \textbf{else}\ 2 * fact(2-1) \rightsquigarrow$$

Execution of factorial example program

$p_{fact} = (e, fact(2))$
where $e(fact) = (n, Int, \textbf{ if } (n \leq 1)\ 1 \textbf{ else } n * fact(n-1), Int)$

$$\begin{aligned}
&fact(2) \rightsquigarrow \\
&\textbf{if } (2 \leq 1)\ 1 \textbf{ else } 2 * fact(2-1) \rightsquigarrow \\
&\textbf{if } (\mathit{false})\ 1 \textbf{ else } 2 * fact(2-1) \rightsquigarrow
\end{aligned}$$

Execution of factorial example program

$p_{fact} = (e, fact(2))$
where $e(fact) = (n, Int,\ \textbf{if}\ (n \leq 1)\ 1\ \textbf{else}\ n * fact(n-1), Int)$

$$
\begin{aligned}
&fact(2) \rightsquigarrow \\
&\textbf{if}\ (2 \leq 1)\ 1\ \textbf{else}\ 2 * fact(2-1) \rightsquigarrow \\
&\textbf{if}\ (false)\ 1\ \textbf{else}\ 2 * fact(2-1) \rightsquigarrow \\
&2 * fact(2-1) \rightsquigarrow
\end{aligned}
$$

Execution of factorial example program

$p_{fact} = (e, fact(2))$
where $e(fact) = (n, Int,\ \textbf{if}\ (n \leq 1)\ 1\ \textbf{else}\ n * fact(n-1), Int)$

$$
\begin{aligned}
&fact(2) \rightsquigarrow \\
&\textbf{if}\ (2 \leq 1)\ 1\ \textbf{else}\ 2 * fact(2-1) \rightsquigarrow \\
&\textbf{if}\ (\mathit{false})\ 1\ \textbf{else}\ 2 * fact(2-1) \rightsquigarrow \\
&2 * fact(2-1) \rightsquigarrow \\
&2 * fact(1) \rightsquigarrow
\end{aligned}
$$

Execution of factorial example program

$$p_{fact} = (e, fact(2))$$
where $e(fact) = (n, Int,\ \textbf{if}\ (n \leq 1)\ 1\ \textbf{else}\ n * fact(n-1), Int)$

$$\begin{aligned}
& fact(2) \rightsquigarrow \\
& \textbf{if}\ (2 \leq 1)\ 1\ \textbf{else}\ 2 * fact(2-1) \rightsquigarrow \\
& \textbf{if}\ (false)\ 1\ \textbf{else}\ 2 * fact(2-1) \rightsquigarrow \\
& 2 * fact(2-1) \rightsquigarrow \\
& 2 * fact(1) \rightsquigarrow \\
& 2 * (\textbf{if}\ (1 \leq 1)\ 1\ \textbf{else}\ 1 * fact(1-1)) \rightsquigarrow
\end{aligned}$$

Execution of factorial example program

$$p_{fact} = (e, fact(2))$$
where $e(fact) = (n, Int,\ \textbf{if}\ (n \le 1)\ 1\ \textbf{else}\ n * fact(n-1), Int)$

$$\begin{aligned}
& fact(2) \rightsquigarrow \\
& \textbf{if}\ (2 \le 1)\ 1\ \textbf{else}\ 2 * fact(2-1) \rightsquigarrow \\
& \textbf{if}\ (false)\ 1\ \textbf{else}\ 2 * fact(2-1) \rightsquigarrow \\
& 2 * fact(2-1) \rightsquigarrow \\
& 2 * fact(1) \rightsquigarrow \\
& 2 * (\textbf{if}\ (1 \le 1)\ 1\ \textbf{else}\ 1 * fact(1-1)) \rightsquigarrow \\
& 2 * (\textbf{if}\ (true)\ 1\ \textbf{else}\ 1 * fact(1-1)) \rightsquigarrow
\end{aligned}$$

Execution of factorial example program

$p_{fact} = (e, fact(2))$
where $e(fact) = (n, Int,\ \textbf{if}\ (n \leq 1)\ 1\ \textbf{else}\ n * fact(n-1), Int)$

$$
\begin{aligned}
& fact(2) \rightsquigarrow \\
& \textbf{if}\ (2 \leq 1)\ 1\ \textbf{else}\ 2 * fact(2-1) \rightsquigarrow \\
& \textbf{if}\ (false)\ 1\ \textbf{else}\ 2 * fact(2-1) \rightsquigarrow \\
& 2 * fact(2-1) \rightsquigarrow \\
& 2 * fact(1) \rightsquigarrow \\
& 2 * (\textbf{if}\ (1 \leq 1)\ 1\ \textbf{else}\ 1 * fact(1-1)) \rightsquigarrow \\
& 2 * (\textbf{if}\ (true)\ 1\ \textbf{else}\ 1 * fact(1-1)) \rightsquigarrow \\
& 2 * 1 \rightsquigarrow
\end{aligned}
$$

Execution of factorial example program

$$p_{fact} = (e, fact(2))$$
where $e(fact) = (n, Int,\ \textbf{if}\ (n \leq 1)\ 1\ \textbf{else}\ n * fact(n-1), Int)$

$$\begin{aligned}
& fact(2) \rightsquigarrow \\
& \textbf{if}\ (2 \leq 1)\ 1\ \textbf{else}\ 2 * fact(2-1) \rightsquigarrow \\
& \textbf{if}\ (false)\ 1\ \textbf{else}\ 2 * fact(2-1) \rightsquigarrow \\
& 2 * fact(2-1) \rightsquigarrow \\
& 2 * fact(1) \rightsquigarrow \\
& 2 * (\textbf{if}\ (1 \leq 1)\ 1\ \textbf{else}\ 1 * fact(1-1)) \rightsquigarrow \\
& 2 * (\textbf{if}\ (true)\ 1\ \textbf{else}\ 1 * fact(1-1)) \rightsquigarrow \\
& 2 * 1 \rightsquigarrow \\
& 2
\end{aligned}$$