Languages details - draft

July 17, 2012

1 Labeled language

TODO: short intro + some facts important from implementation perspective (one context etc) + maybe some motivation behind choosing one formalism over others; may be split across subsections

1.1 Statics

1.1.1 Language

This is a description of actual language, as presented in [1].

$$\mathrm{hyp_L}\frac{w \in \Omega}{\Omega; (x : A@w) \cup \Gamma \vdash \mathrm{hyp_L} x : A@w}$$

$$\mathrm{lam_L}\frac{\Omega; (x : A@w) \cup \Gamma \vdash M : A'@w}{\Omega; \Gamma \vdash \lambda_L x, M : A \to A'@w}$$

$$\mathrm{appl_L}\frac{\Omega; \Gamma \vdash M : A \to A'@w \quad \Omega; \Gamma \vdash N : A@w}{\Omega; \Gamma \vdash \mathrm{appl_L} M N : A'@w}$$

$$\mathrm{box_L}\frac{w \in \Omega \quad w' \mathrm{fresh} \quad w' \cup \Omega; \Gamma \vdash M : A@w'}{\Omega; \Gamma \vdash \mathrm{box_L} w'.M : \Box A@w}$$

$$\mathrm{unbox_L}\frac{\Omega; \Gamma \vdash M : \Box A@w}{\Omega; \Gamma \vdash \mathrm{unbox_L} M : A@w}$$

$$\mathrm{here_L}\frac{\Omega; \Gamma \vdash M : A@w}{\Omega; \Gamma \vdash \mathrm{here_L} M : \Diamond A@w}$$

$$\mathrm{letd_L}\frac{\Omega;\Gamma \vdash M : \Diamond A@w \quad w'\mathrm{fresh} \quad \cup\,\Omega;(A,w')\cup\Gamma \vdash N : B@w}{\Omega;\Gamma \vdash \mathrm{letdia_L}x := M\mathrm{in}N : B@w}$$

$$\mathrm{fetch_L}\frac{w \in \Omega \quad \Omega;\Gamma \vdash M : \Box A@w'}{\Omega;\Gamma \vdash \mathrm{fetch_L}w'M : \Box A@w'}$$

$$\mathrm{get_L}\frac{w \in \Omega \quad \Omega;\Gamma \vdash M : \Diamond A@w'}{\Omega;\Gamma \vdash \mathrm{get_L}w'.M : \Diamond A@w'}$$

1.1.2 Implementation

Note 1 *This is just current, not final implementation of labeled language. Planned changes include using locally nameless instead of de Bruijn indices for variables and replacing current implementation of Ω (using fsets from Metatheory library) with one using lists and their permutations.*
Both changes are due to compatibility in translation with label-free language.

$$\mathrm{hyp_L}\frac{w \in \Omega \quad \Gamma[n] = (A,w)}{\Omega;\Gamma \vdash \mathrm{hyp_L}n : A@w}$$

$$\mathrm{lam_L}\frac{\Omega;(A,w)::\Gamma \vdash M : A'@w}{\Omega;\Gamma \vdash \lambda_L A,M : A \to A'@w}$$

$$\mathrm{appl_L}\frac{\Omega;\Gamma \vdash M : A \to A'@w \quad \Omega;\Gamma \vdash N : A@w}{\Omega;\Gamma \vdash \mathrm{appl_L}MN : A'@w}$$

$$\mathrm{box_L}\frac{w \in \Omega \quad \forall_w w \notin \mathrm{L} \to w' \cup \Omega;\Gamma \vdash M^{w'} : A@w'}{\Omega;\Gamma \vdash \mathrm{box_L}M : \Box A@w}$$

$$\mathrm{unbox_L}\frac{\Omega;\Gamma \vdash M : \Box A@w}{\Omega;\Gamma \vdash \mathrm{unbox_L}M : A@w}$$

$$\mathrm{here_L}\frac{\Omega;\Gamma \vdash M : A@w}{\Omega;\Gamma \vdash \mathrm{here_L}M : \Diamond A@w}$$

$$\mathrm{letd_L}\frac{w \in \Omega \quad \Omega;\Gamma \vdash M : \Diamond A@w \quad \forall_{w'} w' \notin \mathrm{L} \to w' \cup \Omega;(x : A,w')::\Gamma \vdash N^{w'} : B@w}{\Omega;\Gamma \vdash \mathrm{letdia_L}M\mathrm{in}N : B@w}$$

$$\mathrm{fetch_L}\frac{w \in \Omega \quad \Omega;\Gamma \vdash M : \Box A@w'}{\Omega;\Gamma \vdash \mathrm{fetch_L}w'M : \Box A@w'}$$

$$\mathrm{get_L}\frac{w \in \Omega \quad \Omega; \Gamma \vdash M : \Diamond A@w'}{\Omega; \Gamma \vdash \mathrm{get_L} w' M : \Diamond A@w'}$$

1.2 Dynamics

1.2.1 Values

$$\mathrm{value_L}(\lambda_L A, M)$$

$$\mathrm{value_L}(\mathrm{box_L} M)$$

$$\mathrm{value_L}(M) \to \mathrm{value_L}(\mathrm{here_L} M)$$

1.2.2 Reductions

For reduction steps, we silently use assumption that all terms are locally closed with respect to worlds. This requirement may later be moved to static semantics, as is is widely used both in definitions and lemmas.
We do not list trivial inductive reduction steps here.

$$\mathrm{appl_L}(\lambda_L A, M) N, w \mapsto [N/0]M, w$$

$$\mathrm{unbox_L}(\mathrm{box_L} M), w \mapsto M^w, w$$

$$\mathrm{letd_L}(\mathrm{here_L} M) \mathrm{in} N, w \mapsto [M/0]N^w, w$$

$$\mathrm{value_L} M \to \mathrm{fetch_L} w M, w' \mapsto \{w'/w\}M, w'$$

$$\mathrm{value_L} M \to \mathrm{get_L} w (\mathrm{here_L} M), w' \mapsto \mathrm{here_L}\{w'/w\}M, w'$$

1.3 Substitution

Since the labeled language has only one context stack, both definition and actual implementation of substitution functions is streight forward. For the sake of completeness, we will provide the definitions.

1.3.1 "world" substitution

We use helper function shift, which does the following:

- On free world, it is an identity function
- On bound world, it increases the binding number by 1

$$\{w_1/w_2\}(\mathrm{hyp_L}n) \Rightarrow \mathrm{hyp_L}n$$
$$\{w_1/w_2\}(\lambda_L A, M) \Rightarrow \lambda_L A, \{w_1/w_2\}M$$
$$\{w_1/w_2\}(\mathrm{appl_L}MN) \Rightarrow \mathrm{appl_L}\{w_1/w_2\}M\{w_1/w_2\}N$$
$$\{w_1/w_2\}(\mathrm{box_L}M) \Rightarrow \mathrm{box_L}\{\mathrm{shift}w_1/\mathrm{shift}w_2'\}M$$
$$\{w_1/w_2\}(\mathrm{unbox_L}M) \Rightarrow \mathrm{unbox_L}\{w_1/w_2\}M$$
$$\{w_1/w_2\}(\mathrm{here_L}n) \Rightarrow \mathrm{here_L}\{w_1/w_2\}M$$
$$\{w_1/w_2\}(\mathrm{letd_L}MN) \Rightarrow \mathrm{letd_L}\{w_1/w_2\}M\{\mathrm{shift}w_1/\mathrm{shift}w_2\}M$$
$$\{w_1/w_2\}(\mathrm{fetch_L}w_2M) \Rightarrow \mathrm{fetch_L}w_1\{w_1/w_2\}M$$
$$\{w_1/w_2\}(\mathrm{fetch_L}wM) \Rightarrow \mathrm{fetch_L}w\{w_1/w_2\}M$$
$$\{w_1/w_2\}(\mathrm{get_L}w_2M) \Rightarrow \mathrm{get_L}w_1\{w_1/w_2\}M$$
$$\{w_1/w_2\}(\mathrm{get_L}wM) \Rightarrow \mathrm{get_L}w\{w_1/w_2\}M$$

1.3.2 "term" substitution

$$[M/n]\mathrm{hyp_L}n \Rightarrow M$$
$$[M/n]\mathrm{hyp_L}m \Rightarrow \mathrm{hyp_L}m$$
$$[M/n]\lambda_L A, N \Rightarrow \lambda_L A, [M/\mathrm{S}n]N$$
$$[M/n]\mathrm{appl_L}N_1N_2 \Rightarrow \mathrm{appl_L}[M/n]N_1[M/n]N_2$$
$$[M/n]\mathrm{box_L}N \Rightarrow \mathrm{box_L}[M/n]N$$
$$[M/n]\mathrm{unbox_L}N \Rightarrow \mathrm{unbox_L}[M/n]N$$
$$[M/n]\mathrm{here_L}N \Rightarrow \mathrm{here_L}[M/n]N$$
$$[M/n]\mathrm{letd_L}N_1\mathrm{in}N_2 \Rightarrow \mathrm{letd_L}[M/n]N_1\mathrm{in}[M/\mathrm{S}n]N_2$$
$$[M/n]\mathrm{fetch_L}N \Rightarrow \mathrm{fetch_L}[M/n]N$$
$$[M/n]\mathrm{get_L}N \Rightarrow \mathrm{get_L}[M/n]N$$

2 Label-free language

The original description of logic behind this language is in [2] and is, indeed, label-free. The paper does not provide any terms to go with the logic, so we had to add them on out own. We will use the example of $\Box A$ type.
$\Box_I$ rule is very streightforward

$$\text{box}_{\text{LF}}\frac{\Gamma; G \vdash \emptyset \vdash M : A}{G \vdash \Gamma \vdash \text{box}_{\text{LF}} M : \Box A}$$

First, we wanted $\Box_E$ rules to be the following:

$$\text{unbox}_{\text{LF}}\frac{G \vdash \Gamma \vdash M : \Box A}{G \vdash \Gamma \vdash \text{unbox}_{\text{LF}} M : A}$$

$$\text{unbox\_fetch}_{\text{LF}}\frac{\Gamma; G \vdash \Gamma' \vdash M : \Box A}{\Gamma'; G \vdash \Gamma \vdash \text{unbox\_fetch}_{\text{LF}} M : A}$$

So without explicitly stating in the term, what is the context used for fetch operation. But this causes problems when we start to think of how to define reduction for $\Box$ type. We want to have this proof

$$\frac{\dfrac{\overset{\mathcal{D}}{G; \Gamma; \Gamma' \vdash \emptyset \vdash M : A}}{G; \Gamma \vdash \Gamma' \vdash \text{box} M : \Box A}}{G; \Gamma' \vdash \Gamma \vdash \text{unbox\_fetch}(\text{box} M) : A}$$

reduced to

$$\overset{\mathcal{D}}{G; \Gamma' \vdash \Gamma \vdash M : A}$$

with some modifications on $\mathcal{D}$ and M - namely, e.g. whenever we see unbox$_{\text{f}}$etch within M, we have to figure out whether we are exchanging the current context for Γ - if so, we should use unbox instead.

Of course this approach would require looking at entire judgements, not just terms, in order to make the change - since we do not know, what context will be switched by unbox$_{\text{f}}$etch.

One solution would be to try and unify unbox_fetch with unbox terms by stating:

$$\text{unbox}_{\text{LF}}\frac{G \vdash \Gamma \vdash M : \Box A}{G \vdash \Gamma \vdash \text{unbox\_fetchLF} M : A}$$

but this would be incorrect, since then we add disambiguity to the typing system.

Let us then try to add context to the term when we are switching it. The rule for fetch would change to the following:

$$\text{unbox\_fetch}_{\text{LF}}\frac{\Gamma; G \vdash \Gamma' \vdash M : \Box A}{\Gamma'; G \vdash \Gamma \vdash \text{unbox\_fetch}_{\text{LF}}\Gamma' M : A}$$

This allows us to easily make modifications on M, but brings our attention to a different problem. What if two identical contexts Γ exist in the system, one being the 'close' context and the other the 'background' one, $G; \Gamma \vdash \Gamma \vdash M : A$?
Then we can no longer be sure that our modified M is correctly transformed, since we cannot distinguish between two contexts, one being in the background and the other in the front.

It seems that the only solution is to avoid such situation - ensure that contexts are easily distinguishable. One obvious way is to state that each variable name occurs only once in all the contexts. It solves almost all of the problems - except for the case of empty contexts. To distinguish between those, we do in fact need to explicitly name each context with a unique name.

In other applications we may consider undistinguishable empty contexts, but since we do intend to create a language equivalent to labeled IS5, here we have to tell them apart.

2.1 Statics

2.1.1 Language

$$\text{hyp}_{\text{LF}}\frac{}{G \vdash w, (x : A) \cup \Gamma \vdash \text{hyp}_{\text{LF}} x : A}$$

$$\text{lam}_{\text{LF}}\frac{G \vdash w, (x : A) \cup \Gamma \vdash M : A'}{G \vdash w, \Gamma \vdash \lambda_{LF} x, M : A \to A'}$$

$$\text{appl}_{\text{LF}}\frac{G \vdash w, \Gamma \vdash M : A \to A' \quad G \vdash w, \Gamma \vdash N : A}{G \vdash w, \Gamma \vdash \text{appl}_{\text{LF}} M N : A'}$$

$$\text{box}_{\text{LF}}\frac{w' \text{fresh} \quad (w, \Gamma) \cup G \vdash w', \emptyset \vdash M : A}{G \vdash w, \Gamma \vdash \text{box}_{\text{LF}} M : \Box A}$$

$$\text{unbox}_{\text{LF}}\frac{G \vdash w, \Gamma \vdash M : \Box A}{G \vdash w, \Gamma \vdash \text{unbox}_{\text{LF}} M : A}$$

$$\text{unbox\_fetch}_{\text{LF}}\frac{(w,\Gamma)\cup G\vdash w',\Gamma'\vdash M:\Box A}{(w',\Gamma')\cup G\vdash w,\Gamma\vdash \text{unbox\_fetch}_{\text{LF}}w'M:A}$$

$$\text{here}_{\text{LF}}\frac{G\vdash w,\Gamma\vdash M:A}{G\vdash w,\Gamma\vdash \text{here}_{\text{LF}}M:\Diamond A}$$

$$\text{get\_here}_{\text{LF}}\frac{(w,\Gamma)\cup G\vdash w',\Gamma'\vdash M:\Diamond A}{(w',\Gamma')\cup G\vdash w,\Gamma\vdash \text{get\_here}_{\text{LF}}w'.M:\Diamond A}$$

$$\text{letdia}_{\text{LF}}\frac{G\vdash w,\Gamma\vdash M:\Diamond A\quad w'\text{fresh}\quad (x:A,w')\cup G\vdash w,\Gamma\vdash N:B}{G\vdash w,\Gamma\vdash \text{letdia}_{\text{LF}}x:=M\text{in}N:B}$$

$$\text{letdia\_get}_{\text{LF}}\frac{(w,\Gamma)\cup G\vdash w',\Gamma'\vdash M:\Diamond A\quad w'\text{fresh}\quad (x:A,w')\cup G\vdash w,\Gamma\vdash N:B}{(w',\Gamma')\cup G\vdash w,\Gamma\vdash \text{letdia\_get}_{\text{LF}}x:=M\text{in}N:B}$$

2.2 Implementation

*implementation details to come as they will be finalized in the development

2.3 Dynamics

* with trees :)

2.4 Substitution

3 Translation

4 Normalization properties

References

[1] Tom Murphy VII, Karl Crary, Robert Harper, Frank Pfenning A Symetric Modal Lambda calculus for Distributed Computing

[2] Didier Galmiche, Yakoub Salhi Label-free Proof Systems for Intuitionistic Modal Logic IS5