Decomposing Montagovian Dynamics

Greg Kobele

May 5, 2016

1 Introduction

de Groote [2006] demonstrates how to give a dynamic interpretation to formulae in higher order logic, not by, as is done in most works on dynamic semantics, giving a non-standard model-theoretic interpretation of such formulae, but rather by giving a systematic syntactic translation of these formulae into other formulae which, when interpreted in the standard way, are equivalent to the dynamic interpretation of the original formulae. This is achieved by lifting the types of expressions to be functions from left contexts (representing information about already established discourse referents) to functions from right contexts (representing discourse continuations) to truth values.

The goal of this short note is to show that the dynamic translation of de Groote [2006] can be factored into the composition of two simpler parts; a left-context translation and a right-context translation, or, more precisely, a semantics dealing with dynamism without discourse referents, and a semantics dealing with discourse referents without dynamism.

This is a literate Coq document, with source code available on the web at `https://github.com/gkobele/decomposing-montague`.

2 The formal language

I begin by defining the basic structures of the language. This will serve both as a metalanguage (the target language of the translations) and also as an object language (the source languages of the translations).

The basic idea of a translation procedure is that an uninterpreted object language is translated into a language that is already understand, the meta-language. The same syntactic forms will be used (a simple fragment of the lambda calculus) for both. This offers a sort of *modularity* of semantic description. From the semanticist's perspective, this means that certain aspects of meaning can be systematically ignored, knowing that the semantic terms actually written down are (via the translation process) notational shorthands for richer meaning representations. More speculatively, one might conjecture that each successive translation might have some sort of deeper reality, perhaps recapitulating the stages of a

$\top$	*true*	*farmer*	$p0$
$\bot$	*false*	*donkey*	$p1$
$\neg$	*not*	*laugh*	$p2$
$\wedge$	*and*	*bray*	$p3$
$\vee$	*or*	*own*	$r0$
$\leadsto$	*imply*	*beat*	$r1$

Figure 1: Abbreviations for constants

learner's semantic language acquisition, or perhaps even as reflective of different levels of deep semantic analysis of a sentence made as needed during language processing.

The language is typed, with two basic types; e and t. Nothing is assumed about expressions of type e; in particular, no expression of the language will have a type ending in e. In contrast, the type t, is viewed as the type of complete propositions, is the locus of recursion in this language. Thus, *and* is a binary term forming operator. There are infinitely many inhabitants of type t, many of which are intuitively semantically equivalent. There are constants *true* and *false*, boolean connectives *and*, *or* and *not*, term forming operators $p\ n$ and $r\ n$ (the nth unary and binary relations respectively), and generalized quantifiers *some*, *all* and *pro* (representing a pronoun). These provide the basic alphabet for building semantic terms (of type t). With the exception of *pro*, these can be interpreted in the usual way in higher order models. *pro* is not (intended to be understood as) a term in the meta-language, and will only be given a meaning during the translation process. de Groote [2006] also has a constant *who*, which is however definable as usual in terms of property conjunction: **who** $P\ Q = \lambda x.Px \wedge Qx$.

`Inductive` t : `Set` :=
| *true* : t
| *false* : t
| p : $nat \to e \to t$
| r : $nat \to e \to e \to t$
| *and* : $t \to t \to t$
| *or* : $t \to t \to t$
| *imply* : $t \to t \to t$
| *not* : $t \to t$
| *some* : $(e \to t) \to t$
| *all* : $(e \to t) \to t$
| *pro* : $(e \to t) \to t$.

To make terms more readable, I introduce familiar notation, as shown in figure 1.

`Example` *Every_donkey_brayed* : ***all*** (`fun` $x \Rightarrow$ *donkey* $x \leadsto$ *bray* x) = ***all*** (`fun` $x \Rightarrow (p\ 1\ x) \leadsto (p\ 3\ x)$).

The intended interpretation of terms of type t is of course in an algebra (boolean or heyting) where $\top$ is understood as truth (the top element of the algebra), and $\bot$ as falsity (the bottom element of the algebra). The connective symbols are to be understood as appropriate operations in the algebra (meet, join, and complement if in a boolean algebra, and meet, join and relative complement in a heyting algebra). Although this setup is independent of one's particular choice of logic (classical or intuitionistic), classical logic is more familiar in linguistics, and generates a coarser equivalence relation over terms. I will write $\phi \equiv \psi$ to indicate that ϕ is classically equivalent to ψ. The notation $i\hat{}\hat{}n\ o$ abbreviates the type $\underbrace{i \to \cdots \to i}_{n\ times} \to o$.

3 Dynamism

In this section, dynamism is introduced to the system above. Although the dynamic properties of connectives are typically argued for based on the behaviour of pronouns, the statement of the dynamic properties of connectives needn't make reference to pronouns, which is exploited here.

The intuitive understanding of the dynamic meaning of a sentence is as explicating its contribution to the discourse; it specifies how the meaning of the upcoming discourse is influenced by its own meaning. Accordingly the dynamic type Ω of a sentence is $t \to t$; a function which modifies the meaning of an upcoming discourse. Dynamic translations of terms of type t into terms of type Ω are now defined. These terms of type Ω are intended to capture in an intuitive way the *dynamic* meaning of the original terms of type t.

Atomic sentences, qua n-ary predicates, are simply conjoined with the discourse continuation. That is, the meaning of a discourse consisting of only atomic sentences is hereby stipulated to be their conjunction.

```
Fixpoint atom_d_tr (n : nat) : (e^^n) t → (e^^n) Ω :=
  match n with
  | O ⇒ fun atom φ ⇒ atom ∧ φ
  | S m ⇒ fun atom x ⇒ atom_d_tr m (atom x)
  end.
```

```
Example dynamic_beats : atom_d_tr 2 beat = fun x y φ ⇒ beat x y ∧ φ.
```

The atomic sentences $\top$ and $\bot$, representing the always true and always false sentences, obtain from the general treatment of atomic sentences above the meanings `fun` $\phi \Rightarrow \top \wedge \phi$ and `fun` $\phi \Rightarrow \bot \wedge \phi$ respectively, which are semantically equivalent to the identity function and the constant false function respectively. To simplify the statement of the main result, the translations of $\top$ and $\bot$ are simply taken to be these simpler functions.

```
Definition true_d_tr : Ω := fun φ ⇒ φ.
```

```
Example true_id : ∀ φ, atom_d_tr O ⊤ φ ≡ true_d_tr φ.
```

```
Definition false_d_tr : Ω := fun _ ⇒ ⊥.
```

```
Example false_false : ∀ φ, atom_d_tr O ⊥ φ ≡ false_d_tr φ.
```

Connectives are classified as externally dynamic just in case a discourse referent introduced internally to them is accessible externally. They are internally dynamic just in case a discourse referent introduced inside of one conjunct is accessible in the other.

As conjunction is externally dynamic, the discourse continuation of a conjunction should be in the scope of both conjuncts. As it is also internally dynamic, the second conjunct should be in the scope of the first. The discoure continuation of the first conjunct P is thus the result of continuing the second conjunct Q with the discourse continuation of the entire coordination, ϕ. In other words, a discourse $P \wedge Q.\ D$ is interpreted as $P.\ Q.\ D$.

`Definition` $\mathit{and\_d\_tr} : \Omega \to \Omega \to \Omega :=$
  `fun` $(P\ Q : \Omega)\ (\phi : t) \Rightarrow P\ (Q\ \phi)$.

Negation is externally static (i.e. not externally dynamic); so the discourse continuation of a negated expression P must not be in its scope. Instead, the negated expression P is given a trivial discourse continuation, $\top$, and its negation is treated as an atomic proposition; i.e. it is conjoined with the remainder of the discourse.

`Definition` $\mathit{not\_d\_tr} : \Omega \to \Omega :=$
  `fun` $(P : \Omega) \Rightarrow \mathit{atom\_d\_tr}\ 0\ (\neg\ (P\ \top))$.

`Example` $\mathit{not\_d\_tr\_externally\_static}$:
  $\forall\ (P : \Omega)\ (\phi : t),\ \mathit{not\_d\_tr}\ P\ \phi = \neg\ (P\ \top) \wedge \phi$.

The other dynamic connectives will be defined in terms of dynamic conjunction and dynamic negation. Dynamic behaviour, whether internal or external, will implicate a dynamic conjunction, and static behaviour a dynamic negation.

Implication is externally static, but internally dynamic from antecedent P to consequent Q. This is captured by its classically valid reformulation in terms of $\wedge$ and $\neg$: $P \to Q \equiv \neg(P \wedge \neg Q)$

`Definition` $\mathit{imp\_d\_tr} : \Omega \to \Omega \to \Omega :=$
  `fun` $(P\ Q : \Omega) \Rightarrow \mathit{not\_d\_tr}\ (\mathit{and\_d\_tr}\ P\ (\mathit{not\_d\_tr}\ Q))$.

The dynamic behaviour of disjunction is less clear [Groenendijk and Stokhof, 1991] . Sentences like: “Either this house doesn’t have a bathroom, or it is hidden” suggest that disjunction should be (at least) internally dynamic, however it is common to assume that disjunction is in fact completely static. This is in fact predicted by its classical reformulation in terms of negation and conjunction: $P \vee Q \equiv \neg(\neg P \wedge \neg Q)$

`Definition` $\mathit{or\_d\_tr} : \Omega \to \Omega \to \Omega :=$
  `fun` $(P\ Q : \Omega) \Rightarrow \mathit{not\_d\_tr}\ (\mathit{and\_d\_tr}\ (\mathit{not\_d\_tr}\ P)\ (\mathit{not\_d\_tr}\ Q))$.

The generic treatment of dynamic GQs has it that the discourse continuation of the entire sentence is moved into the scope of the GQ.

`Definition` $\mathit{quant\_d\_tr}\ (q : (e \to t) \to t) : (e \to \Omega) \to \Omega :=$
  `fun` $(P : e \to \Omega)\ (\phi : t) \Rightarrow q$ (`fun` $x \Rightarrow P\ x\ \phi$).

Exceptionally, from this perspective, universal quantification is externally static, which is captured by its de Morgan reformulation in terms of existential quantification: $\forall x.Px \equiv \neg\exists x.\neg Px$

```
Definition all_d_tr : (e → Ω) → Ω :=
  fun (P : e → Ω) ⇒ not_d_tr (quant_d_tr some (fun x ⇒ not_d_tr (P x))).
```

These definitions allow for a concise statement of the translation *d_tr* (mnemonic for *dynamic* translation) from terms of type t, with their dynamic potential *implicit* into terms of type Ω with their dynamic meanings made *explicit*. We see that the translation *d_tr* is just a homomorphism.

```
Fixpoint d_tr (ϕ : t) : Ω :=
  match ϕ with
    | ⊤ ⇒ true_d_tr
    | ⊥ ⇒ false_d_tr
    | p n x ⇒ atom_d_tr 1 (p n) x
    | r n x y ⇒ atom_d_tr 2 (r n) x y
    | ¬ ψ ⇒ not_d_tr (d_tr ψ)
    | ψ ∧ χ ⇒ and_d_tr (d_tr ψ) (d_tr χ)
    | ψ ∨ χ ⇒ or_d_tr (d_tr ψ) (d_tr χ)
    | ψ ⇝ χ ⇒ imp_d_tr (d_tr ψ) (d_tr χ)
    | pro P ⇒ quant_d_tr pro (fun x ⇒ d_tr (P x))
    | some P ⇒ quant_d_tr some (fun x ⇒ d_tr (P x))
    | all P ⇒ all_d_tr (fun x ⇒ d_tr (P x))
  end.
```

This translation, as can be seen by the examples to follow, allows existentials to dynamically extend their scope over upcoming sentences, while disallowing universals from doing the same.

```
Example Some_farmer_laughed :
  ∀ ϕ : t,
  d_tr (some (fun x ⇒ farmer x ∧ laugh x)) ϕ
  ≡ some (fun x ⇒ farmer x ∧ laugh x ∧ ϕ).
```

```
Example Every_farmer_laughed :
  ∀ ϕ : t,
    d_tr (all (fun x ⇒ farmer x ⇝ laugh x)) ϕ
  ≡ (all (fun x : e ⇒ farmer x ⇝ laugh x)) ∧ ϕ.
```

4 Contexts

Here we present an interpretation of context-independent sentences into context *dependent* ones. This does not presuppose (or introduce) dynamism.

Following de Groote [2006], we introduce a new type γ for contexts. This type is *abstract*, in the sense that its properties will be given axiomatically; *anything* that satisfies these properties may be used as a context. We require only that we can interact with objects of type γ in two ways. First, that a context may be updated with an individual to obtain a new context.

```
Variable update : e → γ → γ.
```

The notation $x :: c$ will abbreviate the more cumbersome *update* x c.

And second, that salient individuals may be retrieved from a context.

```
Variable sel : γ → e.
```

In particular, *sel* should be thought of as a placeholder for one's favourite pronoun resolution algorithm; in this view, the job of semantics is to give the appropriate input to the pronoun resolution algorithm, but the inner workings of this algorithm are semantically opaque. de Groote [2006] takes for concreteness γ to be the type of lists of individuals, updating to be achieved by adding an individual to a list, and selection to be retrieving some element of a list.

Terms of type t will be viewed as having implicit contexts. We will define a translation from terms of type t which makes their contexts, context updating, and context passing explicit.

Contexts are incorporated by adding a context parameter to every atomic type.

```
Definition G := γ → t.

Definition E := γ → e.
```

Thus a term of type t will be translated into one of type G, and one of type e will be translated into one of type E.

Inherently context insenstitive n-ary connectives, functions, and predicates can be lifted to context sensitive ones by simply distributing the context to their context-sensitive arguments.

```
Fixpoint distr_g_tr {A B : Type} (n : nat) : (γ → (A ^^ n) B) → ((γ → A) ^^ n) (γ →
B) :=
  match n as n0 return (γ → (A ^^ n0) B) → ((γ → A) ^^ n0) (γ → B) with
  | O ⇒ fun xx ⇒ xx
  | S m ⇒ fun xx φ ⇒ distr_g_tr m (fun c : γ ⇒ xx c (φ c))
  end.

Definition lift_g_tr {A B : Type} (n : nat) (m : (A ^^ n) B) : ((γ → A) ^^ n) (γ → B)
:=
  @distr_g_tr A B n (fun _ ⇒ m).

Example contextual_individual (j : e) : lift_g_tr (A := e) 0 j = (fun _ ⇒ j).

Example contextual_beats : lift_g_tr 2 beat = fun x y c ⇒ beat (x c) (y c).
```

A pronoun *pro* is translated as a generalized quantifier generated from the (context-sensitive) individual *sel*.

```
Definition pro_g_tr : (E → G) → G := fun P ⇒ P sel.
```

The individual argument to the property of a GQ is a lifted context-insensitive variable. The context passed to this property is enriched with the unlifted variable.

```
Definition $quant\_g\_tr$ $(gq : (e \to t) \to t) : (E \to G) \to G :=$
  fun $P\ (c : \gamma) \Rightarrow gq$ (fun $x \Rightarrow P\ (lift\_g\_tr\ (A := e)\ 0\ x)\ (x :: c)$).
```

The function g_tr (mnemonic for *gamma* translation) makes explicit the implicit context manipulation in terms of type t.

```
Fixpoint $g\_tr\ (\phi : t) : G :=$
  match $\phi$ with
    | $\top \Rightarrow lift\_g\_tr\ (A := t)\ 0\ \top$
    | $\bot \Rightarrow lift\_g\_tr\ (A := t)\ 0\ \bot$
    | $\neg\ \psi \Rightarrow lift\_g\_tr\ 1\ \neg\ (g\_tr\ \psi)$
    | $\psi \wedge \chi \Rightarrow lift\_g\_tr\ 2 \wedge (g\_tr\ \psi)\ (g\_tr\ \chi)$
    | $\psi \rightsquigarrow \chi \Rightarrow lift\_g\_tr\ 2 \rightsquigarrow (g\_tr\ \psi)\ (g\_tr\ \chi)$
    | $\psi \vee \chi \Rightarrow lift\_g\_tr\ 2 \vee (g\_tr\ \psi)\ (g\_tr\ \chi)$
    | $p\ n\ a \Rightarrow lift\_g\_tr\ 1\ (p\ n)\ (lift\_g\_tr\ (A := e)\ 0\ a)$
    | $r\ n\ a\ b \Rightarrow lift\_g\_tr\ 2\ (r\ n)\ (lift\_g\_tr\ (A := e)\ 0\ a)\ (lift\_g\_tr\ (A := e)\ 0\ b)$
    | $\textbf{some}\ P \Rightarrow quant\_g\_tr\ \textbf{some}$ (fun $(x : E)\ (c : \gamma) \Rightarrow g\_tr\ (P\ (x\ c))\ c$)
    | $\textbf{all}\ P \Rightarrow quant\_g\_tr\ \textbf{all}$ (fun $(x : E)\ (c : \gamma) \Rightarrow g\_tr\ (P\ (x\ c))\ c$)
    | $pro\ P \Rightarrow pro\_g\_tr$ (fun $(x : E)\ (c : \gamma) \Rightarrow g\_tr\ (P\ (x\ c))\ c$)
  end.
```

In the example below, which is the translation of the sentence *he laughed* into context-sensitive terms, note that the referent of the pronoun *he* must be found in the context of utterance c.

```
Example $he\_laughed$ :
  $\forall\ (c : \gamma),\ g\_tr\ (pro\ \mathsf{laugh})\ c = \mathsf{laugh}\ (sel\ c)$.
```

In the following example, the translation of the sentence *some farmer owned it*, note that the referent of *it* is to be found in the context of utterance c which has been updated with a discourse referent x which is a farmer. Of course, for multiple reasons (the pronoun is incompatible with the animacy of the farmer, and the pronoun is syntactically too local) the pronoun should not in this case be resolved to the discourse referent x.

```
Example $some\_farmer\_owned\_it$ :
  $\forall\ (c : \gamma),\ g\_tr\ (\textbf{some}$ (fun $x \Rightarrow \mathsf{farmer}\ x \wedge (pro$ (fun $y \Rightarrow \mathsf{own}\ y\ x)))) c$
                   $= \textbf{some}$ (fun $x : e \Rightarrow \mathsf{farmer}\ x \wedge \mathsf{own}\ (sel\ (x :: c))\ x$).
```

5 De Groote's Montagovian Dynamics

The type of a sentence ω is defined to be $\gamma \to (\gamma \to t) \to t$; a function from a context γ to the type $(\gamma \to t) \to t$ of the *continuation* of a context.

Atomic predicates do not modify their context; they simply conjoin their static meaning with the meaning of the discourse continuation in the context.

```
Definition $pred\_dg\_tr$ $(n : nat) : e \to \omega :=$
  fun $(x : e)$ $(c : \gamma)$ $(\phi : \gamma \to t) \Rightarrow (p\ n\ x) \wedge (\phi\ c)$.
```

```
Definition $rel\_dg\_tr$ $(n : nat) : e \to e \to \omega :=$
  fun $(x\ y : e)$ $(c : \gamma)$ $(\phi : \gamma \to t) \Rightarrow (r\ n\ x\ y) \wedge (\phi\ c)$.
```

de Groote treats verbs as taking generalized quantifiers as arguments, as opposed to individuals (in contrast to nouns). This decision is orthogonal to the question of interest here, and requires a richer setup than present in his 2006 paper to treat the composition of semantic translations.

While de Groote [2006] does not deal explicitly with sentential connectives, his later work (as described by Lebedeva [2012]) does, and is as follows.

```
Definition $and\_dg\_tr : \omega \to \omega \to \omega :=$
  fun $(P\ Q : \omega)$ $(c : \gamma)$ $(\phi : \gamma \to t) \Rightarrow P\ c$ (fun $d \Rightarrow Q\ d\ \phi$).
```

```
Definition $not\_dg\_tr : \omega \to \omega :=$
  fun $(P : \omega)$ $(c : \gamma)$ $(\phi : \gamma \to t) \Rightarrow (\neg\ (P\ c$ (fun _ $\Rightarrow \top))) \wedge (\phi\ c)$.
```

```
Definition $or\_dg\_tr : \omega \to \omega \to \omega :=$
  fun $(P\ Q : \omega) \Rightarrow not\_dg\_tr\ (and\_dg\_tr\ (not\_dg\_tr\ P)\ (not\_dg\_tr\ Q))$.
```

```
Definition $imply\_dg\_tr : \omega \to \omega \to \omega :=$
  fun $(P\ Q : \omega) \Rightarrow not\_dg\_tr\ (and\_dg\_tr\ P\ (not\_dg\_tr\ Q))$.
```

```
Definition $some\_dg\_tr : (e \to \omega) \to \omega :=$
  fun $(P : e \to \omega)$ $(c : \gamma)$ $(\phi : \gamma \to t) \Rightarrow$ some (fun $(x : e) \Rightarrow P\ x\ (x :: c)\ \phi$).
```

```
Definition $every\_dg\_tr : (e \to \omega) \to \omega :=$
  fun $(P : e \to \omega) \Rightarrow not\_dg\_tr\ (some\_dg\_tr$ (fun $x \Rightarrow not\_dg\_tr\ (P\ x)$)).
```

```
Definition $pro\_dg\_tr : (e \to \omega) \to \omega :=$
  fun $(P : e \to \omega)$ $(c : \gamma)$ $(\phi : \gamma \to t) \Rightarrow P\ (sel\ c)\ c\ \phi$.
```

The function *dg_tr* (short for 'De Groote' translation), makes the implicit dynamism and context manipulation in terms explicit.

```
Fixpoint $dg\_tr\ (f : t) : \omega :=$
  match $f$ with
    | $\top \Rightarrow$ fun $c\ \phi \Rightarrow \phi\ c$
    | $\bot \Rightarrow$ fun _ _ $\Rightarrow \bot$
    | $P \wedge Q \Rightarrow and\_dg\_tr\ (dg\_tr\ P)\ (dg\_tr\ Q)$
    | $P \vee Q \Rightarrow or\_dg\_tr\ (dg\_tr\ P)\ (dg\_tr\ Q)$
    | $P \rightsquigarrow Q \Rightarrow imply\_dg\_tr\ (dg\_tr\ P)\ (dg\_tr\ Q)$
    | $\neg\ P \Rightarrow not\_dg\_tr\ (dg\_tr\ P)$
    | some $P \Rightarrow some\_dg\_tr$ (fun $x \Rightarrow dg\_tr\ (P\ x)$)
    | all $P \Rightarrow every\_dg\_tr$ (fun $x \Rightarrow dg\_tr\ (P\ x)$)
    | $p\ n\ x \Rightarrow pred\_dg\_tr\ n\ x$
```

    $\mid r\ n\ x\ y \Rightarrow rel\_dg\_tr\ n\ x\ y$
    $\mid pro\ P \Rightarrow pro\_dg\_tr$ (fun $x \Rightarrow dg\_tr\ (P\ x)$)
  end.

Example $if\_a\_farmer\_owns\_a\_donkey\_he\_beats\_it : \forall\ c\ \phi,$
    dg_tr ((**some** (fun $x \Rightarrow$ farmer $x \wedge$ **some** (fun $y \Rightarrow$ donkey $y \wedge$ own $y\ x$))) $\rightsquigarrow$ pro (fun $u \Rightarrow pro$ (fun $v \Rightarrow$ beat $v\ u$))) $c\ \phi$
        $\equiv$
    ((**all** (fun $x : e \Rightarrow$ farmer $x \rightsquigarrow$
        all (fun $y : e \Rightarrow$ (donkey $y \wedge$ own $y\ x$) $\rightsquigarrow$
            beat (sel ($y :: x :: c$)) (sel ($y :: x :: c$)))))
      $\wedge\ \phi\ c$).

6 The equivalence

The translations must be restricted so as to use the same type γ and same accessor functions *update* and *sel*.

Definition $g\_trans := g\_tr\ \gamma\ update\ sel.$

Definition $dg\_trans := dg\_tr\ \gamma\ update\ sel.$

In order to prove the equivalence between de Groote's translation and the composition of the gamma and dynamic translations, I assume that functions are extensional; that two functions are identical iff they compute the same input-output relation.

Hypothesis $fun\_ext : \forall\ (A\ B : \texttt{Set})\ (f\ g : A \rightarrow B), (\forall\ x, f\ x = g\ x) \rightarrow f = g.$

The main theorem states that, for any formula ϕ of type t, the context-change poential of the contextification of its dynamic translation is identical to the context-change potential of de Groote's translation of it.

Theorem dg_g_d :
  $\forall\ (\phi\ \psi : t),\ g\_trans\ ((d\_tr\ \phi)\ \psi) =$ fun $c \Rightarrow (dg\_trans\ \phi)\ c\ (g\_trans\ \psi).$

Proof. This is proven by induction on ϕ, followed by simple but tedious case analysis. □

One peculiarity about the statement of the theorem concerns the function *g_trans* on the right hand side of the equation. This is because the discourse context desired on the left hand side (ψ) is of type t, whereas that desired on the right ($g\_trans\ \psi$) is of type $\gamma \rightarrow t$. In order for the theorem to make sense, both sides must be given the same discourse continuation. The term *g_trans* on the right of the equation coerces a term of type t into one of type $\gamma \rightarrow t$. In essence, the theorem says only that *g_trans* composed with *d_tr* has the same effect on a discourse configuration of type t that *dg_trans* has on its image under *g_trans*.

In practice, this is not much of a restriction, as for any discourse ϕ, the theorem implies that both left and right sides are interpreted identically in the empty discourse continuation.

Corollary $dg\_g\_d\_true$:
  $\forall\ (\phi\ :\ t),\ g\_trans\ (d\_tr\ \phi\ \top) = \texttt{fun}\ c \Rightarrow dg\_trans\ \phi\ c\ (\texttt{fun}\ \_ \Rightarrow \top)$.

7 Conclusion

The simple perspective offered by de Groote [2006] on dynamism and dynamic updating has been shown to be decomposible into two simpler and logically independent parts. His 2006 system, modeled here, involves an mapping of *second-order* terms in a source language into those in a target language. This amounts to a tree-to-term homomorphism, and allows de Groote to avoid defining λ-homomorphisms. This has been improved upon in later work [Lebedeva, 2012], where the source language is made higher order. In order to have a simple statement of the two parts that De Groote's system can be factored into (dynamism and context-sensitivity), I have departed from his original presentation in some ways. In particular, I have presented the source and target languages as one and the same, whereas de Groote treats the source language as properly syntactic, and the target language as properly semantic. Collapsing source and target languages has the aesthetically unpleasant consequence of introducing a 'meaningless' term, *pro*, into the language. While the same could have been done with the term **who**, I have chosen to leave it out of the language altogether, as it plays no illustrative semantic role. The second difference concerns the type of verbal elements in the source language, where, for de Groote, they take generalized quantifier denoting expressions as arguments: $[\![\texttt{beat}]\!] := \lambda X, Y.X(\lambda x.Y(\lambda y.\textit{beat}\ x\ y))$. This is made very simple in his presentation by treating `beat` as a constant in a language different from that of *beat*. This is not available to me, as both g_tr and d_tr should be defined independently of each other, and therefore must operate on the same source language, whence at least d_tr must map one language to itself. While this is a salient theoretical decision regarding the proper treatment of quantifier scope, it does not matter for the decomposition of de Groote's translation into the two simpler ones given here.

References

P. de Groote. Towards a montagovian account of dynamics. In M. Gibson and J. Howell, editors, *Proceedings of SALT 16*, pages 1–16, 2006.

J. Groenendijk and M. Stokhof. Dynamic predicate logic. *Linguistics and Philosophy*, 14: 39–100, 1991.

E. Lebedeva. *Expression de la dynamique du discours à l'aide de continuations*. PhD thesis, Université de Lorraine, 2012.