

# Value Conversion in IL1 after Lambda Hoisting

Lee Gao (lg342)

April 15, 2013

Given the closure-conversion and then hoisted restricted language IL1 detailed below

$$\begin{aligned} v &::= n \mid x \mid \text{halt} \\ e &::= v \mid v_0 + v_1 \mid (v_0, \dots, v_n) \mid \pi_n v \\ c &::= \text{let } x = e \text{ in } c \mid v_0 \ v_1 \ v_2 \mid v_0 \ v_1 \end{aligned}$$

we want to “lift” numbers and halt into expressions (as Val used in bindings only) and leave only variables as values, so in effect our language will now look like

$$\begin{aligned} v &::= x \\ e &::= v \mid \text{val}(n) \mid \text{val}(\text{halt})v_0 + v_1 \mid (v_0, \dots, v_n) \mid \pi_n v \\ c &::= \text{let } x = e \text{ in } c \mid v_0 \ v_1 \ v_2 \mid v_0 \ v_1 \end{aligned}$$

We want to define a set of “lowering” translation  $\mathcal{LV}[\![v]\!]$ ,  $\mathcal{LE}[\![e]\!]$ ,  $\mathcal{LC}[\![c]\!]$  that binds all non-variable values (integers and halts) into their own variables. Therefore, we need to have both the value and the expressions translation be able to be abstracted as bindings.

$$\begin{aligned} \mathcal{LV}[\![v]\!] &: (\text{var} \times e)\text{list} \times \text{var} \\ \mathcal{LE}[\![e]\!] &: (\text{var} \times e)\text{list} \times e \\ \mathcal{LC}[\![c]\!] &: c \end{aligned}$$

We will use the notation

$$\text{let } x_i = e_i \text{ in } x$$

to mean

$$([(x_i, e_i); \dots (x_n, e_n)], x)$$

and respective syntax sugary for expressions.

## 1 Values

$$\begin{aligned} \mathcal{LV}[\![n]\!] &= \text{let } x = n \text{ in } x \\ \mathcal{LV}[\![x]\!] &= ([], x) \\ \mathcal{LV}[\![\text{halt}]\!] &= \text{let } x = \text{halt} \text{ in } x \end{aligned}$$

## 2 Expressions

Notice immediately that we only ever translate  $e$  in the context of let expressions, so we don't have to translate the case of just a value.

$$\begin{aligned}\mathcal{LE}\llbracket v_0 + v_1 \rrbracket &= \text{let}(l_0, x_0) = \mathcal{LV}\llbracket v_0 \rrbracket \text{ in let}(l_1, x_1) = \mathcal{LV}\llbracket v_1 \rrbracket \text{ in } (l_0 \cup l_1, x_0 + x_1) \\ \mathcal{LE}\llbracket (v_0, \dots, v_n) \rrbracket &= \text{let}(l_0, x_0) = \mathcal{LV}\llbracket v_0 \rrbracket \text{ in } \dots \text{let}(l_n, x_n) = \mathcal{LV}\llbracket v_n \rrbracket \text{ in } (\bigcup l_k, (x_1, \dots, x_n)) \\ \mathcal{LE}\llbracket \pi_n v \rrbracket &= \text{let}(l, x) = \mathcal{LV}\llbracket v \rrbracket \text{ in } (l, \pi_n x)\end{aligned}$$