Do different syntactic trees yield different logical readings? Some remarks on head variables in typed lambda calculus.

No Author Given

No Institute Given

Abstract. A natural question in categorial grammar is the relation between a syntactic analysis and its logical form, i.e. the logical formula obtained from this syntactic analysis, once provided with semantic lambda terms. More precisely, do different syntactic analyses fed with equal semantic terms, lead to equal logical form? We shall show that when this question is too simply formulated, the answer is "no" while with some constraints on semantic lambda terms the answer is "yes".

1 Introduction: Lambek calculus and formal semantics

The Lambek calculus [6] is a logic developed for analyzing natural language. For Lambek grammars, the grammaticality of a sentence corresponds to the derivability of a statement in the logical calculus, given a lexicon mapping the words of the sentence to formulas. Lambek calculus proofs correspond to logical formulas in a simple and systematic way [2]. The questions which interests us in this paper is when different syntactic proofs correspond to different logical readings.

1.1 Proof theory

Introducing the Lambek calculus a bit more formally, given a sentence $w_1, \ldots, w_n$ of words, a function *Lex* mapping words to formulas, and a goal formula (we typically used s for *sentence*), we say that this sentence is grammatical if and only if there exists, for each i, an A_i in $Lex(w_i)$ and $A_1, \ldots A_n \vdash s$ is derivable.

Table 1 gives a simple Lambek calculus lexicon. Some words, like "John" are assigned atomic formulas (here np for 'noun phrase'). Similarly, "student" is assigned atomic formula n for 'noun'. The article "the" is assigned formula np/n indicating it is looking for a noun n (such as "student") to its right to form a noun phrase. Similarly, "slept" is assign formula $np\backslash s$, indicating is is looking for a noun phrase np (such as "the student" or "John") to form a sentence.

The natural deduction rules for the Lambek calculus are shown in Table 2. The elimination rules $/E$ and $\backslash E$ are simply directional versions of the modus ponens rule. The introduction rules $/I$ and $\backslash I$ require us the withdraw exactly one occurrence of the B formula (we use an index j unique to the proof to keep

$$
\begin{aligned}
lex(John) &= np & \qquad lex(ran) &= np\backslash s\\
lex(Mary) &= np & \qquad lex(slept) &= np\backslash s\\
lex(the) &= np/n & \qquad lex(ate) &= (np\backslash s)/np\\
lex(report) &= n & \qquad lex(wrote) &= (np\backslash s)/np\\
lex(student) &= n & \qquad lex(everyone) &= s/(np\backslash s)\\
lex(pizza) &= n & \qquad lex(someone) &= (s/np)\backslash s\\
lex(who) &= (n\backslash n)/(np\backslash s) & \qquad lex(every) &= (s/(np\backslash s))/n\\
lex(whom) &= (n\backslash n)/(s/np) & \qquad lex(some) &= ((s/np)\backslash s)/n
\end{aligned}
$$

Table 1. Lambek calculus lexicon

track of where each the B hypothesis of each introduction rule is withdrawn). The introduction rules have the additional condition that the withdrawn formula B must be the leftmost (resp. rightmost) free hypothesis in the subproof ending in A for the $\backslash I$ rule (resp. the $/I$ rule) *and* that there must be at least one other formula not already withdrawn (in other words, we exclude so-call empty antecedent proofs of the form $\vdash A/A$ and $\vdash A\backslash A$).

$$
\frac{A/B \quad B}{A}\ [/E] \qquad \frac{\begin{array}{c}\cdots\cdots [B]^j\\ \vdots\\ A\end{array}}{A/B}\ [/I]_j
$$

$$
\frac{B \quad B\backslash A}{A}\ [\backslash E] \qquad \frac{\begin{array}{c}[B]^j \cdots\cdots\\ \vdots\\ A\end{array}}{B\backslash A}\ [\backslash I]_j
$$

Table 2. Natural deduction rules for **L**.

Figure 1 shows an example proof for the sentence "every student wrote some report". Each non-discharged formula in the proof corresponds to a word in the lexicon, and we have written this word above the formula. Note that this is a η-long normal form proof: we have an elimination rule immediately followed by an introduction rule for the np hypothesis marked 1.

It is fairly easy, given a lexicon, to enumerate all long normal form proofs for a sentence [9], and with the lexicon of Table 1 the sentence "every student wrote some report" has exactly two such proofs, with the second shown in Figure 2.

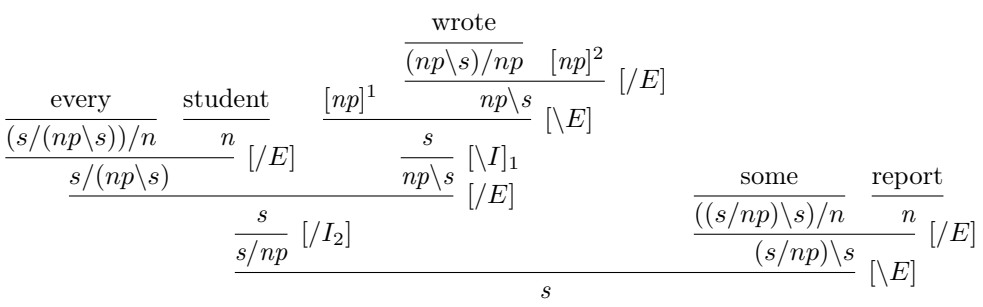

Fig. 1. Proof of "every student wrote some report"

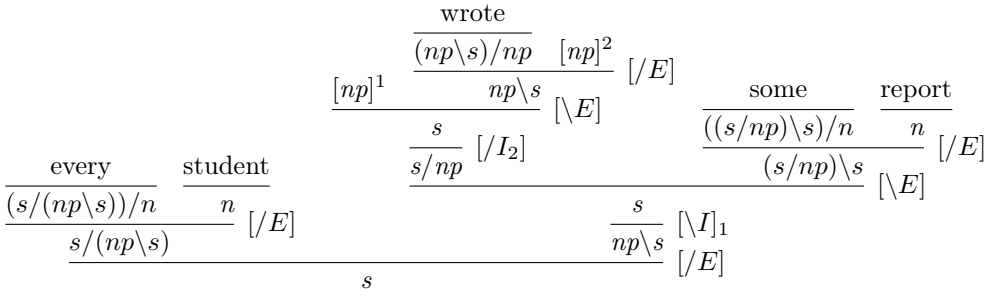

Fig. 2. Second proof of "every student wrote some report"

These two proofs correspond to the two readings of the sentence in a Montague-style treatment of quantification [7]: one where the existential quantifier "some" has wide scope over the universal quantifier "every", and one where "every" outscopes "some".

1.2 Semantic term assignment

There is a very direct way to turn the two proofs of the previous section into (lambda term representations) of the logical formulas representing the two possible meanings of the sentence. There is a division of labor here: the Lambek calculus proof specifies how the word in the lexicon are combined (in the form of a linear lambda term) and the lexical entry for each word specifies a (not necessarily linear) lambda term corresponding to the meaning of the word.

We first turn to the term assignment for the Lambek calculus proofs. The Lambek calculus is a non-commutative logic. For the semantic term assignment we are generally not interested in whether a left or right implication was used. In other words, semantic term assignment is done for proofs in the Lambek-van Benthem calculus **LP** (also known as multiplicative intuitionistic linear logic [3]).

In the Lambek-van Benthem calculus, there is only a single implication, the linear implication '$\multimap$'. The trivial a forgetful mapping from Lambek calculus connectives to those of **LP** is the following:

$$\begin{aligned} p^* &= p \\ (A/B)^* &= B^* \multimap A^* \\ (B\backslash A)^* &= B^* \multimap A^* \end{aligned}$$

Definition 1. *A lambda term M is linear [5] whenever:*

- *each free variable occurs exactly once, and*
- *for each subterm $\lambda x.N$ of M, x has exactly one free occurrence in N*

Proofs in the Lambek-van Benthem calculus correspond to linear lambda terms. Table 3 shows the natural deduction rules for **LP** together with term assignment for the proof. The elimination rule has the condition that M and N do not share variables. The introduction rule has the condition that exactly one occurrence of the formula B (with variable x) is withdrawn. The mapping $.^*$ has the property that it not only translates **L** formulas to **LP** formulas, but also **L** derivation rule (and therefore derivations) to **LP** derivation rules (and derivations).

Although we use the Lambek calculus as an example in this paper, most type-logical grammars (including the multi-modal non associative Lambek calculus) have a similar forgetful mapping from their logical connectives (and the corresponding derivation rules) to **LP** [8].

The proofs in Figures 1 and 2 correspond to the lambda terms given in 1 and 2 respectively.

$$\frac{N : B \quad M : B \multimap A}{(M\,N) : A}\ [\multimap E] \qquad \frac{\begin{array}{c}[x : B]^j \\ \vdots \\ M : A\end{array}}{\lambda x.M : B \multimap A}\ [\multimap I]_j$$

Table 3. Natural deduction rules for **LP**/multiplicative intuitionistic linear logic with term labeling.

$$(w_4\,w_5)(\lambda y((w_1\,w_2)\,(\lambda x((w_3\,y)\,x)))) \tag{1}$$
$$(w_1\,w_2)(\lambda x((w_4\,w_5)\,(\lambda y((w_3\,y)\,x)))) \tag{2}$$

Finally, to obtain a representation of the meaning of the sentence we substitute the lexical meaning for each word. Following Montague, we leave some words unanalyzed, using the constant *student* as the meaning of the word "student", and similarly for "wrote" and "report", which are assigned the meaning *write* and *report* respectively. The interesting words in this example are "every" and "some". Using the constants $\forall$ and $\exists$, both of type $(e \to t) \to t$, to represent the universal and the existential quantifier, and the constants $\wedge$, $\vee$ and $\Rightarrow$ of type $t \to (t \to t)$ to represent the binary logical connectives, we can assign the following lambda term to "every" and to "some":

$$\lambda P \lambda Q \forall (\lambda x.(\Rightarrow (P\,x))(Q\,x)) \tag{3}$$

$$\lambda P \lambda Q \exists (\lambda x.(\wedge (P\,x))(Q\,x)) \tag{4}$$

Substituting the lexical terms for each of the corresponding variables derived terms 1 and 2 produces the following two terms.

$$(\lambda P \lambda Q \exists (\lambda z.(\wedge (P\,z))(Q\,z))\ \mathit{report})(\lambda y((\lambda R \lambda S \forall (\lambda v.(\Rightarrow (R\,v))(S\,v))\ \mathit{student})\ (\lambda x((\mathit{write}\,y)\,x)))) \tag{5}$$
$$(\lambda R \lambda S \forall (\lambda v.(\Rightarrow (R\,v))(S\,v))\ \mathit{student})(\lambda x((\lambda P \lambda Q \exists (\lambda z.(\wedge (P\,z))(Q\,z))\ \mathit{report})\ (\lambda y((\mathit{write}\,y)\,x)))) \tag{6}$$

These terms normalize to:

$$\exists (\lambda z.(\wedge (\mathit{report}\,z))(\forall (\lambda v.(\Rightarrow (\mathit{student}\,v))((\mathit{write}\,z)\,v)) \tag{7}$$
$$\forall (\lambda v.(\Rightarrow (\mathit{student}\,v))(\exists (\lambda z.(\wedge (\mathit{report}\,z))((\mathit{write}\,z)\,v)) \tag{8}$$

In more standard logical notation, these terms represent the following two formulas (it is always the case that closed β-normal η-long terms of type t with

constants of some logical system can unambiguously be interpreted as logical formulas, see e.g. [9, Ch. 3]).

$$\exists z.report(z) \wedge \forall v.[student(v) \Rightarrow write(v, z)] \tag{9}$$

$$\forall v.student(v) \Rightarrow \exists z.[report(z) \wedge write(v, z)] \tag{10}$$

So we have two Lambek calculus proofs producing two different readings. Now, while it is the case that two different natural deduction proofs for the Lambek calculus always produce two different linear lambda terms (terms like 1 and 2), the question which will interest us in the rest of this paper is the following: when can we guarantee that different Lambek calculus proofs (or proofs in another system of type-logical grammar) produce different meanings? By this, we mean different meaning in the sense of different lambda terms after lexical substitution and normalization, and not terms representing logical formulas which are not logically equivalent. This distinction is obvious when we replace our example sentence by "some student wrote some report". Here we still produce two different terms, where the two existential quantifiers have different scope with respect to each other. However, these two terms correspond to logically equivalent formulas.

Example 1. Even when we require different terms instead of terms representing logical formulas which are not equivalent, the property doesn't hold in general. For example, given a binary concatenation operator "+" (which, for convenience, we write as an infix operator[1]), the following substitution produces

$$every + student + wrote + some + report$$

for both 1 and 2.

$$\begin{aligned} w_1 &:= \lambda P \lambda Q.Q(every + P) \\ w_2 &:= student \\ w_3 &:= \lambda y \lambda x.x + wrote + y \\ w_4 &:= \lambda P \lambda Q.Q(some + P) \\ w_5 &= report \end{aligned}$$

This example uses lambda terms to produce string, as is done in lambda-grammars/abstract categorial grammars [11,4,10], and shows different meanings can correspond to the same string.

2 When do different proofs produce different meanings?

A natural question when computing the logical formula associated from a Lambek analysis and the semantic lambda term associated with each word is the following:

[1] We can define "+" as $\lambda y \lambda x \lambda z x(y\, z)$ (i.e. as function composition). Then, w_3 gets assigned the term $\lambda y \lambda x \lambda z(x\,(wrote\,(y\, z)))$, and similarly for the other terms.

Question 1. Assume that the sentence $w_1 \cdots w_n$ has two syntactic analyses P_1 and P_2, when replacing each w_i (a free variable representing m_i in the syntactic analysis that is a linear lambda term) by the associated semantic lambda term t_i (non linear, with constants) in P_1 and in P_2 does beta reduction give different lambda terms (i.e. logical formulas), i.e. does one have

$$P_1[w_1 := t_1] \cdots [w_n := t_n] \overset{\beta}{\neq} P_2[w_1 := t_1] \cdots [w_n := t_n] \quad ?$$

We shall see that in its full generality, this question must be answered negatively, but we shall consider restrictions on the semantic lambda terms and consider strongly different syntactic analyses.

Definition 2. *A syntactic λ-term is a β-normal, simply-typed linear λ-term with one occurrence of each free variable in $w_1, \ldots, w_n$ with $n > 0$ — those free variables are the words of some analyzed sentence.*

2.1 Semantic lambda terms and lambda I calculus

The answer to our question is negative:

Proposition 1. *There exist P_1, P_2 two syntactic λ-terms both of type σ and with the same free variables $w_1, w_2 \ldots w_n$, and and there exist $t_1, t_2 \ldots, t_n$ n semantic λ-terms such*

$$P_1 \overset{\beta}{\neq} P_2 \quad AND \quad P_1[w_1 := t_1] \cdots [w_n := t_n] \overset{\beta}{=} P_1[w_1 := t_1] \cdots [w_n := t_n]$$

Proof. Take

$$P_1 \equiv w_1((w_2 w_3) w_4)$$
$$P_2 \equiv w_1((w_2 w_4) w_3)$$

where $w_1 : t \to t$, $w_2 : e \to (e \to t)$ and w_3, w_4 are both of type e. Moreover take

$$t_1 \equiv \lambda y.k_1 \quad t_2 \equiv \lambda x_1 \lambda x_2((k_2 x_1) x_2)) \quad t_3 \equiv k_3 \quad t_4 \equiv k_4$$

where $k_1 : t$, $y : t$, $k_2 : e \to (e \to t)$ and t_3, t_4, x_1, x_2 are of type e. Make the following substitution.

$$P_1[w_1 := t_1][w_2 := t_2][w_3 := t_3][w_4 := t_4] \qquad P_2[w_1 := t_1][w_2 := t_2][w_3 := t_3][w_4 := t_4]$$

Both terms reduces to k_1

This proposition (counter example to our question) holds because β-reduction can delete i.e., when we have $t := \lambda x t'$ in which $x \notin Fv(t')$ tM reduces in one step to t' for all t, M. Therefore is it essential to exclude such cases if one hopes to give a positive answer to the claim. However people who experimented categorial lexicons have probably noticed that semantic lambda terms are lambda I terms i.e. β deduction never erase any subterm (it would be strange that a semantic

function of several arguments does not take one of its argument into account). A λ-term t is called a λ_I term [5,1] iff for each sub-term of the form $\lambda x.M$ in t, x occurs free in M at least once. The class of λ_I-terms are closed under $\beta\eta$ reduction i.e., every term obtained by reducing a member of the class is also a member of the class.

One may wonder whether semantic lambda terms could be asked to be linear. This would be damaging since the quantifiers are not linear lambda terms:

$$every : \lambda P \lambda Q \forall (\lambda x.(\Rightarrow (P\, x))(Q\, x))$$

2.2 Simple semantic lambda terms

Given that we want the difference between the syntactic analyses to be preserved during β reduction, we focus on having constants as head variables in the semantic lambda terms that are substituted with the free variables. Indeed, otherwise when the head variable is bound, the reduction of the corresponding redex creates a redex and turns an argument into a function: this is a substantial modification of the lambda term which may identify different lambda terms as opposed to the claim we would like to prove.

Definition 3. *A* simple semantic lambda term *is a β-normal η-long λ_I-term with constants whose head variable is a constant i.e.,*

$$t := \lambda z_1, \cdots z_n k T_1 T_2 \cdots T_m$$

where k is a constant [2]

The lambda terms given in Example 1 are not simple according to our definition: the terms for "every" and "some" have a bound variable as head term; the same is true for "wrote" when we write out the definition of "+" as indicated in Footnote 1.

Furthermore, this requirement, to have a constant as the head variable suggest a way to avoid some reductions to yield the same term, by a symmetric treatment of the symmetric head constant.

Proposition 2. *There exist P_1, P_2 two syntactic λ-terms both of type σ and with the same free variables $w_1, w_2, \ldots w_n$, and and there exist $t_1, t_2 \ldots, t_n$ n simple semantic λ-terms such that*

$$P_1 \overset{\beta}{\neq} P_2 \quad AND \quad P_1[w_1 := t_1] \cdots [w_n := t_n] \overset{\beta}{=} P_1[w_1 := t_1] \cdots [w_n := t_n]$$

Proof. take

$$P_1 \equiv ((w_1 w_2) w_3)$$

$$P_2 \equiv ((w_1 w_3) w_2)$$

[2] Because this is a λ_I term Iz_i as a free occurrence in one of the T_i.

with $w_1 : e \to (e \to t)$ and w_2, w_3 of type e

$$t_1 \equiv \lambda x_1 \lambda x_2((k_1 x_1)x_2) \quad t_2 \equiv k_2 \quad t_3 \equiv k_2$$

with $k_1 : e \to (e \to t)$, x_2, x_2, k_2 of type e and make the following substitution

$$P_1[w_1 := t_1][w_2 := t_2][w_3 := t_3] \quad P_2[w_1 := t_1][w_2 := t_2][w_3 := t_3]$$

After β-reduction the two terms become *beta*-equal.

This counterexample shows that we should also require, at least, that the n simple semantic lambda terms have different head-constant

2.3 Restrictions on the semantic lambda terms are not enough

Unfortunately — even with this restriction — if one formalizes the notion of difference between the two syntactic analyses of the sentence $w_1 \cdots w_n$ in terms of β-difference between syntactic terms one is doomed to failure.

Proposition 3. *There exist P_1, P_2 two syntactic terms, both of type σ, with the same free variables $w_1, \ldots, w_n$ and $t_1, t_2 \ldots, t_n$ n simple semantic lambda terms such that $\forall i \forall j$ $1 \leq i \leq j \leq n$ if $i \neq j$ then the head-constant of t_i is different from the head-constant of t_j.*

$$P_1 \overset{\beta}{\neq} P_2 \quad AND \quad P_1[w_1 := t_1] \cdots [w_n := t_n] \overset{\beta}{=} P_1[w_1 := t_1] \cdots [w_n := t_n]$$

Proof. take

$$P_1 \equiv w_1(\lambda x \lambda y((w_2 x)y))$$
$$P_2 \equiv w_1(\lambda y \lambda x((w_2 x)y))$$

where $x : e, y : e, w_2 : e \to (e \to t), w_1 : (e \to (e \to t)) \to t$. Take

$$t_1 \equiv \lambda P(k_1((Px)x)) \quad t_2 \equiv (\lambda z \lambda y((k_2 z)y))$$

where $P : (e \to (e \to t)) \to t$, $k_1 : t \to t$, $k_2 : e \to (e \to t)$ and x, z, y are of type e. And make the following substitution

$$P_1[w_1 := t_1][w_2 := t_2] \quad P_2[w_1 := t_1][w_2 := t_2]$$

After β-reduction the two terms become *beta*-equal.

3 Strong differences in syntactic analyses yield different readings

In the last section we have seen that the question has a negative answer if the difference between the two syntactic analyses is just syntactic difference (up to the renaming of bound variables). By consequence we can attack the problem by using at least two different strategies.

Strategy 1 Refining our notion of syntactic lambda-terms. In fact we know that not all linear lambda terms have a corresponding Lambek calculus derivation, because Lambek calculus is not commutative and therefore is proofs correspond to a proper subset of the linear lambda terms. This may be a good strategy, because syntactic analyses enjoy this restriction, even though it may be hard to state precisely and succinctly (though we could follow the ideas of [12] for a directional lambda calculus)

Strategy 2 Define a stronger notion of difference between syntactic analyses and the resulting lambda terms.

The first strategy seems promising: we know that the two syntactic terms $P_1 \equiv w_1(\lambda y \lambda x((w_2 x)y))$ $P_2 \equiv w_1(\lambda y \lambda x((w_2 x)y))$ of Proposition 2 could not correspond to two parses obtained by the same category assignment to the free variables w_1 and w_2. However it is really difficult to exactly characterize the sub-class of typed linear lambda terms that corresponds to derivations in the Lambek-calculus. The translation from the latter to the former is not injective and thus some information that could be relevant is lost.

The second strategy could be pursued only if the new notion of difference captures some interesting case of differences between syntactic analysis of the same sentence e.g. scope ambiguity for quantifiers. We are going to pursue this path by defining a notion of *dominance* between occurrences of unbound terms in a normal λ-term.

3.1 Some remarks on head variables and constants

Before proceeding with dominance, let us state some useful and easy propositions.

Proposition 4. *Let t be a simply typed λ_i-term in head-normal form having the following shape.*

$$t := \lambda z_1, \cdots z_n k T_1 T_2 \cdots T_m$$

where k is a constant. Call M the normal form of t. M has k as head-variable.

Corollary 1. *Let t_1, t_2 be simply typed λ_i-terms in head-normal form having the following shape.*

$$t_1 := \lambda z_1, \cdots z_n k_1 T_1 T_2 \cdots T_m \quad t_2 := \lambda z'_1, \cdots z'_n k_2 T'_1 T'_2 \cdots T'_m$$

where k_1, k_2 are constants. If k_1, the constant of t_1, is different from k_2, the constant of t_2, then $t_1 \neq_\beta t_2$

Proposition 5. *Let t be a simply typed λ_I term in head-normal form in which the head variable is free. Let R be a simple semantic term in which k is the head-constant. Then the normal form M of $t[x := R]$ has k has head-variable.*

3.2 A positive answer when syntactic analyses define different dominance relations

Definition 4 (Dominance). *In a term M, occurrences of constants and free variables are endowed with a dominance relation as follows.*

If the term M is a sequence of abstractions $\lambda\vec{x}.\ t$ (t is not itself an abstraction) then the dominance relations are the ones in t.

If the term M is a sequence of application $T_0T_1\cdots T_n$ (T_0 is not itself an application) the dominance relations are the union of the ones in each of the T_i, plus in some cases, additional relations: if T_0 has a a head variable which is a constant then this constant dominates all constants in all the T_i's.

When x dominates y we will write this as $x \lhd y$.

It should be observed that when a lambda term is normal, the dominance relation is the transitive closure of "the head constant h dominates the head constants in the terms it is applied to".

A first obvious remark:

Proposition 6. *Let P be a syntactic lambda term with words $w_1, \ldots, w_n$. Let t_i be the corresponding simple semantic lambda terms with head constant c_i. If $w_{i_0} \lhd w_{i_1}$ in P then $c_{i_0} \lhd c_{i_1}$ in $P[\vec{w} := \vec{t}]$.*

Proposition 7. *Let U be a typed lambda I term (without erasing/weakening) including two occurrences of constants c and c' such that $c \lhd c'$ in t. Assume $U \xrightarrow{\beta} U'$. Then each trace c_i of c is associated with a set of occurrences $c_i'^j$ of c' in U' with $c_i \lhd c_i'^j$ in U' — the sets $C_i' = \{c_i'^j\}$ define a partition of the traces of c'. In particular there never is a relation the other way round after reduction: $c_i' \not\lhd c_i$ in U' for all i.*

Proof. Checking this for the reduction of one redex is enough. A redex in U looks like $U = V[(\lambda x.A)B]$ For each relation $c \lhd c'$ (by induction on the number of β reductions performed so far there might already be several such pairs), because of the definition of dominance c and c' are

1. outside of the redex, i.e. elsewhere in V, c and c' do not move
2. both in A, c and c' do not move
3. both in B, c and c' moves in A being possibly duplicated
4. $A = cA_1\cdots A_k$ and c' is in B: c' might be duplicated but c still dominates each of the duplicates of c' after reduction.
   (a) c' is in some A_{k_0} c and c' do not move
   (b) c' is in B c' moves in A being possibly duplicated, but $c \lhd c'$ remains for every pair of duplicates.

In any case $c \lhd c'$ also holds after reduction to $A[x := B]$. In case the "initial" c and c' are both in B and A contains several occurrences of x, they have several traces which are in a one to one correspondence with $c \lhd c'$ in t'. In case c is the head variable of A the reduction duplicates c' and c dominates every duplicate of c'.

Note that having λ_I terms is crucial otherwise both c and c' or just c' may disappear during the β reduction.

It should be observed that dominance relations, even between occurrences of constants, can be introduced by β-reduction (when a constant becomes the head variable of some term) but this does not does not prevent the proposition to hold, since we only require the dominance relation to not inverse an already existing relation between two occurrences of constants.

Corollary 2. *Assume two syntactic analyses P_1 and P_2 give opposite dominance relation between two words, $u \lhd u'$ in P_1 and $u \lhd u'$ in P_2. Whatever the semantic lambda terms for u and u' with different head constant c and c' are, the associated logical forms will be different.*

Proof. We have $c \lhd c'$ in $p_1[\vec{u} := \vec{t}]$ so the traces of c dominate the traces of c' in its normal form $p_1[\vec{u} := \vec{t}]^*$ because of the proposition 7.

We have $c' \lhd c$ in $p_2[\vec{u} := \vec{t}]$ so the traces of c' dominate the traces of c in its normal form $p_2[\vec{u} := \vec{t}]^*$ because of the proposition 7.

So these normal forms cannot be equal.

As an example, recall that there are two syntactic analyses of *every student passed some exam*. In the $\forall\exists$ reading one has $\forall \lhd \exists$ while in the $\forall\exists$ reading one has $\forall \lhd \exists$. Consequently we know in advance the logical forms are not going to be the same – they are obtained by inserting the semantic lambda terms and applying β-reduction.

The previous corollary does not mean that the logical forms cannot be logically equivalent. For instance, in a sentence like *every student passed all exams* there are two syntactic analyses one with *all* $\lhd$ *every* and one in which *every* $\lhd$ *all*, this will be true as well in the logical form: one $\forall$ dominates the other in the normal form, and, depending on the syntactic analysis it is not the same one. However, the logical formulas are equivalent, just like the formulas $\forall x P(x) \Rightarrow \forall y Q(y) \Rightarrow R(x, y)$ and $\forall y Q(y) \Rightarrow \forall x P(x) \Rightarrow R(x, y)$ are not equal but equivalent.

4 Conclusion

We have shown that some quite reasonable formalizations of the claim "different syntactic analyses yield different readings" are false. We nevertheless established that with stronger hypotheses on semantic terms, and using dominance relation between constants, the claim is true.

In the future we would like to both weaken the condition on semantic lambda terms in order to incorporate reflexives (they look like $\lambda P \lambda x.\ Pxx$, they have no head constant) and in balance to refine the notion of syntactic terms.

Indeed, syntactic lambda terms should be characterized as the class of linear lambda terms that are issued from the Lambek calculus, i.e. linear lambda terms that are typable in a non-commutative calculus. This path seems promising. As we have mentioned at the beginning of Section 3, the two terms of Proposition 3 can not correspond correspond to two parses obtained by the same category assignment to the free variables w_1 and w_2. This aspect seems to be tied to the non-commutativity of the Lambek calculus.

References

1. Barendregt, H.P.: The lambda calculus: its syntax and semantics, Studies in Logic and the Foundations of Mathematics, vol. 103
2. van Benthem, J.: Language in Action: Categories, Lambdas and Dynamic Logic. MIT Press, Cambridge, Massachusetts (1995)
3. Girard, J.Y.: Linear logic. Theoretical Computer Science 50, 1–102 (1987)
4. de Groote, P.: Towards abstract categorial grammars. In: Proceedings of the 39th Annual Meeting on Association for Computational Linguistics. pp. 252–259. Association for Computational Linguistics (2001)
5. Hindley, J.R.: Basic Simple Type Theory, Cambridge Tracts in Theoretical Computer Science, vol. 42. Cambridge University Press (1997), corrected edition, 2008
6. Lambek, J.: The mathematics of sentence structure. American Mathematical Monthly 65, 154–170 (1958)
7. Montague, R.: The proper treatment of quantification in ordinary English. In: Thomason, R. (ed.) Formal Philosophy. Selected Papers of Richard Montague. Yale University Press, New Haven (1974)
8. Moortgat, M.: Categorial type logics. In: van Benthem, J., ter Meulen, A. (eds.) Handbook of Logic and Language, chap. 2, pp. 93–177. Elsevier/MIT Press (1997)
9. Moot, R., Retoré, C.: The Logic of Categorial Grammars: A Deductive Account of Natural Language Syntax and Semantics. No. 6850 in Lecture Notes in Artificial Intelligence, Springer (2012)
10. Muskens, R.: Languages, lambdas and logic. In: Kruijff, G.J., Oehrle, R.T. (eds.) Resource Sensitivity in Binding and Anaphora, pp. 23–54. Studies in Linguistics and Philosophy, Kluwer (2003)
11. Oehrle, R.T.: Term-labeled categorial type systems. Linguistics & Philosophy 17(6), 633–678 (1994)
12. Wansing, H.: Formulas-as-types for a hierarchy of sublogics of intuitionistic propositional logic. In: Pearce, D., Wansing, H. (eds.) Nonclassical Logics and Information Processing. pp. 125–145. Springer Berlin Heidelberg, Berlin, Heidelberg (1992)