1 Labelled Transition System for Complete Head Linear Reduction

1.1 Notes

State is a tuple $\langle$ λ-term with underlined node; context; list of pairs (argument, context) $\rangle$, where

- λ-term (a tree; by considering λ-term as a tree it becomes possible to cross arguments out of tree with a parent application node (denotes as $A_{\cancel{@}}$)) is a usual lambda term with one underlined node;
- Context Γ is an ordered (first-in) set of elements of the following form $(variable \mapsto (term,\ Context))$;
- Δ is an ordered list of pairs (λ-term, Context) (one can also think about Δ as an ordered list of pointers to the corresponding subtree stored together with a corresponding context);
- Initial configuration is $\langle \lambda - term\ with\ underlined\ root,\ \emptyset,\ \$\rangle$;
- Final configuration is $\langle M\,[\underline{x}]$ (term in normal form), $\Gamma,\ \$\rangle$;
- Note, that Complete Head Linear Reduction (CHLR) can be seen as usual Head Linear Reduction (HLR) that is repeatedly applied to the all arguments;
- From the below LTS for CHLR.

1.2 Labelled transition system for CHLR

$$\langle A\,[e_1 \underline{@} e_2]\,;\ \Gamma;\ \Delta\rangle \longrightarrow \left\langle A\,\left[\underline{e_1}@e_2\right]\,;\ \Gamma;\ (e_2,\Gamma)\bullet\Delta\right\rangle \qquad \text{[App]}$$

$$\langle A\,[\underline{\lambda x}.e]\,;\ \Gamma;\ \$\bullet\Delta\rangle \longrightarrow \langle A\,[\lambda x.\underline{e}]\,;\ \Gamma;\ \$\bullet\Delta\rangle \qquad \text{[Lam-Non-Elim]}$$

$$\langle A\,[\underline{\lambda x}.e]\,;\ \Gamma;\ (B,\Gamma')\bullet\Delta\rangle \longrightarrow \langle A_{\cancel{@}}\,[\cancel{\lambda x}.\underline{e}]\,;\ x\mapsto(B,\Gamma'),\ \Gamma;\ \Delta\rangle \qquad \text{[Lam-Elim]}$$

$$\langle A\,[\underline{x}]\,;\ x\mapsto(B,\Gamma'),\ \Gamma;\ \Delta\rangle \longrightarrow \langle A\,[\underline{B}]\,;\ \Gamma';\ \Delta\rangle \qquad \text{[BVar]}$$

$$\langle A\,[M\,[\underline{x}]\,@B]\,;\ \Gamma;\ (B,\Gamma')\bullet\$\bullet\Delta\rangle \longrightarrow \langle A\,[M[x]@\underline{B}]\,;\ \Gamma';\ \$\bullet\Delta\rangle,\ x\notin dom(\Gamma) \qquad \text{[FVar-0]}$$

$$\langle A\,[M\,[\underline{x}]\,@B]\,;\ \Gamma;\ (B,\Gamma')\bullet C\bullet\Delta\rangle \longrightarrow \langle A\,[M[x]@\underline{B}]\,;\ \Gamma';\ \$\bullet C\bullet\Delta\rangle,\ C\neq\$,\ x\notin dom(\Gamma) \qquad \text{[FVar-1]}$$

$$\langle A\,[M\,[\underline{x}]\,@B]\,;\ \Gamma;\ \$\bullet(B,\Gamma')\bullet\Delta\rangle \longrightarrow \langle A\,[M\,[x]\,@\underline{B}]\,;\ \Gamma';\ \$\bullet\Delta\rangle,\ \ x\notin dom(\Gamma) \qquad \text{[FVar-2]}$$

1.3 Correctness proof

Note that that labelled transition system for HLR can be obtained by removing [FVAR-*] rules from the LTS for CHLR. In other words, first four rules describe HLR while [FVAR-*] rules describes a repeated application of HLR to the arguments.

LTS for HLR .
Remark: it is obvious that in [Lam-Non-Elim] rule for HLR we can safely replace $\$\bullet\Delta$ by empty list since there is no rule that pop $ sign from Δ.

$$\langle A\,[e_1 \underline{@} e_2]\,;\ \Gamma;\ \Delta\rangle \longrightarrow \left\langle A\,\left[\underline{e_1}@e_2\right]\,;\ \Gamma;\ (e_2,\Gamma)\bullet\Delta\right\rangle \qquad \text{[App]}$$

$$\langle A\,[\underline{\lambda x}.e]\,;\ \Gamma;\ []\rangle \longrightarrow \langle A\,[\lambda x.\underline{e}]\,;\ \Gamma;\ []\rangle \qquad \text{[Lam-Non-Elim]}$$

$$\langle A\,[\underline{\lambda x}.e]\,;\ \Gamma;\ (B,\Gamma')\bullet\Delta\rangle \longrightarrow \left\langle A_{\cancel{@}}\,[\cancel{\lambda x}.\underline{e}]\,;\ x\mapsto(B,\Gamma'),\ \Gamma;\ \Delta\right\rangle \qquad \text{[Lam-Elim]}$$

$$\langle A\,[\underline{x}]\,;\ x\mapsto(B,\Gamma'),\ \Gamma;\ \Delta\rangle \longrightarrow \langle A\,[\underline{B}]\,;\ \Gamma';\ \Delta\rangle \qquad \text{[BVar]}$$

Expansion function. To prove correctness of HLR we introduce an axiliary expansion function exp. This function being applied to some state of LTS for HLR results in some λ-term which we will call an expansion.

$$exp\ \langle M[\underline{A}];\ \Gamma,\ x \mapsto (B, \Gamma');\ \Delta\rangle = exp\ \langle M[\underline{A}[x/B[\Gamma']]];\ \Gamma;\ \Delta\rangle \tag{1}$$

$$exp\ \langle M_B[\underline{A}];\ \emptyset;\ (B, \Gamma') \bullet \Delta\rangle = exp\ \langle M_{B[\Gamma']}[\underline{A}];\ \emptyset;\ \Delta\rangle \tag{2}$$

$$exp\ \langle M;\ \emptyset;\ []\rangle = M \tag{3}$$

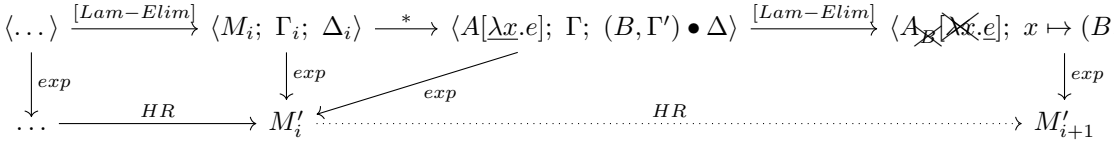
$$\text{where } B[\Gamma'] = exp\ \langle \underline{B};\ \Gamma';\ []\rangle \tag{4}$$

Note that (1) substitudes term $B[\Gamma']$ instead of all occurence of variable x in the subtree of term M that has an underlined node as a root (i.e. subtree A). Each recurcive call of (2) substitudes all variables with respect to the corresponding context exactly in one argument term.

Now consider LTS for HLR. Note that:

- Only $[Lam - Elim]$ rule can change an expansion;
- The number of transitions without applying $[Lam - Elim]$ rule is limited since the definition of context, input term has a final size, and only $[Lam - Elim]$ rule expands the context; Hereafter we will denote a sequence of transitions without applying $[Lam - Elim]$ rule by $\xrightarrow{*}$;
- The expansion function exp cannot change a path from the root of term to the underlined node since:
  - recurcive call (1) can only change the subtree of the current term with underlined node as a root; (I)
  - recurcive call (2) can only change arguments of the current term that are above the underlined node; Moreover, each application of (2) change exactly one argument (that corresponds to the first element of the pair in Δ) leaving all other arguments and the subtree of the current term with underlined node as a root intact. (II)

Theorem . HLR and HR coincide as follows:



Proof by induction on number of $[Lam - Elim]$ steps.

Base: trivial since first element being added to Γ is a head redex by definition.

Induction step. We know that after i^{th} application of rule $[Lam - Elim]$ we got some LTS state $\langle M_i;\ \Gamma_i;\ \Delta_i\rangle$ which expansion is some term M'_i. As was mentioned above $\xrightarrow{*}$ has not change the expansion and the number of steps in $\xrightarrow{*}$ is finite. Thus, we have two possible cases:

first, LTS got stuck and nothing to prove,

and second, we can apply the $[Lam - Elim]$ rule. In the second case we know the form of the state (following the $[Lam - Elim]$ definition), i.e. $\langle A[\underline{\lambda x}.e];\ \Gamma;\ (B, \Gamma') \bullet \Delta\rangle$ (let us denote it (i)), and by assumption the result of applying $[Lam - Elim]$ rule is a state $\langle A_{\not B[\not\lambda x.\underline{e}]};\ x \mapsto (B, \Gamma'), \Gamma;\ \Delta\rangle$ (let us denote it (ii)). Now we have to show that applying function exp to that state results in the same term as a step of head reduction from term M'_i. (i.e. in the term M'_{i+1})

Let see what will happend if we apply exp function to state (i):

$$exp\ \langle A[\underline{\lambda x}.\ e];\ \Gamma;\ (B, \Gamma') \bullet \Delta\rangle \tag{5}$$

$$\xrightarrow{exp}{}^{*}\ exp\ \langle A'[\underline{\lambda x}.\ e[\Gamma]];\ \emptyset;\ (B, \Gamma') \bullet \Delta\rangle \qquad \text{by fully applying (1)} \tag{6}$$

$$\xrightarrow{exp}\ exp\ \langle A'_{B[\Gamma']}[\underline{\lambda x}.\ e[\Gamma]];\ \emptyset;\ \Delta\rangle \qquad \text{by one step of (2)} \tag{7}$$

$$\xrightarrow{exp}\ \ldots \tag{8}$$

to state (ii):

$$exp\langle A_{\not B[\not\lambda x.\ \underline{e}]};\ x \mapsto (B, \Gamma'), \Gamma;\ \Delta\rangle \tag{9}$$

$$\xrightarrow{exp}{}^{*}\ exp\ \langle A_{\not B[\not\lambda x.\ \underline{e}[\Gamma]]};\ x \mapsto (B, \Gamma');\ \Delta\rangle \qquad \text{by elliminating } \Gamma \text{ following (1)} \tag{10}$$

$$\xrightarrow{exp}\ exp\ \langle A_{\not B[\not\lambda x.\ \underline{e}[\Gamma][x/B[\Gamma']]]};\ \emptyset;\ \Delta\rangle \qquad \text{by one step of (2)} \tag{11}$$

$$\xrightarrow{exp}\ \ldots \tag{12}$$

It is easy to see that term in state in (7), $A'_{B[\Gamma']}[\underline{\lambda x}.\ e[\Gamma]]$, has the head redex $(\lambda x,\ B)$ (since context is already empty). Thus, if we apply head reduction to this term, it will results exactly in term $A_{\not B[\not\lambda x.\ \underline{e}[\Gamma][x/B[\Gamma']]]}$ (by head reduction definition) that is equal to the first component of state (11). Following (I) and (II), if we will continue to apply exp function then elliminating the same Δ in both cases will lead no effect on argument B and underlined subtrees. Thus,

$(\lambda x,\ B)$ is a head redex in M'_i and the expansion of state (ii) is M'_{i+1}. Qed

Consequences:

- if head linear reduction terminates then head reduction terminates;
- if head reduction terminates then head linear reduction terminates;
- Since expansion is not able to change a path from the root to the underlined node, the first component of the final state of LTS for head linear reduction contains a term which leftmost path is a part of the head normal form and the expansion of the final state is a node of the Boehm-tree of the input term;
  Thus, a repeated application of head linear reduction to all arguments (complete head linear reduction) leeds to the normal form.

2 Labelled Transition System for UNP

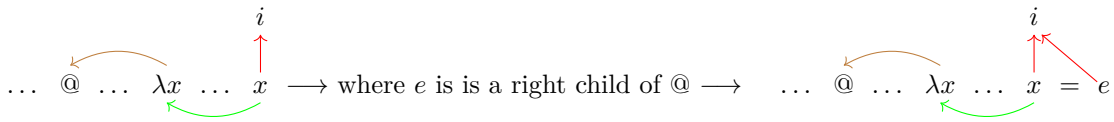
This section contains LTS for UNP. This LTS has the same number of rules that the LTS for CHLR. Moreover, it is easy to see that rules with the same name corresponds to each other.

There are two pointer types:

- Binder pointer (denoted as →); from the bound variables binder pointer points to the dynamic binder in the history and for all other nodes it points points to the parent; (these kind of pointer is omitted in these notes since they can be easly defined by recurcive procedure)
- Pointer to the last incomplete application; there are two subtypes of this pointer:
  - Brown pointer (→) can binds either lambda node and application (in this case they arrise as spine redexes) or any other node and some application;
  - Vioilet pointer (→) has the same meaning (pointer to the last incamplete application) together with $ in LTS for CHLR;
  - Red pointer (→) is used when the real colour (brown or violet) is not the metter.

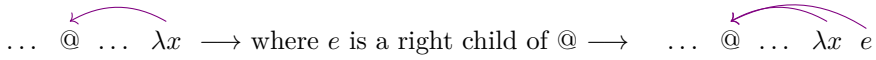
Rules to construct a traversal:

- $(BVar)$

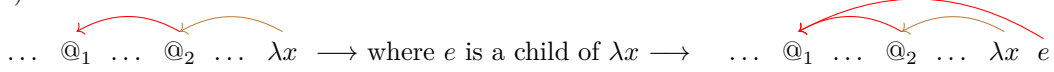


NB: the colour of the added (red) pointer has to be the same as the colour of the first "red"pointer

- $(Lam - Non - Elim)$

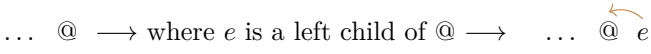


- $(Lam - Elim)$

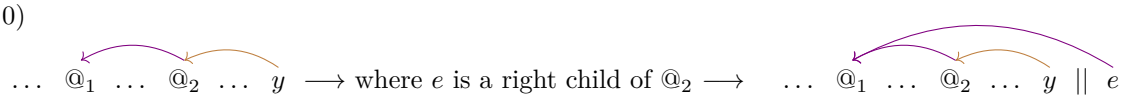


NB: the colour of the added (red) pointer has to be the same as the colour of the first "red"pointer

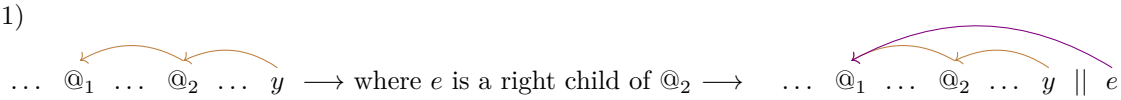
- (App)



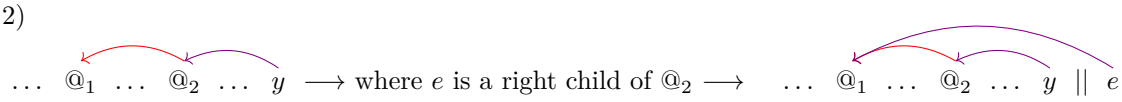
- $(FVar - 0)$



- $(FVar - 1)$



- $(FVar - 2)$



NB: the colour of the "red"pointer is not the metter in this case

Note that Δ and Γ can be easly defined from UNP LTS state as follows:

- Δ can be construct by following incomplete application pointer and storing right childs of all reachable applications in reverse order with corresponding contexts (see next item how to define context from traversal);
- Γ can be construct by following binder pointers, and when faces with a lambda–tokens λx which has a brown pointer to some application @, store $x \mapsto (e, \Gamma')$ where e is a right child of apllication @ and Γ' is a context (which is defined recursively from token @).