Dynamic Semantics in Direct Style

Scott Martin

April 29, 2010

1 Call-by-value $\lambda\mu$-calculus

The call-by-value $\lambda\mu$-calculus ($\lambda\mu_V$, de Groote, 2008) is an extension of typed λ-calculus, adapted from the $\lambda\mu$-calculus of Parigot (1992). $\lambda\mu_V$ differs from Parigot's calculus in that:

- The notion of a class of terms called **values** is introduced (see Definition 2). These terms are treated specially with respect to certain reduction rules.
- A "symmetric" reduction rule (μ_V) is added that is similar to rule R_4 of Parigot's (2000) symmetric $\lambda\mu$-calculus. A reduction rule for μ-variables (Rename) is also added.
- A new reset operator is introduced, with typing and reduction rules that govern its use.

Like the $\lambda\mu$-calculus of Parigot (1992), $\lambda\mu_V$ allows a term to be "named" and used (via μ-application and μ-abstraction) so that its name is preserved under reduction. We mildly extend de Groote's $\lambda\mu_V$ by adding product for both terms and types.

Definition 1 (Terms)**.** Terms are generated from two disjoint sets of variables, λ-variables (represented by x) and μ-variables (represented by a), according to the following rules:

$$M ::= c \mid x \mid \lambda_x M \mid (M\ M) \mid \langle M, M\rangle \mid \mu_a M \mid (a\ M) \mid \lceil M \rfloor$$

where c is a constant. We adopt the convention that applications and pairings associate to the left and abstractions associate to the right. Parentheses are sometimes abbreviated using . in the usual way (e.g., $\lambda_x \mu_a . M$).

Definition 2 (Values)**.** The set of values is the subset of the set M of terms defined as:

$$V ::= x \mid \lambda_x M \mid (c\ V \ldots V) \mid \langle V, V\rangle$$

where c, x, and M are as defined in Definition 1.

Definition 3 (Typing)**.** The set of types T is generated according to the following rules:

$$T ::= A \mid \neg T \mid T \to T \mid T \times T$$

where A is atomic. Typing judgments take the form

$$\Gamma; \Delta \vdash M : T$$

$$x : A; \vdash x : A \qquad \text{(Hyp)}$$

$$\frac{\Gamma, x : A; \Delta \vdash M : B}{\Gamma; \Delta \vdash \lambda_x M : A \to B} \qquad (\text{Abs}_\lambda)$$

$$\frac{\Gamma; \Delta \vdash M : A \to B \qquad \Gamma; \Delta \vdash N : A}{\Gamma; \Delta \vdash (M\ N) : B} \qquad (\text{App}_\lambda)$$

$$\frac{\Gamma; \Delta \vdash M : A \qquad \Gamma; \Delta \vdash N : B}{\Gamma; \Delta \vdash \langle M, N \rangle : A \times B} \qquad \text{(Prod)}$$

$$\frac{\Gamma; \Delta \vdash M : T_0 \times T_1}{\Gamma; \Delta \vdash (\pi_i\ M) : T_i} \qquad (\text{Proj}_i\ (i \in \{0, 1\}))$$

Figure 1: Typing rules of typed λ-calculus.

and are interpreted as saying that the term M is of type T in the context Γ (of λ-variables) and co-context Δ (of μ-variables).

The typing rules for introducing hypothetical λ-variables, λ-abstraction, λ-application, and product are preserved from typed λ-calculus (see Figure 1). Extended rules for abstraction and application with respect to μ-variables and a special reset rule are added to these rules (Figure 2), where the type $\mathcal{R}$ is special but not atomic (cf. Parigot, 2000). Letting $\mathcal{R} = \bot$, there is a

$$\frac{\Gamma; \Delta, a : \neg A \vdash M : \mathcal{R}}{\Gamma; \Delta \vdash \mu_a M : A} \qquad (\text{Abs}_\mu)$$

$$\frac{\Gamma; \Delta \vdash M : A}{\Gamma; \Delta, a : \neg A \vdash (a\ M) : \mathcal{R}} \qquad (\text{App}_\mu)$$

$$\frac{\Gamma; \Delta \vdash M : \mathcal{R}}{\Gamma; \Delta \vdash \lceil M \rfloor : \mathcal{R}} \qquad \text{(Res)}$$

Figure 2: Additional typing rules of $\lambda\mu_V$.

metaphorical resemblance (first noticed by Griffin, 1990) between types of $\lambda\mu_V$ and formulas of classical propositional logic.

Definition 4 (Reduction)**.** In the reduction rules for $\lambda\mu_V$, we use the notational conventions that

- $L[M/N]$ is the usual capture-avoiding substitution where M replaces free occurrences of N in L, and
- $L[M/^*N]$ is the term obtained by replacing every instance of a term of the form N with M in L.

The rule β_V is the same as for ordinary typed λ-calculus, with the proviso that the argument in a β-reduction must be a value. The extended reduction rules of $\lambda\mu_V$ are given in Figure 3.

$$
\begin{array}{rll}
\beta_V\colon & & \\
 & (\lambda_x M\ V) \rhd M[V/x] & (V \text{ a value}) \\
\pi\colon & & \\
 & (\pi_i\ \langle M_0, M_1\rangle) \rhd M_i & (i \in \{0,1\}) \\
\mu\colon & & \\
 & (\mu_a M\ N) \rhd \mu_b M[(b\ (L\ N))/^*(a\ L)] & \\
\mu_V\colon & & \\
 & (V\ \mu_a M) \rhd \mu_b M[(b\ (V\ L))/^*(a\ L)] & (V \text{ a value}) \\
\text{Rename}\colon & & \\
 & (a\ \mu_b M) \rhd M[a/b] & \\
\eta_\mu\colon & & \\
 & \mu_a(a\ M) \rhd M & (a \text{ does not occur in } M) \\
\text{Reset}_\mu\colon & & \\
 & \lceil \mu_a M \rfloor \rhd \lceil M[N/^*(a\ N)] \rfloor & \\
\text{Reset}_V\colon & & \\
 & \lceil V \rfloor \rhd V & (V \text{ a value})
\end{array}
$$

Figure 3: Reduction rules for $\lambda\mu_V$.

We will sometimes write $M \rhd^+ N$ when M reduces to N after multiple applications of these reduction rules.

2 Using $\lambda\mu_V$ to Model Discourse

Like the CPS version (Martin and Pollard, 2009), our adaptation of de Groote's direct-style dynamic logic is based on the hyperintensional semantics of Pollard (2008).

2.1 Types

The type system is almost the same as for the CPS version. The basic types we will use are:

e (entities)
t (truth values)
p (propositions)

We let the special type $\mathcal{R}$ be p, the type of propositions. Using the type constructors $\to$ and $\times$, we define:

$$\begin{aligned}
\epsilon &=_{\text{def}} \text{e} \to \text{t} && \text{(sets of entities)} \\
\rho &=_{\text{def}} \{r \in \text{e} \to \text{e} \to \text{t} \mid r \text{ is a preorder}\} && \text{(resolutions)} \\
\sigma &=_{\text{def}} \epsilon \times \rho \times \text{p} && \text{(information structures)} \\
\pi &=_{\text{def}} \sigma \to \sigma && \text{(dynamic propositions)}
\end{aligned}$$

Our structures are an enrichment of de Groote's, which are just sets of entities. Following de Groote, a dynamic proposition is now simply a "state transformer" on structures.

Dynamic relations are recursively defined by

$$\begin{aligned}
\delta_0 &=_{\text{def}} \pi && \text{(nullary relations)} \\
\delta_{(\mathbf{suc}\, n)} &=_{\text{def}} \text{e} \to \delta_n && (n\text{-ary relations, where } n > 0)
\end{aligned}$$

where $\lambda_n.(\mathbf{suc}\, n) : \omega \to \omega$ yields any natural number's successor. We abbreviate δ_1 as simply δ.

2.2 Nonlinguistic Constants

We define some helpful functions for working with structures:

$$\begin{aligned}
&\mathbf{e} : \sigma \to \epsilon && \text{(first projection of } \sigma) \\
&\mathbf{r} : \sigma \to \rho && \text{(second projection of } \sigma) \\
&\mathbf{c} : \sigma \to \text{p} && \text{(third projection of } \sigma) \\
&\bullet : \epsilon \to \text{e} \to \epsilon && \text{(extends a set of entities)} \\
&* : \rho \to \text{e} \to \rho && \text{(noncommittally extends a resolution)}
\end{aligned}$$

The functions $\bullet$ (analog of de Groote's ::) and $*$ are written infix, the projections functions are written prefix.

2.2.1 Extending Information Structures

To extend an information structure with a new entity, we replace de Groote's :: with **intro**:

$$\mathbf{intro} =_{\text{def}} \lambda_{sx} \langle (\mathbf{e}\, s) \bullet x, (\mathbf{r}\, s) * x, (\mathbf{c}\, s) \rangle : \sigma \to \text{e} \to \sigma \tag{1}$$

We add the ability to update the common ground of a structure with a new proposition with **upd**:

$$\mathbf{upd} =_{\text{def}} \lambda_{sp} \langle (\mathbf{e}\, s), (\mathbf{r}\, s), (\mathbf{c}\, s) \wedge p \rangle : \sigma \to \text{p} \to \sigma \tag{2}$$

where $\wedge$ is propositional, not boolean, conjunction.

2.2.2 Dynamic Relations

For a functions denoting an n-ary (static) relation R_n on entities, the corresponding dynamic relation ($\mathbf{dyn}\, R_n$) is defined as:

$$(\mathbf{dyn}\, R_n) =_{\text{def}} \lambda_{x_1 \ldots x_n s} \mu_k.(R\, x_1 \ldots x_n) \wedge (k\, (\mathbf{upd}\, s\, (R\, x_1 \ldots x_n))) \tag{3}$$

so that the common ground is always updated (via **upd**) with the relation's content. Examples:

$$\begin{aligned}
(\mathbf{dyn}\, \mathsf{rain}) &= \lambda_s \mu_k.\mathsf{rain} \wedge (k\, (\mathbf{upd}\, s\, \mathsf{rain})) : \pi \\
(\mathbf{dyn}\, \mathsf{donkey}) &= \lambda_{xs} \mu_k.(\mathsf{donkey}\, x) \wedge (k\, (\mathbf{upd}\, s\, (\mathsf{donkey}\, x))) : \delta \\
(\mathbf{dyn}\, \mathsf{bray}) &= \lambda_{xs} \mu_k.(\mathsf{bray}\, x) \wedge (k\, (\mathbf{upd}\, s\, (\mathsf{bray}\, x))) : \delta
\end{aligned}$$

2.2.3 Contributed Content

To get at the contributed (static) content of a dynamic proposition, we use **cont**, a direct analog of de Groote's READ:

$$\mathbf{cont} =_{\text{def}} \lambda_{sA} \lceil (\lambda_s \mathsf{true}\, (A\, s)) \rfloor : \sigma \to \pi \to \mathrm{p} \tag{4}$$

Example use of **cont** with RAIN = ($\mathbf{dyn}\, \mathsf{rain}$) and a hypothetical structure s:

$$\begin{aligned}
(\mathbf{cont}\, s\, \textsc{rain}) &= (\mathbf{cont}\, s\, \lambda_s \mu_k.\mathsf{rain} \wedge (k\, (\mathbf{upd}\, s\, \mathsf{rain}))) && (5) \\
&= \lceil (\lambda_s \mathsf{true}\, (\lambda_s \mu_k.\mathsf{rain} \wedge (k\, (\mathbf{upd}\, s\, \mathsf{rain}))\, s)) \rfloor \\
&\rhd \lceil (\lambda_s \mathsf{true}\, \mu_k.\mathsf{rain} \wedge (k\, (\mathbf{upd}\, s\, \mathsf{rain}))) \rfloor && (\beta_V) \\
&\rhd \lceil \mu_k.\mathsf{rain} \wedge (k\, (\lambda_s \mathsf{true}\, (\mathbf{upd}\, s\, \mathsf{rain}))) \rfloor && (\mu_V) \\
&\rhd \lceil \mu_k.\mathsf{rain} \wedge (k\, \mathsf{true}) \rfloor && (\beta_V) \\
&\rhd \lceil \mathsf{rain} \wedge \mathsf{true} \rfloor && (\mathrm{Reset}_\mu) \\
&\rhd \mathsf{rain} \wedge \mathsf{true} && (\mathrm{Reset}_V) \\
&\equiv \mathsf{rain} && (\text{propositional equivalence})
\end{aligned}$$

2.3 Dynamic Logic

We adapt de Groote's work as follows:

- The connectives are mnemonically-named propositional analogs of de Groote's boolean ones.
- In the quantifiers, de Groote's :: is replaced with our **intro** from Equation (1).
- We define dynamic implication ($\Rightarrow$) based on de Morgan's laws (see Equation (8)).

In the following definitions, $\wedge$, $\exists$, $\neg$, and $\forall$ are propositional, not boolean:

$$\textsc{and} =_{\text{def}} \lambda_{ABs}.B\, (A\, s) : \pi \to \pi \to \pi \tag{6}$$

$$\textsc{not} =_{\text{def}} \lambda_{As} \mu_k.(\neg\, (\mathbf{cont}\, s\, A)) \wedge (k\, s) : \pi \to \pi \tag{7}$$

$$\Rightarrow =_{\text{def}} \lambda_{AB}.\textsc{not}\, (A\ \textsc{and}\ (\textsc{not}\, B)) : \pi \to \pi \to \pi \tag{8}$$

$$\textsc{exists} =_{\text{def}} \lambda_{Ds} \mu_k.\exists\, \lambda_x (k\, (D\, x\, (\mathbf{intro}\, s\, x))) : \delta \to \pi \tag{9}$$

$$\textsc{forall} =_{\text{def}} \lambda_{Ds} \mu_k.(\forall\, \lambda_x (\mathbf{cont}\, (\mathbf{intro}\, s\, x)\, (D\, x))) \wedge (k\, s) : \delta \to \pi \tag{10}$$

The connectives AND and $\Rightarrow$ are written infix, all others are written prefix.

Example 1 (Sucky Weather)**.** The connective AND is used to model both sentence conjunction and declarative sentences that occur in sequence. Both of the examples in (1) have the same translation:

(1) a. It rained and it snowed.
b. It rained. It snowed.

In (11), RAIN = (**dyn** rain) and SNOW = (**dyn** snow) both update the common ground:

$$
\begin{aligned}
&(\text{RAIN AND SNOW}) : \pi && (11)\\
&= \lambda_s(\text{SNOW}\ (\text{RAIN}\ s))\\
&= \lambda_s(\lambda_{s_2}\mu_{k_2}(\mathsf{snow} \wedge (k_2\ (\mathbf{upd}\ s_2\ \mathsf{snow})))\ (\lambda_{s_1}\mu_{k_1}(\mathsf{rain} \wedge (k_1\ (\mathbf{upd}\ s_1\ \mathsf{rain})))\ s))\\
&\rhd\ \lambda_s(\lambda_{s_2}\mu_{k_2}(\mathsf{snow} \wedge (k_2\ (\mathbf{upd}\ s_2\ \mathsf{snow})))\ \mu_{k_1}.\mathsf{rain} \wedge (k_1\ (\mathbf{upd}\ s\ \mathsf{rain}))) && (\beta_V)\\
&\rhd\ \lambda_s\mu_{k_1}.\mathsf{rain} \wedge (k_1\ (\lambda_{s_2}\mu_{k_2}(\mathsf{snow} \wedge (k_2\ (\mathbf{upd}\ s_2\ \mathsf{snow})))\ (\mathbf{upd}\ s\ \mathsf{rain}))) && (\mu_V)\\
&\rhd\ \lambda_s\mu_{k_1}.\mathsf{rain} \wedge (k_1\ \mu_{k_2}.\mathsf{snow} \wedge (k_2\ (\mathbf{upd}\ (\mathbf{upd}\ s\ \mathsf{rain})\ \mathsf{snow}))) && (\beta_V)\\
&\rhd\ \lambda_s\mu_{k_1}.\mathsf{rain} \wedge \mathsf{snow} \wedge (k_1\ (\mathbf{upd}\ (\mathbf{upd}\ s\ \mathsf{rain})\ \mathsf{snow})) && (\text{Rename})
\end{aligned}
$$

Example 2 (The Silence of the Donkeys)**.** An existential can occur inside the scope of a negation:

(2) No way a donkey brays.

We need to constrain the scope of the entity introduced by the indefinite determiner *a* while still allowing presuppositions to pass through. We use a propositional analog of de Groote's indefinite:

$$\text{A} =_{\text{def}} \lambda_{DE}.\text{EXISTS}\ \lambda_x.(D\ x)\ \text{AND}\ (E\ x) : \delta \to \delta \to \pi \tag{12}$$

With DONKEY = (**dyn** donkey) and BRAY = (**dyn** bray), the negated portion of the utterance is

$$
\begin{aligned}
&((\text{A DONKEY})\ \text{BRAY}) : \pi && (13)\\
&= (\text{EXISTS}\ \lambda_x.(\text{DONKEY}\ x)\ \text{AND}\ (\text{BRAY}\ x))\\
&= \lambda_s\mu_k.\exists\ \lambda_x(k\ (\lambda_x((\text{DONKEY}\ x)\ \text{AND}\ (\text{BRAY}\ x))\ x\ (\mathbf{intro}\ s\ x)))\\
&\rhd^+\ \lambda_s\mu_k.\exists\ \lambda_x.(\mathsf{donkey}\ x) \wedge (\mathsf{bray}\ x) \wedge (k\ (\mathbf{upd}\ (\mathbf{upd}\ (\mathbf{intro}\ s\ x)\ (\mathsf{donkey}\ x))\ (\mathsf{bray}\ x)))
\end{aligned}
$$

Translating *no way* as NOT, the dynamic meaning of (2) is

$$
\begin{aligned}
&(\text{NOT}\ ((\text{A DONKEY})\ \text{BRAY})) : \pi && (14)\\
&= \lambda_s\mu_k.(\neg\ (\mathbf{cont}\ s\ ((\text{A DONKEY})\ \text{BRAY}))) \wedge (k\ s)\\
&\rhd^+\ \lambda_s\mu_k.(\neg\ (\exists\ \lambda_x((\mathsf{donkey}\ x) \wedge (\mathsf{bray}\ x) \wedge \mathsf{true}))) \wedge (k\ s)\\
&\equiv \lambda_s\mu_k.(\neg\ (\exists\ \lambda_x((\mathsf{donkey}\ x) \wedge (\mathsf{bray}\ x)))) \wedge (k\ s)
\end{aligned}
$$

So negation restricts the scope of existential quantification while being a "hole" for presupposition.

Example 3 (Enter the Donkey)**.** In the discourse in (3), the noun phrase *the donkey* can only refer to one of the discourse referents introduced prior to its use:

(3) a. A donkey enters.
b. A mule enters.
c. The donkey brays.

To capture the presuppositions associated with the English definite determiner *the*, we introduce a definiteness operator to select the "highest" entity in the resolution of a structure s with the property $(\mathbf{cont}\ (D\ x))$ for a hypothetical dynamic property D and entity x:

$$\mathbf{def} =_{\mathrm{def}} \lambda_{sD}.\bigsqcup_{(\mathbf{r}\ s)} \lambda_x.(\mathbf{c}\ s) \to (\mathbf{cont}\ (D\ x)) : \sigma \to \delta \to \mathrm{e} \tag{15}$$

The definite determiner[1] uses **def** to select the right discourse referent from a structure s:

$$\textsc{the} =_{\mathrm{def}} \lambda_{DEs}\mu_k.k\ (\lambda_x((D\ x)\ \textsc{and}\ (E\ x))\ (\mathbf{def}\ s\ D)\ s) : \delta \to \delta \to \pi \tag{16}$$

With $\textsc{mule} = (\mathbf{dyn}\ \mathsf{mule})$, $\textsc{enter} = (\mathbf{dyn}\ \mathsf{enter})$, and A, DONKEY, and BRAY as for Example 2, we can now model the discourse in (3) as

$$((((\textsc{a donkey})\ \textsc{enter})\ \textsc{and}\ ((\textsc{a mule})\ \textsc{enter}))\ \textsc{and}\ ((\textsc{the donkey})\ \textsc{bray})) \tag{17}$$

We start with the leftmost conjunct, which reduces similarly to (13):

$$\begin{aligned} &((\textsc{a donkey})\ \textsc{enter}) : \pi \\ \rhd^+\ &\lambda_s\mu_k.\exists\ \lambda_x.(\mathsf{donkey}\ x) \wedge (\mathsf{enter}\ x) \wedge (k\ (\mathbf{upd}\ (\mathbf{upd}\ (\mathbf{intro}\ s\ x)\ (\mathsf{donkey}\ x))\ (\mathsf{enter}\ x))) \end{aligned} \tag{18}$$

Continuing with the entire left conjunct of the discourse:

$$\begin{aligned} &(((\textsc{a donkey})\ \textsc{enter})\ \textsc{and}\ ((\textsc{a mule})\ \textsc{enter})) : \pi \\ \rhd^+\ &\lambda_s\mu_k.\exists\ \lambda_x.(\mathsf{donkey}\ x) \wedge (\mathsf{enter}\ x) \wedge (\exists\ \lambda_y.(\mathsf{mule}\ y) \wedge (\mathsf{enter}\ y) \wedge (k\ \varsigma)) \end{aligned} \tag{19}$$

where ς represents the structure that results from this application of AND:

$$\varsigma = (\mathbf{upd}\ (\mathbf{upd}\ (\mathbf{intro}\ (\mathbf{upd}\ (\mathbf{upd}\ (\mathbf{intro}\ s\ x)\ (\mathsf{donkey}\ x))\ (\mathsf{enter}\ x))\ y)\ (\mathsf{mule}\ y))\ (\mathsf{enter}\ y)) \tag{20}$$

The right conjunct then picks out the most salient DONKEY from the preceding discourse:

$$\begin{aligned} &((\textsc{the donkey})\ \textsc{bray}) : \pi \\ =\ &\lambda_s\mu_k.k\ (\lambda_x((\textsc{donkey}\ x)\ \textsc{and}\ (\textsc{bray}\ x))\ (\mathbf{def}\ s\ \textsc{donkey})\ s) \\ \rhd\ &\lambda_s\mu_k.k\ (((\textsc{donkey}\ (\mathbf{def}\ s\ \textsc{donkey}))\ \textsc{and}\ (\textsc{bray}\ (\mathbf{def}\ s\ \textsc{donkey})))\ s) \qquad (\beta_V) \\ \rhd^+\ &\lambda_s\mu_k.(\mathsf{donkey}\ (\mathbf{def}\ s\ \textsc{donkey})) \wedge (\mathsf{bray}\ (\mathbf{def}\ s\ \textsc{donkey})) \\ &\quad \wedge (k\ (\mathbf{upd}\ (\mathbf{upd}\ s\ (\mathsf{donkey}\ (\mathbf{def}\ s\ \textsc{donkey})))\ (\mathsf{bray}\ (\mathbf{def}\ s\ \textsc{donkey})))) \end{aligned} \tag{21}$$

With $(\mathbf{def}\ \varsigma\ \textsc{donkey}) = x$, the term in (17) reduces as:

$$\lambda_s\mu_k.\exists\ \lambda_x.(\mathsf{donkey}\ x) \wedge (\mathsf{enter}\ x) \wedge (\exists\ \lambda_y.(\mathsf{mule}\ y) \wedge (\mathsf{enter}\ y) \wedge (\mathsf{donkey}\ x) \wedge (\mathsf{bray}\ x) \wedge (k\ \varsigma')) \tag{22}$$

where $\varsigma' = (\mathbf{upd}\ (\mathbf{upd}\ \varsigma\ (\mathsf{donkey}\ x))\ (\mathsf{bray}\ x))$ is the new structure resulting from applying AND to (19) and (21).

Example 4 (An Asinine Denial)**.** We want the meaning of *every* to be a universally quantified conditional relation between two dynamic properties. This relation should additionally allow the consequent to access the information structure updated by the antecedent. For example, in

[1] Hat tip to Michael White for noticing an egregious flaw in an earlier version of the definite determiner.

(4) Every donkey denies it brays.

the pronoun *it* is "anteceded by" the noun phrase *every donkey*.

Our dynamic meaning of "every" is analogous to de Groote's:

$$\textsc{every} =_{\text{def}} \lambda_{DE}.\textsc{forall}\ \lambda_x.(D\ x) \Rightarrow (E\ x) : \delta \to \delta \to \pi \tag{23}$$

Rather than adopt de Groote's `sel`, we translate *it* to capture its presuppositions:

$$\textsc{it} =_{\text{def}} \lambda_{Ds}.D\ (\mathbf{def}\ s\ \textsc{nonhuman})\ s : \delta \to \pi \tag{24}$$

where $\textsc{nonhuman} = (\mathbf{dyn}\ \mathsf{nonhuman})$. With *denies* translated as

$$\textsc{deny} =_{\text{def}} \lambda_{Axs}\mu_k.(\mathsf{deny}\ x\ (\mathbf{cont}\ s\ A)) \wedge (k\ (\mathbf{upd}\ s\ (\mathsf{deny}\ x\ (\mathbf{cont}\ s\ A)))) : \pi \to \delta \tag{25}$$

and DONKEY and BRAY as in Example 2, we are in a position to give the dynamic meaning of (4):

$$\begin{aligned}
&(\textsc{every donkey}\ (\textsc{deny}\ (\textsc{it bray}))) : \pi \qquad (26)\\
&= (\textsc{every donkey}\ (\textsc{deny}\ \lambda_s(\textsc{bray}\ (\mathbf{def}\ s\ \textsc{nonhuman})\ s)))\\
&= (\textsc{forall}\ \lambda_x.(\textsc{donkey}\ x) \Rightarrow (\textsc{deny}\ \lambda_s(\textsc{bray}\ (\mathbf{def}\ s\ \textsc{nonhuman})\ s)\ x))\\
&= (\textsc{forall}\ \lambda_x.\textsc{not}\ ((\textsc{donkey}\ x)\ \textsc{and}\ (\textsc{not}\ (\textsc{deny}\ \lambda_s(\textsc{bray}\ (\mathbf{def}\ s\ \textsc{nonhuman})\ s)\ x))))\\
&\rhd^+ \lambda_s\mu_k.(\forall\ \lambda_x(\neg\ ((\mathsf{donkey}\ x)\\
&\qquad \wedge (\neg\ (\mathsf{deny}\ x\ (\mathsf{bray}\ (\mathbf{def}\ (\mathbf{upd}\ (\mathbf{intro}\ s\ x)\ (\mathsf{donkey}\ x))\ \textsc{nonhuman}))\\
&\qquad \wedge \mathsf{true}) \wedge \mathsf{true}) \wedge \mathsf{true}))) \wedge (k\ s)\\
&\equiv \lambda_s\mu_k.(\forall\ \lambda_x(\neg\ ((\mathsf{donkey}\ x)\\
&\qquad \wedge (\neg\ (\mathsf{deny}\ x\ (\mathsf{bray}\ (\mathbf{def}\ (\mathbf{upd}\ (\mathbf{intro}\ s\ x)\ (\mathsf{donkey}\ x))\ \textsc{nonhuman}))))))) \wedge (k\ s)
\end{aligned}$$

With $(\mathbf{def}\ (\mathbf{upd}\ (\mathbf{intro}\ s\ x)\ (\mathsf{donkey}\ x))\ \textsc{nonhuman}) = x$, we have

$$\lambda_s\mu_k.(\forall\ \lambda_x(\neg\ ((\mathsf{donkey}\ x) \wedge (\neg\ (\mathsf{deny}\ x\ (\mathsf{bray}\ x)))))) \wedge (k\ s) \tag{27}$$

which constrains the scope of the universal quantification while passing presuppositions through.

References

Philippe de Groote. Typing binding and anaphora: Dynamic contexts as $\lambda\mu$-terms. Presented at the ESSLLI 2008 Workshop on Symmetric Calculi and Ludics for Semantic Interpretation, 2008. Available at `http://www.ling.ohio-state.edu/research/groups/commies/past/autumn2009/degroote-esslli08.pdf`.

Timothy Griffin. A formulae-as-types notion of control. In *Conference Record of POPL 17*, 1990.

Scott Martin and Carl Pollard. Enriching contexts for type-theoretic dynamics. Presented at the CAuLD Workshop on Logical Methods for Discourse, 2009. Available at `http://www.ling.ohio-state.edu/~scott/talks/cauld/cauld_slides.pdf`.

Michel Parigot. $\lambda\mu$-calculus: An algorithmic interpretation of classical natural deduction. *Lecture Notes in Computer Science*, 624:190–201, 1992.

Michel Parigot. On the computational interpretation of negation. *Lecture Notes in Computer Science*, 1862:472–484, 2000.

Carl Pollard. Hyperintensions. *Journal of Logic and Computation*, 18(2):257–282, 2008.