Anaphoric Linking at Run Time: A Type-Logical Account of Discourse Representation*

Kazushige Terui†
terui@abelard.flet.keio.ac.jp
Keio University

Abstract

A number of attempts have been made to combine Discourse Representation Theory with type-logical grammar in order to obtain a compositional (bottom-up) framework for discourse representation (e.g., Muskens[Mus94], van Eijck & Kamp[vEK97]). In those attempts, however, it is usually assumed that anaphoric links are given in advance of the construction of discourse representation structures (DRSs). This assumption causes a difficulty, typically dealing with plural anaphora, because generally suitable referents of plural pronouns are not present before the construction, but should be created *through* the construction of DRSs.

To settle this difficulty, we shall propose a new type-logical framework of DRT in which anaphoric linking takes place during the construction of DRSs (*Run Time Anaphoric Linking*). The construction is bottom-up in the sense that it is based on the sequent calculus proof search of Lambek calculus. We exploit quantifier types to represent the run time anaphoric linking mechanism. We shall briefly illustrate how plural anaphora are treated in our framework. In particular, we shall show that *Summation* and *Abstraction*, which are used to *create* suitable antecedent discourse referents for plural pronouns in [KR93], are available in our type-logical setting using the linear logic modality.

1 Introduction

Discourse Representation Theory (DRT, [Kam81], [Hei82], [KR93]) offers an attractive account of anaphoric phenomena in discourse, and it explains some puzzles involving indefinites and anaphoric pronouns such as donkey sentences very well. The standard DRT, however, relies on a *procedural* account, that is, interpretation of discourse is explained by means of a set of interpretation rules (*DRS construction rules*) that provides a *top-down* procedure to construct a *Discourse Representation Structure* (DRS) from a given sequence of sentences. Although the procedural explanation is intuitively understandable and practically useful, it has been often pointed out from a theoretical point of view that it does not give a mathematically clean semantics to natural languages, due to the lack of *compositionality*. Hence, a number of attempts have been made to formulate a suitable compositional (*bottom-up*) DRT (e.g. [Zee89], [Ash93], [Mus94], [KKP95], [Mus96], [vEK97]). The ways in which they achieve compositionality are more or less related to the theory of *Type-logical Categorial Grammar* (see [Moo88], [Mor94], [Moo97], [Car98]) in that they assign each word a meaning of higher-order type and the composition of those meanings are based on function-application of typed lambda calculus which can be well described in terms of some type-logics. Among those, [Mus94] explicitly refers to the theory of Lambek categorial grammar and investigates the combination of these two theories.

In those attempts, however, it is usually assumed that *anaphoric links are given in advance*. For example, the problem addressed in [vEK97] is the following:

*This work was partially supported by the Spinoza project 'Logic in Action' at ILLC, the University of Amsterdam. The author would like to thank Takako Iseda, Makoto Kanazawa, Norihiro Ogata, Maarten de Rijke, and Christopher Tancredi for their encouragement and valuable comments.

†Research Fellow of the Japan Society for the Promotion of Science

> Assuming that an anaphoric indexing for a sentence is given, ..., give an algorithm for updating an available representation structure with the information from that sentence. ([vEK97], p.216)

In other words, the inputs to their compositional DRTs are supposed to be coindexed texts such as

(1) The [man who smiles]^1 does not hate Bill. He$_1$ respects Bill.

Although this approach seems to work for certain restricted fragment of natural languages, it is by no means obvious that we can always give an input text a coindexation indicating the intended anaphoric links. Indeed, it would get in trouble when it came to handling plural anaphora. For example, consider the following texts;

(2) Susan has found most books which Bill needs. They are on his desk.

In (2), the pronoun 'They' in the second sentence is naturally interpreted as referring to the set of books which Susan has found and Bill needs. But there seems to be no *principled* way to give the anaphoric indices to the text in advance of interpretation, because apparently there is no overt expression which represents by itself the set of such books. Rather, the suitable referent of the pronoun should be created *through* interpretation of the first sentence. We may, therefore, reasonably claim that *anaphoric links should be established at run time of interpretation*, and that a proper theory of discourse representation should offer a suitable mechanism of *Run Time Anaphoric Linking*.

The main purpose of this paper is to propose an appropriate type-logical framework of DRT which offers such a run time anaphoric linking mechanism. We do not assume that referential expressions in input sentences are given anaphoric indices. In particular, a pronoun is represented by a single lexical entry, rather than by an infinite number of lexical entries each of which corresponds to one anaphoric index, as found in e.g. [Mus94] and [vEK97]. The mechanism consists of two components. First, some expressions like indefinites introduce a new *reference marker* (RM, hereafter) into the context(*RM-introduction*). Second, pronouns catch a RM (possibly) from the context (*RM-catching*). The core of our proposal lies in the type-logical representation of these two mechanisms by means of quantifier types.

Our framework is based on *Lambek sequent calculus* ([Lam58], [vB91]) and we make use of labels and quantifier types. Since our use of quantifiers looks somewhat unusual, let us now explain the ideas behind it. It comes from the following two sources.

1. Proof Search as Computation

It is widely known that bottom-up proof search in sequent calculi for certain logics offers computational interpretations to those logics (see [MNPS91]). This paradigm has been successfully applied to give a logical basis to certain logic programming languages, constraint programming languages, concurrent process calculi, etc. For instance, proof search in linear logic naturally expresses a concurrent computation involving multiple *agents* (or *processes*) (e.g. [AP91], [HM94], [KY95]). According to this view of proof search as concurrent computation, we have the following correspondence;

formula	=	agent (or process)
inference rule	=	transition rule
bottom-up proof search	=	computation.

For example, the following instance of $\multimap$-left rule of linear logic

$$\frac{\overline{\alpha \vdash \alpha} \quad B, \Gamma \vdash}{\alpha, \alpha \multimap B, \Gamma \vdash}$$

(where α denotes an atomic formula, Γ denotes a list of formulas, and $\multimap$ denotes linear implication) may be read in a bottom-up manner as: "agent $\alpha \multimap B$ receives an information token α from the

context and then becomes B." In this way, the inference rules determine the behavior of agents (we say that the inference rules give agents their *operational meanings*).

In this paper, we shall apply the paradigm to Lambek calculus. Since a formula of Lambek calculus represents a syntactic category, we have the following correspondence;

syntactic category = formula = agent (or process).

2. Labelled Deductive Systems

There is another influential view of logical-computational systems that the basic unit of logical deduction (or computation) is a formula with a label, which is written of the form $a:A$ and called a *declarative unit* ([Gab96]). Here formula A represents the logical component of the declarative unit, while label a expresses some additional information. In a type-logical setting, labels typically express semantic contents (by λ-terms) while formulas specify the types of their labels. In connection with the proof-search-as-computation paradigm, we may consider the following correspondence;

declarative unit	=	agent (or process)
label	=	semantic content of an agent
formula	=	behavioral feature of an agent.

In view of these two paradigms, our use of quantifiers may be explained as follows. As mentioned above, it is the inference rules that give agents their operational meanings. In the standard sequent calculi, first-order quantifiers have the following inference rules (here we only consider the left-rules);

$$\frac{A[t/x], \Gamma \vdash}{\forall x A, \Gamma \vdash}\ \forall L \qquad\qquad \frac{A[u/x], \Gamma \vdash}{\exists x A, \Gamma \vdash}\ \exists L$$

(t is an arbitrary term) (u does not occur in the lower sequent)

If we read these inference rules in a bottom-up manner, we can give quantified agents the following operational meanings (cf. [KY95]);

$\forall x A$: Choose an appropriate term t, possibly from the context, then execute $A[t/x]$;
$\exists x A$: Introduce a *new* free variable u, then execute $A[u/x]$.

Our main idea is to use the existential quantifier to represent RM-introduction and the universal quantifier to represent RM-catching so that they together constitute the run time anaphoric linking mechanism. To achieve this, we allow a quantifier in a formula to bind a variable in a label, because RMs, which we would like to bind, occur in labels.

In Section 2, we explain our basic ideas a bit more by concrete examples. Then, the ideas are formalized as logical calculus **Lq** in Section 3 and Section 4. It is very crucial what exactly are the 'correct' DRSs, because in our labelled sequent calculus the labels are taken to be 'correct' (λ-)DRSs and the constraints on the labels operation prevent illegal sentences and illegal readings from being interpreted as DRSs. In Section 3, we shall give a precise definition of DRSs which are 'correct' in our sense (called *strict* DRSs), and their higher order versions (called *legal λpDRSs*, or *labels*). In Section 4, we define labelled sequent calculus **Lq**, which offers us a deductive basis for discourse interpretation. **Lq** satisfies the cut elimination property (Theorem 1).

In Section 5, we give **Lq** a computational interpretation, so that we can describe explicitly how to construct a DRS from a given text. The interpretation is based on an *operational semantics* which embodies a clear conception how the process of execution goes. Our main theorem is the completeness of the operational semantics with respect to **Lq** (Theorem 2). It establishes a tight correspondence between logical deduction (**Lq** sequent calculus) and computation (DRS construction via operational rules).

In Section 6, several examples are considered in order to examine our calculus from a more practical viewpoint. **Lq** uses Moortgat's scoping operator $\Uparrow$ ([Moo88], [Moo91]) to represent scoping expressions. The first example shows that the operator offers interesting mechanisms of *box-entering* and *box-leaving* in the light of DRT. The second example shows that our calculus, a

combination of DRT and type-logical grammar, offers a better account on the scope of indefinites than purely type-logical frameworks, because DRT provides a strict distinction between indefinites and other quantifiers while the pure type-logical grammar not. The third example shows that our calculus naturally embodies certain sentential grammar principles. In particular, some special cases of i-within-i effects are explained in our framework very naturally.

In Section 7, we briefly illustrate the treatment of plural anaphora in our framework. The explanation in [KR93] is essentially based on two mechanisms: *Abstraction* and *Summation*. There is no difficulty to incorporate these mechanisms into their theory because the explanation given is completely procedural; it simply suffices to introduce new DRS construction rules which create new RMs from DRSs which have already constructed. However, it is by no means a trivial matter whether these mechanisms are available in type-logical settings as well. We shall show an extension of our calculus which incorporates such mechanisms type-logically. Here the modality of *linear logic* ([Gir87], [Gir95]) plays a central role. Using the modality, the mechanisms are represented as *daemon processes*, which are reusable as many times as we like.

Section 8 concludes the paper with some discussions. The proofs of Theorem 1 (the cut elimination theorem for **Lq**) and Theorem 2 (the operational completeness theorem) are given in Appendix.

2 Basic Ideas

Traditional applicative categorial grammar has the following transition rules (*operational rules*);

(Receive-Left) $(\Gamma_1, a{:}A, b{:}A\backslash B, \Gamma_2) \longrightarrow (\Gamma_1, b \bullet a{:}B, \Gamma_2)$;

(Receive-Right) $(\Gamma_1, b{:}B/A, a{:}A, \Gamma_2) \longrightarrow (\Gamma_1, b \bullet a{:}B, \Gamma_2)$.

where Γ_1 and Γ_2 stand for lists of declarative units of the form $a : A$. For the moment, let us assume that labels are λ terms and $b \bullet a$ denote the normal form of ba in the above. These rules characterize the operational behavior of slash types. Note that these rules corresponds to the left inference rules of slashes in Lambek calculus, if we read these rules in a bottom-up manner;

$$\frac{(A \vdash A) \quad \Gamma_1, B, \Gamma_2 \vdash C}{\Gamma_1, A, A\backslash B, \Gamma_2 \vdash C} \qquad \frac{(A \vdash A) \quad \Gamma_1, B, \Gamma_2 \vdash C}{\Gamma_1, B/A, A, \Gamma_2 \vdash C}\ .$$

In order to motivate our use of quantifiers, let us see how the existing frameworks of compositional DRT treat indefinites and pronouns. For example, [vEK97] introduces an infinite number of lexical entries for indefinites and pronouns each of which corresponds to an anaphoric index; Below is a slight simplification of the lexical entries for 'a' and 'he';

$$\mathrm{a}_i \Rightarrow \lambda PQ.(\mathbf{u_i}; P(\mathbf{u_i}); Q(\mathbf{u_i})) : ((s/(np\backslash s))/n)$$
$$\mathrm{he}_i \Rightarrow \lambda P.P(\mathbf{u_i}) : s/(np\backslash s)$$

where the subscript i indicates an anaphoric index[1]. Since indices are given as part of the lexical entries, 'a' and 'he' are infinitely ambiguous.

Our main idea is to *bind* each reference marker (RM) occurring in a lexical entry by a quantifier, rather than to give it an index in advance. So we write the lexical entries for 'he' and 'a' become as follows;

$$\mathrm{a} \Rightarrow \lambda PQ.(\mathbf{x}; P(\mathbf{x}); Q(\mathbf{x})) : \exists \mathbf{x}((s/(np\backslash s))/n)$$
$$\mathrm{he} \Rightarrow \mathbf{x} : \forall \mathbf{x} np.$$

Here RM $\mathbf{x}$ in the entry for 'a' is bound by an existential quantifier, while $\mathbf{x}$ in the entry for 'he' is bound by a universal quantifier. This reflects the operational difference between these items: *an indefinite introduces a new RM, while a pronoun catch a RM from the context.* The next two operational rules give these quantifiers their operational meanings.

[1] To be precise, [vEK97] gives the lexicon using the merging operator $\bullet$ instead of the simple sequencing operator ;. But this point is irrelevant to the present argument.

(Introduce-RM) $(\Gamma_1, a{:}\exists \mathbf{x} A, \Gamma_2) \longrightarrow (\Gamma_1, a[\mathbf{u}/\mathbf{x}]{:}A, \Gamma_2)$,
where RM $\mathbf{u}$ does not occur freely in the antecedent configuration;

(Catch-RM) $(\Gamma_1, a{:}\forall \mathbf{x} A, \Gamma_2) \longrightarrow (\Gamma_1, a[\mathbf{t}/\mathbf{x}]{:}A, \Gamma_2)$,
where $\mathbf{t}$ is an arbitrary RM.

Note that these two rules are also based on the left inference rules for quantifiers of the standard sequent calculus (ignoring labels);

$$\frac{\Gamma_1, A[\mathbf{u}/\mathbf{x}], \Gamma_2 \vdash C}{\Gamma_1, \exists \mathbf{x} A[\mathbf{x}], \Gamma_2 \vdash C} \qquad \frac{\Gamma_1, A[\mathbf{t}/\mathbf{x}], \Gamma_2 \vdash C}{\Gamma_1, \forall \mathbf{x} A[\mathbf{x}], \Gamma_2 \vdash C}$$

where $\mathbf{u}$ does not occurs freely in the lower sequent; where $\mathbf{t}$ is arbitrary.

Thus RM $\mathbf{u}$ introduced by (Introduce-RM) corresponds to an *eigenvariable* of the sequent calculus.

Having specified the operational behavior of types with labels, we are ready to describe how the DRS construction looks like in our framework. Let us consider the following text;

(3) A man entered. He smiled.

We would like to construct a DRS expressing the semantic content of text (1). For the moment, let us assume the following lexicon;

a $\Rightarrow \lambda PQ.(\mathbf{x}; P(\mathbf{x}); Q(\mathbf{x})) : \exists \mathbf{x}((s/(np\backslash s))/n)$
no $\Rightarrow \lambda PQ.\neg(\mathbf{x}; P(\mathbf{x}); Q(\mathbf{x})) : \exists \mathbf{x}((s/(np\backslash s))/n)$
man $\Rightarrow man : n$
entered $\Rightarrow entered : np\backslash s$
smiled $\Rightarrow smiled : np\backslash s$
. $\Rightarrow \lambda pq.(p; q) : s\backslash(txt/s)$
he $\Rightarrow \mathbf{x} : \forall \mathbf{x} np$

What we have to do first is to list the lexical entries corresponding to the words occurring in the sentence in the same order;

$$\left(\begin{array}{ll} \lambda PQ.(\mathbf{x}; P(\mathbf{x}); Q(\mathbf{x})) & :\exists \mathbf{x}((s/(np\backslash s))/n) \\ man & :n \\ entered & :np\backslash s \\ \lambda pq.(p; q) & :s\backslash(txt/s) \\ \mathbf{x} & :\forall \mathbf{x} np \\ smiled & :np\backslash s \end{array}\right).$$

Then, we successively apply the transition rules above to this list; (For each step of computation, relevant parts of the configuration are indicated by underlines.)

$$\left(\begin{array}{ll} \underline{\lambda PQ.(\mathbf{x}; P(\mathbf{x}); Q(\mathbf{x}))} & \underline{:\exists \mathbf{x}((s/(np\backslash s))/n)} \\ man & :n \\ entered & :np\backslash s \\ \lambda pq.(p; q) & :s\backslash(txt/s) \\ \mathbf{x} & :\forall \mathbf{x} np \\ smiled & :np\backslash s \end{array}\right) \longrightarrow \text{(Introduce-RM)}$$

$$\left(\begin{array}{ll} \underline{\lambda PQ.(\mathbf{u}; P(\mathbf{u}); Q(\mathbf{u}))} & \underline{:(s/(np\backslash s))/n} \\ \underline{man} & :\underline{n} \\ entered & :np\backslash s \\ \lambda pq.(p; q) & :s\backslash(txt/s) \\ \mathbf{x} & :\forall \mathbf{x} np \\ smiled & :np\backslash s \end{array}\right) \longrightarrow \text{(Receive-Right)}$$

$$\left(\begin{array}{ll}\underline{\lambda Q.(\mathbf{u};man(\mathbf{u});Q(\mathbf{u}))} & :\underline{s/(np\backslash s)}\\ \underline{entered} & :\underline{np\backslash s}\\ \lambda pq.(p;q) & :s\backslash(txt/s)\\ \mathbf{x} & :\forall\mathbf{x}np\\ smiled & :np\backslash s\end{array}\right)\longrightarrow \text{(Receive-Right)}$$

$$\left(\begin{array}{ll}\underline{\mathbf{u};man(\mathbf{u});entered(\mathbf{u})} & :\underline{s}\\ \underline{\lambda pq.(p;q)} & :\underline{s\backslash(txt/s)}\\ \mathbf{x} & :\forall\mathbf{x}np\\ smiled & :np\backslash s\end{array}\right)\longrightarrow \text{(Receive-Left)}$$

$$\left(\begin{array}{ll}\lambda q.(\mathbf{u};man(\mathbf{u});entered(\mathbf{u});q) & :txt/s\\ \underline{\mathbf{x}} & :\underline{\forall\mathbf{x}np}\\ smiled & :np\backslash s\end{array}\right)\longrightarrow \text{(Catch-RM)}$$

$$\left(\begin{array}{ll}\lambda q.(\mathbf{u};man(\mathbf{u});entered(\mathbf{u});q) & :txt/s\\ \underline{\mathbf{u}} & :\underline{np}\\ \underline{smiled} & :\underline{np\backslash s}\end{array}\right)\longrightarrow \text{(Receive-Right)}$$

$$\left(\begin{array}{ll}\underline{\lambda q.(\mathbf{u};man(\mathbf{u});entered(\mathbf{u});q)} & :\underline{txt/s}\\ \underline{smiled(\mathbf{u})} & :\underline{s}\end{array}\right)\longrightarrow \text{(Receive-Right)}$$

$$(\mathbf{u};man(\mathbf{u});entered(\mathbf{u});smiled(\mathbf{u}):txt)$$

In this way we obtain

(4) $\mathbf{u};man(\mathbf{u});entered(\mathbf{u});smiled(\mathbf{u})$.

as one of the readings of (3) according to which 'A man' is the antecedent of 'he'. Since (Catch-RM) can choose an arbitrary RM to instantiate $\mathbf{x}$, we also have

(5) $\mathbf{u};man(\mathbf{u});entered(\mathbf{u});smiled(\mathbf{v})$.

as an interpretation of (3).

As (Introduce-RM) introduces a new RM which does not occur in the context, there is no danger of *variable clashes*. Hence we do not have to use any complicated merging operation as in [vEK97].

We should give an assurance that the following texts never yield DRSs representing the indicated anaphoric links, which are obviously unacceptable.

(6) *[No man]^1 entered. He$_1$ smiled.

(7) *He$_1$ entered. [A man]^1 smiled.

(8) *[A man]^1 entered. She$_1$ smiled.

To achieve this, we impose certain constraints on application of (Receive-Left) and (Receive-Right):

> (Receive-Left) $(\Gamma_1, a:A, b:A\backslash B, \Gamma_2)\longrightarrow(\Gamma_1, b\bullet a:B, \Gamma_2)$ and
> (Receive-Right) $(\Gamma_1, b:B/A, a:A, \Gamma_2)\longrightarrow(\Gamma_1, b\bullet a:B, \Gamma_2)$
> are applicable only if $b\bullet a$ results in a 'correct' DRS (possibly λ-abstracted) with respect to certain accessibility criteria (in the sense which will be clarified in the next section).

The DRS-construction for (6) proceeds almost in the same way as that for (3), except the very last step of computation;

$$\left(\begin{array}{ll}\underline{\lambda q.\neg(\mathbf{u};man(\mathbf{u});entered(\mathbf{u});q)} & :\underline{txt/s}\\ \underline{smiled(\mathbf{u})} & :\underline{s}\end{array}\right)\not\longrightarrow \text{(Receive-Right)}$$

(9) $(\neg(\mathbf{u}; man(\mathbf{u}); entered(\mathbf{u})); smiled(\mathbf{u}) : txt)$

Here, the application of (Receive-Right) gives rise to an occurrence of reference marker $\mathbf{u}$ which violates the accessibility criterion, so the reading of text (6) according to which 'No man' antecedes 'He' is successfully ruled out. This criterion will be made clear in the next section.

In the same way, (7) is out due to the violation of the accessibility criteria. In (8) the gender of 'She' disagrees with the indicated antecedent. In order to prevent the pronoun from being linked with 'A man', we have to rule out DRSs which have inconsistent gender assignments during computation. This could be easily achieved by imposing further constraints on application of (Receive-Left) and (Receive-Right), but we will not consider such constraints in this paper because of simplicity.

So far are our basic ideas. Of course things should be made precise in order to develop the ideas further. In particular, of crucial importance is to give the precise definition of the 'correct' DRSs, because it plays the central role to rule out incorrect readings with illegal anaphoric links.

3 Discourse Representation Structures

The aim of this section is to give a precise definition of 'correct' DRSs. To do so, we follow the approach of [vEK97], but our correct DRSs (called *strict* DRSs) are more restrictive than *proper* DRSs of [vEK97] in that it is sensitive not only to the overlap of *introduced* RMs and *fixed* RMs, but also the overlap of introduced or fixed RMs and *classically bound* RMs. We also consider higher order generalization of strict DRSs using λ-calculus (called *legal* $\lambda pDRSs$), which are also simply called *labels* from the viewpoint of our labelled sequent calculus in the next section.

Definition 1 (Proto-DRSs ([vEK97])) Let A be a set of *constant reference markers* (or *constants*) and U be a set of *variable reference markers* (or *reference marker variables*). U is ranged over by $\mathbf{u}, \mathbf{v}, \mathbf{x}, \mathbf{y}, \ldots$ and $A \cup U$ by $\mathbf{t}, \mathbf{t_1}, \mathbf{t_2}, \ldots$. The language of Proto-DRSs (or pDRSs) is defined as follows;

$$D ::= \delta\mathbf{v} \mid \top \mid P\mathbf{t_1} \cdots \mathbf{t_n} \mid \mathbf{v} \doteq \mathbf{t} \mid \neg D \mid (D_1; D_2).$$

Implication and disjunction are introduced as abbreviations;

1. $D_1 \Rightarrow D_2 \overset{def}{\Leftrightarrow} \neg(D_1; \neg D_2)$;
2. $D_1 \vee D_2 \overset{def}{\Leftrightarrow} \neg(\neg D_1; \neg D_2)$.

We indicate, borrowing notation from [KKP95], an introduced occurrence of a reference marker $\mathbf{v}$ as $\delta\mathbf{v}$, because we need to distinguish a marker (of type e) and a DRS consisting of an introduced marker (of type T) type-theoretically. However, we shall omit δ and simply write $\mathbf{v}$ whenever there is no danger of confusion.

We need to distinguish three types of marker occurrences (see [vEK97]);

1. marker occurrences that get their reference fixed by the larger context,
2. marker occurrences that get introduced in the current context,
3. marker occurrences that get introduced in a subordinate context.

The following definition is given in [vEK97] to state the distinction in a rigorous way;

Definition 2 (var, fix, $intro$, $cbnd$, $activ$ ([vEK97]))

1. $var(P\mathbf{t_1} \cdots \mathbf{t_n}) = \{\mathbf{t_i} | 1 \leq i \leq n, \mathbf{t_i} \in U\}$, $var(\mathbf{v} \doteq \mathbf{t}) = \begin{cases} \{\mathbf{v}, \mathbf{t}\} & \text{if } \mathbf{t} \in U, \\ \{\mathbf{v}\} & \text{otherwise.} \end{cases}$

2. fix (fixed RMs), $intro$ (introduced RMs), $cbnd$ (classically bound RMs): $pDRSs \longrightarrow \mathcal{P}U$ are defined as follows;

- $fix(\delta\mathbf{v}) = \emptyset, intro(\delta\mathbf{v}) = \{\mathbf{v}\}, cbnd(\delta\mathbf{v}) = \emptyset$;
- $fix(\top) = \emptyset, intro(\top) = \emptyset, cbnd(\top) = \emptyset$;
- $fix(P\mathbf{t_1}\cdots\mathbf{t_n}) = var(P\mathbf{t_1}\cdots\mathbf{t_n}), intro(P\mathbf{t_1}\cdots\mathbf{t_n}) = \emptyset, cbnd(P\mathbf{t_1}\cdots\mathbf{t_n}) = \emptyset$;
- $fix(\mathbf{v}\doteq\mathbf{t}) = var(\mathbf{v}\doteq\mathbf{t}), intro(\mathbf{v}\doteq\mathbf{t}) = \emptyset, cbnd(\mathbf{v}\doteq\mathbf{t}) = \emptyset$;
- $fix(\neg D) = fix(D), intro(\neg D) = \emptyset, cbnd(\neg D) = intro(D) \cup cbnd(D)$;
- $fix(D_1;D_2) = fix(D_1) \cup (fix(D_2) - intro(D_1)), intro(D_1;D_2) = intro(D_1) \cup intro(D_2), cbnd(D_1;D_2) = cbnd(D_1) \cup cbnd(D_2)$.

Definition 3 (*cond* ([vEK97])) $cond : pDRSs \longrightarrow \mathcal{P}(pDRSs)$ is defined as follows;

- $cond(\delta\mathbf{v}) = \emptyset$;
- $cond(\top) = \{\top\}$, $cond(P\mathbf{t_1}\cdots\mathbf{t_n}) = \{P\mathbf{t_1}\cdots\mathbf{t_n}\}$, $cond(\mathbf{v}\doteq\mathbf{t}) = \{\mathbf{v}\doteq\mathbf{t}\}$, $cond(\neg D) = \{\neg D\}$;
- $cond(D_1;D_2) = cond(D_1) \cup cond(D_2)$.

Definition 4 (subordinate pDRSs) A pDRS D' is said to be a *subordinate* pDRS of D if D contains $\neg D'$ as subterm, or D' is D itself.

In [vEK97], the notion of proper DRS is defined. The definition essentially says that a pDRS D is proper if for every subordinate pDRS D_i of D, $fix(D_i)$ and $intro(D_i)$ are disjoint and no RM is introduced more than once in D_i. For example, the following are not proper DRSs;

(10) $\mathbf{x};\mathbf{x};P\mathbf{x}$

(11) $P\mathbf{x};\mathbf{x}$

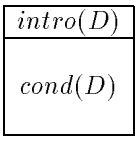
(12) $\neg(P\mathbf{x});\mathbf{x}$

On the other hand, The following are DRSs;

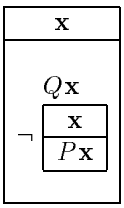
(13) $\mathbf{x};Q\mathbf{x};\neg(\mathbf{x};P\mathbf{x})$

(14) $\neg(\mathbf{x};P\mathbf{x});\mathbf{x}$

If D is a proper DRS, we can replace a sub-DRS of the form $D';\mathbf{u}$ with $\mathbf{u};D'$ without changing its meaning (in terms of dynamic semantics). This means that every D can be represented in box format as



For example, DRS (13) is represented in box format as



For our purpose, however, the constraint on proper DRS is still too liberal, because it says nothing about the overlap of classically bound markers and fixed or introduced markers. We have already seen in the previous section (example text (6)) one of the reasons why we have to distinguish marker occurrences more strictly. We will see the other reasons in the later sections. Now we introduce the notion of *strict* DRS.

Definition 5 (Strict DRSs)

1. $\delta\mathbf{v}$, $\top$, $P\mathbf{t_1}\cdots\mathbf{t_n}$, $\mathbf{v}\doteq\mathbf{t}$, $\langle\mathbf{t_1}\neq\mathbf{t_2}\rangle$ are strict DRSs;
2. If D is a strict DRS, then $\neg D$ is a strict DRS;
3. If D_1 and D_2 are strict DRSs, and (i) $cbnd(D_1)\cap fix(D_2)=\emptyset$, (ii) $(fix(D_1)\cup intro(D_1)\cup cbnd(D_1))\cap intro(D_2)=\emptyset$ and (iii) $(fix(D_1)\cup intro(D_1))\cap cbnd(D_2)=\emptyset$, then $(D_1;D_2)$ is a strict DRS, too.

The definition looks rather complicated, but the intuition can be made clear in terms of the two conditions below: purity and the non-destructiveness. But before we come on to that, we have to assure that the composition is still associative.

Proposition 1 *If $(D_1;D_2);D_3$ is a strict DRS, then so is $D_1;(D_2;D_3)$.*

It is immediately observed that $(D_1;D_2);D_3$ and $D_1;(D_2;D_3)$ have one and the same meaning in terms of dynamic semantics. Therefore we can safely omit the parentheses of $(D_1;D_2);D_3$ and $D_1;(D_2;D_3)$ and denote them as $D_1;D_2;D_3$.

Consider the following two conditions;

(Purity) A pDRS D is *pure* if $fix(D)$, $intro(D)$ and $cbnd(D)$ are pairwise disjoint.

(Non-destructiveness) A pDRS D is *non-destructive* if D is not of the form $D_1;\mathbf{v};D_2;\mathbf{v};D_3$, i.e., no reference marker is introduced more than once.

In view of these conditions, our strict DRSs are characterized as follows;

Proposition 2 *A pDRS D is strict if and only if every subordinate pDRS D' of D is pure and non-destructive.*

Hence (13) and (14) are not strict DRSs because they are not pure. On the other hand, the followings is a strict DRS in our sense;

(15) $(\mathbf{x};P\mathbf{x})\vee(\mathbf{x};Q\mathbf{x})$.

The following pDRS (9) given in the previous section,

$$(\neg(\mathbf{u};man(\mathbf{u});entered(\mathbf{u}));smiled(\mathbf{u}):txt),$$

is ruled out because it is not pure. ($\mathbf{u}$ is classically bound and fixed at the same time.)

Now we consider a typed λ-calculus based on the language of pDRSs.

Definition 6 (λpDRSs)
1. The types $\mathbf{T}$ of λpDRS are constructed from e (for entities) and T (for transitions) using $\rightarrow$, thus;

$$\mathbf{T} ::= e \mid T \mid \mathbf{T}\rightarrow\mathbf{T}.$$

2. The constants of λpDRS are

- Reference markers $\mathbf{u}$,$\mathbf{v}$,... of type e;
- For each $n\geq 0$, n-ary predicate symbols $P,Q,\ldots$ of type $\underbrace{e\rightarrow\cdots e\rightarrow}_{n\text{ times}}T$;

- $\dot{=} : e \rightarrow e \rightarrow T$;
- $\top : T$;
- $\neg : T \rightarrow T$;
- $; : T \rightarrow T \rightarrow T$;
- $\delta : e \rightarrow T$ (RM-introduction operator).

The terms of λpDRS are typed λ terms over the above constants and the set of λ-term variables V_t for each type $t \in \mathbf{T}$. λpDRS terms are denoted as $a, b, c \ldots$. We shall, however, omit the details (see [Bar92]).

As noted before, we often omit δ.

Note that the set $V = \bigcup_{t \in \mathbf{T}} V_t$ of λ-term variables and the set U of reference marker variables are disjoint: λx binds only λ-term variables, so no expression like $\lambda \mathbf{x} M$ is allowed in our system. As seen in the above definition, reference markers (including marker variables) behave as constants from the viewpoint of λ-calculus.

Definition 7 (Linear terms, Relevant terms) A λ-term a is called *linear* if each λx in a binds exactly one occurrence of x in a (see [Bar92]). A λ-term a is called *relevant* if each λx in a binds at least one occurrence of x in a.

For example, $\lambda x.xx$ is relevant, while $\lambda xy.x$ is not.

Having introduced these notions, we can now define our central notion of legal λpDRS term. The legal λpDRS terms serve as the labels of our labelled sequent calculus which will be defined in the next section.

Definition 8 (Legal λpDRS terms) A λpDRS term is *legal* if

1. it is relevant,
2. it is in $\beta\eta$-normal form, and
3. it does not contain a non-strict pDRS as subterm.

The set of legal λpDRS terms is denoted by $\mathbf{L}$.

Relevance is required to prove Lemma 1(3) and 2(2) which are in turn required to prove the cut-elimination theorem (Theorem 1). The relevance condition might be considered too restrictive at a first look, but we do not find any serious difficulty in practice, because usually a lexical entry is described by a relevant λ-term, and the class of relevant λ-terms is closed under application, substitution and $\beta\eta$-reduction. Moreover, our logical basis is Lambek calculus which lacks the structural rules. Hence, due to the lack of weakening, logical deductions in the calculus do not create non-relevant λ terms.

Definition 9 ($a \bullet b$, $\lambda x \bullet a$) We define two partial binary operations on $\mathbf{L}$; $_ \bullet _ : \mathbf{L} \times \mathbf{L} \longrightarrow \mathbf{L}$ and $\lambda _ \bullet _ : V \times \mathbf{L} \longrightarrow \mathbf{L}$, as follows;

$$\begin{array}{rcll} a \bullet b & = & nf(ab) & \text{if } ab \text{ is well-typed and } nf(ab) \in \mathbf{L}; \\ & = & \text{undefined} & \text{otherwise.} \\ \lambda x \bullet a & = & nf(\lambda x.a) & \text{if } nf(\lambda x.a) \in \mathbf{L}; \\ & = & \text{undefined} & \text{otherwise.} \end{array}$$

where $nf(a)$ denotes the $\beta\eta$-normal form of a.

$a[b/\bullet x]$ is defined to be $nf(a[b/x])$ if $nf(a[b/x]) \in \mathbf{L}$ and is undefined otherwise (where $a[b/x]$ denotes the usual substitution of λ-calculus (see [Bar92])). The following lemma is quite useful, together with the next one (Lemma 2 below), when we define our labelled calculus in the next section.

Lemma 1

1. *If $a \in \mathbf{L}$, then $x \bullet a$ and $a \bullet x$ are defined.*
2. *Let $x \in FV(a)$. Then, $\lambda x \bullet a \in \mathbf{L}$ if and only if $a \in \mathbf{L}$ (where $FV(a)$ denotes the set of λ-term variables occurring freely in a).*
3. *if a, b and c are in $\mathbf{L}$ and $(a \bullet b)[c/\bullet x]$ is defined, then $a[c/\bullet x]$ and $b[c/\bullet x]$ are also defined.*

Proof. (1) and (2) are easily shown. As for (3), observe that there is a $\beta\eta$-reduction sequence

$$a[c/\bullet x]b[c/\bullet x] \longrightarrow_{\beta\eta} \cdots \longrightarrow_{\beta\eta} (a \bullet b)[c/\bullet x]$$

by the confluency of λ-calculus. Hence it suffices to show that

(*) If $d \longrightarrow_{\beta\eta} e$ and d contains an illegal pDRS, then so does e.

It easily follows from the relevance of d. ■

Definition 10 (Renamings, Injective Renamings, Variants ([vEK97])) A *marker renaming* is a function $\theta : U \longrightarrow U$, such that its domain $Dom(\theta) = \{\mathbf{v} \in U | \mathbf{v} \neq \theta(\mathbf{v})\}$ is finite. θ is also denoted as $[\mathbf{v}/\mathbf{u}]$ if $Dom(\theta) = \{\mathbf{u}\}$ and $\theta(\mathbf{u}) = \mathbf{v}$. If $X \subseteq U$ then $\theta X = \{\theta(\mathbf{x}) | \mathbf{x} \in X\}$. A marker renaming θ is *injective* on X if $|X| = |\theta X|$.

A marker renaming θ is extended to $\theta : \lambda pDRSs \longrightarrow \lambda pDRSs$ so that for a given a, $a\theta$ is obtained by replacing each occurrences of $\mathbf{x} \in Dom(\theta)$ in a with $\theta(\mathbf{x})$. Hence we may consider a marker renaming θ as a partial function $\theta : \mathbf{L} \longrightarrow \mathbf{L}$, and say that, given $a \in \mathbf{L}$, $a\theta$ is *defined* if $a\theta \in \mathbf{L}$. Note that marker renaming of a $\mathbf{L}$-term might not result in a $\mathbf{L}$-term (due to the strictness condition).

a' is said to be a *variant* of a if there is a marker renaming θ injective on the set of RMS occurring in a such that $a\theta = a'$; in this case, as easily seen, a is a variant of a' as well.

Lemma 2

1. *if $a \in \mathbf{L}$ and a' is a variant of a, then $a' \in \mathbf{L}$.*
2. *if $(a \bullet b)\theta$ is defined, then $a\theta$ and $b\theta$ are also defined.*

Proof. (1) is easily shown. (2) follows from (*) in the proof of Lemma 1. ■

4 Sequent Calculus Lq

In this section we shall define sequent calculus **Lq**. **Lq** is based on the product-free Lambek Calculus, enriched with Moortgat's scoping operator ($\Uparrow$) ([Moo88], [Moo91]) and quantifiers ($\forall$, $\exists$). In addition, legal λpDRS terms, called *labels* in the rest of this paper, are attached to syntactic categories. Thus, our calculus **Lq** may be understood as a sort of labelled sequent calculi. One of the striking features of our calculus is that the (categorial) quantifiers bind reference marker variables occurring in labels. The cut elimination theorem holds for **Lq** (Theorem 1).

Definition 11 (Syntactic Categories)

1. Basic syntactic categories(**Bas**). Our basic categories are s, np, n, txt and their predicational counterparts; i.e. if $p \in \{s, np, n, txt\}$, $k \geq 0$ and $\mathbf{u_1}, \ldots, \mathbf{u_k}$ are RMs, then $p(\mathbf{u_1}, \ldots, \mathbf{u_k}) \in$ **Bas**.
2. Quantifier-free syntactic categories(**Qf**).
   Qf ::= **Bas** | **Qf****Qf** | **Qf**/**Qf** | **Qf** $\Uparrow$ **Qf**.
3. Syntactic categories(**Cat**).
   $A \in$ **Cat** if A is of the form $Q_1\mathbf{x_1} \cdots Q_n\mathbf{x_n}B$ where $B \in$ **Qf**, $Q_i \in \{\forall, \exists\}$, $\mathbf{x_i} \in U$(the set of marker variables) for $1 \leq i \leq n$, and $\mathbf{x_i} \not\equiv \mathbf{x_j}$ for $i \neq j$.

Remark 1
1. In this paper we do not make use of predicational categories, thus we only use s, np, n, txt as basic categories. However, predicational categories are useful to represent feature structures and some grammatical constraints, hence we also include predicational categories in the formal definition of syntactic categories.

2. In the present formalism, syntactic categories are always in prenex form, i.e., quantifiers do not occur inside propositional connectives ($\backslash$, $/$, $\Uparrow$). This is because our quantifiers may bind marker variables occurring in labels, so if we allowed nested occurrences of quantifiers, we would have no way to determine the scopes of quantifiers in labels.

To each syntactic category is assigned a semantic type of $\lambda pDRS$, as follows;

Definition 12 (*type*) A function $type : \mathbf{Cat} \longrightarrow \mathbf{T}$ is defined as follows;

- $type(s(\mathbf{u_1}, \ldots, \mathbf{u_k})) = type(txt(\mathbf{u_1}, \ldots, \mathbf{u_k})) = T$;
- $type(np(\mathbf{u_1}, \ldots, \mathbf{u_k})) = e$; $type(n(\mathbf{u_1}, \ldots, \mathbf{u_k})) = e \rightarrow T$;
- $type(A\backslash B) = type(B/A) = type(A) \rightarrow type(B)$;
- $type(A \Uparrow B) = (type(A) \rightarrow type(B)) \rightarrow type(B)$;
- $type(\forall \mathbf{x} A) = type(\exists \mathbf{x} A) = type(A)$.

Definition 13 (Labels, Declarative units, Sequents) An element of $\mathbf{L}$ is called a *label*. A *declarative unit* (or a *unit*) is of the form $a : A$ where $A \in \mathbf{Cat}$ and a is a label of type $type(A)$.
$\Gamma, \Delta, \Pi, \ldots$ range over finite lists of units (including the empty list).
A *sequent* is of the form $\Gamma \vdash a : A$.

Definition 14 (Bound reference markers, Free reference markers) Let $a[\mathbf{u}] : Q_1 \mathbf{x_1} \ldots Q_n \mathbf{x_n} A$ be a unit, where $Q_i \in \{\forall, \exists\}$ and A quantifier-free. Then $\mathbf{u}$ is *bound* in the unit if $\mathbf{u} = \mathbf{x_i}$ for some i. Otherwise $\mathbf{u}$ is *free* in the unit.

The inference rules of system **Lq** are given in Figure 1.

Remark 2
1. In $\backslash L$, $/L$ and $\Uparrow L$, it is implicitly assumed that the rules are applicable only if $b \bullet a$ is defined. In the same way, $\forall L$ is applicable only if $a[\mathbf{t/x}] \in \mathbf{L}$. On the other hand, it is assured by Lemma 1(1), (2) and Lemma 2(1) that $a \bullet x$ in $\backslash R$, $/R$ and $\Uparrow L$, $\lambda x \bullet xa$ in $\Uparrow R$, $a[\mathbf{u/x}]$ in $\exists RL$ and $\forall R$ are always defined.

2. For existential quantification, we have only one inference rule in which the introductions of $\exists$ to the left hand side and to the right hand side are synchronized. It is not desirable since the expressivity of the calculus is too restricted. Hence it would be much better if we could divide the rule into the following two;

$$\frac{\Gamma_1, a[\mathbf{u/x}] : A[\mathbf{u/x}], \Gamma_2 \vdash c : C}{\Gamma_1, a : \exists \mathbf{x} A, \Gamma_2 \vdash c : C}\ \exists L \qquad \frac{\Gamma \vdash a[\mathbf{u/x}] : A[\mathbf{u/x}]}{\Gamma \vdash a : \exists \mathbf{x} A}\ \exists R$$

But then we would have to give up the cut elimination theorem (Theorem 1) and the operational completeness theorem (Theorem 2). Since these two theorems are so fundamental in our framework, we adopt the synchronized rule ($\exists LR$).

3. $\backslash R$ and $/R$ are equivalent to the following $\backslash R'$ and $/R'$ under the assumption $x \in FV(b)$;

$$\frac{x : A, \Gamma \vdash b : B}{\Gamma \vdash \lambda x \bullet b : A\backslash B}\ \backslash R', \qquad \frac{\Gamma, x : A \vdash b : B}{\Gamma \vdash \lambda x \bullet b : B/A}\ /R',$$

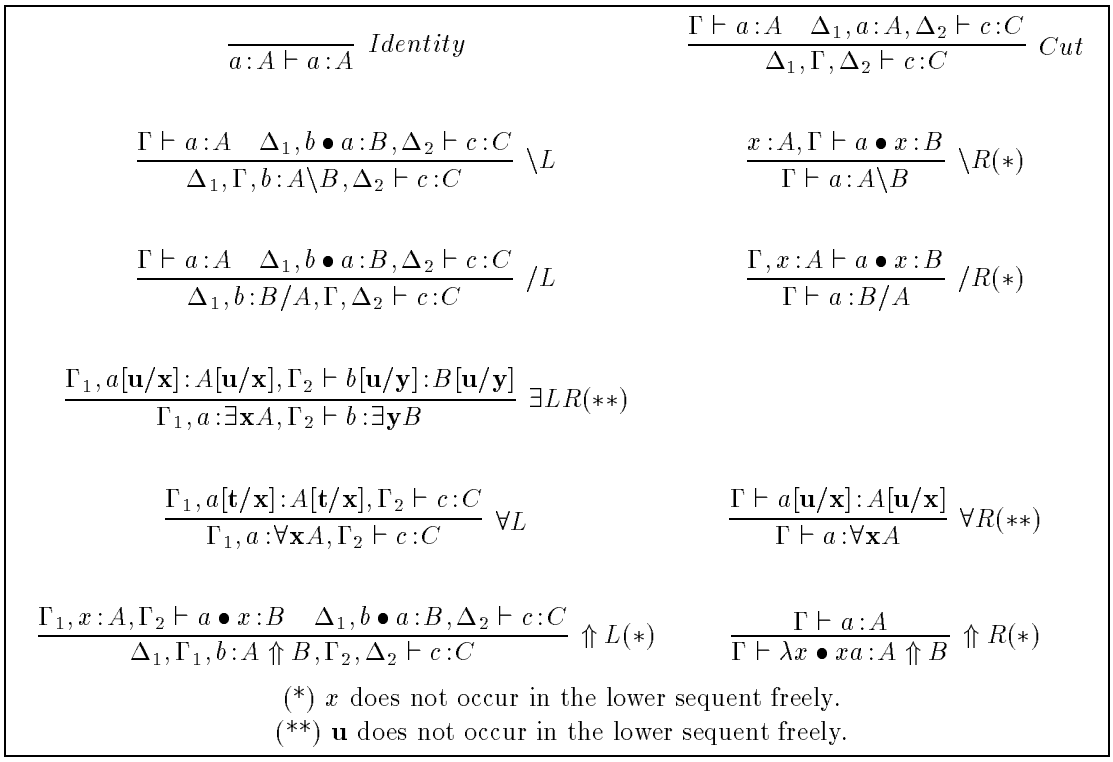


Figure 1: Inference Rules of System **Lq**

For one direction, since $x \in FV(a \bullet x)$ always hold because of the relevance of a, we have

$$\frac{x:A, \Gamma \vdash a \bullet x:B}{\Gamma \vdash \lambda x \bullet (a \bullet x):A\backslash B},$$

and as easily seen, $\lambda x \bullet (a \bullet x) = a$ (by the confluency of lambda calculus). For the other direction, since we are assuming $x \in FV(b)$, we see $b = (\lambda x \bullet b) \bullet x$. Hence $\backslash R'$ is a particular instance of $\backslash R$. We adopt, however, $\backslash R$ and $/R$ for the official inference rules of **Lq** in order to make the number of assumptions least.

The fundamental theorem of **Lq** is

Theorem 1 (Cut Elimination Theorem for Lq) *If S is provable in* **Lq**, *then S has a cut-free proof in* **Lq**.

The proof is given in Appendix.

5 Operational Semantics for Lq

The sequent calculus **Lq** provides a way of categorial deduction. However, the calculus itself says very little about how to compute DRSs from a given list of lexical entries. In order to describe how the computation goes, we shall give an operational semantics to **Lq** in terms of a transition system. The operational semantics should specify how a declarative unit behaves *in the correlation to other declarative units*. Therefore the transition relation should be defined over the set of configurations, rather than the set of units. We shall prove the completeness of the operational semantics with respect to **Lq**, which will establish a tight correspondence between logic (**Lq** sequent calculus) and computation (the operational rules).

Definition 15 (Meta-variables, Meta-substitutions) We assume that the set $W = \bigcup_{t \in \mathbf{T}} W_t$ of (typed) *meta-variables* are given which is disjoint with U and V. For each type t, a meta-variable $X_0^t \in W_t$ is fixed and called an *initial meta-variable* (in the sequel we simply write X_0 to denote X_0^t for any t). A *meta-substitution* τ is a list of the form $[a_1/X_1, \ldots, a_n/X_n]$, where X_i is a meta-variable, a_i is a label possibly containing some meta-variables[2] and X_i and a_i are of the same type. For τ above, $Dom(\tau)$ denotes the set $\{X_1, \ldots, X_n\}$. The empty list is denoted by $[]$, and the concatenation of two lists τ_1 and τ_2 is denoted by $\tau_1 \circ \tau_2$. Given a label (possibly containing meta-variables) a and a meta-substitution τ, $a\tau$ is defined as follows;

1. if $\tau = []$, then $a\tau = a$;
2. if $\tau = [b/X]$, then $x\tau = x$; $X\tau = b$; $Y\tau = Y (Y \not\equiv X)$, $(cd)\tau = (c\tau)(d\tau)$; $(\lambda x.c)\tau = \lambda x.(c\tau)$;
3. if $\tau = ([b/X] \circ \tau')$, then $a\tau = (a[b/X])\tau'$.

Finally we define $a \bullet \tau$ by

$$\begin{array}{lcll} a \bullet \tau & = & nf(a\tau) & \text{if } nf(a\tau) \in \mathbf{L}; \\ & = & undefined & otherwise. \end{array}$$

Note that, according to the above definition, substitution of a label a may result in an increase in the number of bound λ-term variables; for example $(\lambda y.X) \bullet [y/X] = (\lambda y.X)[y/X] = \lambda y.y$. Hence the meta-substitution should be distinguished from the substitution in the sense of λ-calculus.

Definition 16 (Configuration) A *configuration* is either a pair of the form $(\Gamma \vdash X : A; \tau)$ or a pair of the form $(\mathbf{suc}; \tau)$, where A is a quantifier-free category, X is a meta-variable of type $type(A)$ and τ is a meta-substitution. An *initial configuration* is a configuration of the form $(\Gamma \vdash X_0 : A; [])$.

u occurs freely in a configuration $(\Gamma \vdash X : A; \tau)$ if either it occurs freely in $\Gamma \vdash X : A$ or it occurs in some $a \in Range(\tau)$.

Informally, configuration $(\Gamma \vdash X : A; \tau)$ describes the current state of computation, where Γ represents the state of units, A represents the goal category toward which the computation goes, and τ represents the state of the store; it may be considered as a type-logical reconstruction of the Cooper's storage. Once an initial configuration $(\Gamma_0 \vdash X_0 : A_0; [])$ reaches $(\mathbf{suc}; \tau)$, $X_0 \bullet \tau$ will give the result.

The operational semantics is given based on the transition rules (called *operational rules*) on configurations in Figure 2. Those rules are basically obtained by reading the inference rules of **Lq** in a bottom-up manner. We attached to each operational rule the name of the corresponding inference rule of **Lq**. In Figure 2, for readability, irrelevant parts of configurations are omitted as $\ldots$; thus $(\ldots, a : A, \ldots \vdash X : C; \tau)$ abbreviates $(\Gamma_1, a : A, \Gamma_2 \vdash X : C; \tau)$ for some Γ_1 and Γ_2. Formally;

Definition 17 We simultaneously define two binary relations $\longrightarrow$ (one-step reduction) and $\longrightarrow\!\!\!\!\!\rightarrow$ (multiple-steps reduction) on configurations as the least relations satisfying the operational rules given in Figure 2, where inference figure $\dfrac{S_1, \ldots, S_n}{S}$ $(n \leq 0)$ means that if statement S_i holds for every $1 \leq i \leq n$, then statement S also holds. (In Figure 2, $\Phi_1, \Phi_2, \ldots$ stand for configurations.)

Definition 18 (Reachability relation $\mapsto$) Let Γ be a list of units. We say that Γ *reaches* $a : A$ $(\Gamma \mapsto a : A)$ if $(\Gamma \vdash X_0 : A; []) \longrightarrow\!\!\!\!\!\rightarrow (\mathbf{suc}; \tau)$ and $a = X_0 \bullet \tau$.

Remark 3
1. (Zoom-In), (Receive-Left) and (Receive-Right) embody a sub-computation mechanism. For instance, (Receive-Left) should be read as:

[2] The set of labels containing meta-variables is defined in a natural way.

(Introduce-RM) $(\exists L)$

$$\frac{\mathbf{u} \text{ does not occur freely in } (\ldots, a{:}\exists \mathbf{x} A, \ldots \vdash X_0{:}C;\tau)}{(\ldots, a{:}\exists \mathbf{x} A, \ldots \vdash X_0{:}C;\tau) \longrightarrow (\ldots, a[\mathbf{u}/\mathbf{x}]{:}A[\mathbf{u}/\mathbf{x}], \ldots \vdash X_0{:}C;\tau)}$$

(Catch-RM) $(\forall L)$

$$\frac{a[\mathbf{t}/\mathbf{x}] \in \mathbf{L}}{(\ldots, a{:}\forall \mathbf{x} A, \ldots \vdash X{:}C;\tau) \longrightarrow (\ldots, a[\mathbf{t}/\mathbf{x}]{:}A[\mathbf{t}/\mathbf{x}], \ldots \vdash X{:}C;\tau)}$$

(Hypothesize-Left) $(\backslash R)$

$$\frac{}{(\ldots \vdash X{:}A\backslash B;\tau) \longrightarrow (x{:}A, \ldots \vdash Y{:}B;\tau \circ [\lambda x.Y/X])} \ (*)(**)$$

(Hypothesize-Right) $(/R)$

$$\frac{}{(\ldots \vdash X{:}B/A;\tau) \longrightarrow (\ldots, x{:}A \vdash Y{:}B;\tau \circ [\lambda x.Y/X])} \ (*)(**)$$

(Receive-Left) $(\backslash L)$

$$\frac{(\Gamma \vdash Y{:}A;[]) \longrightarrow\!\!\!\!\!\rightarrow (\mathbf{suc};\tau') \quad a = Y \bullet \tau' \quad b \bullet a \text{ is defined} \quad Y \not\equiv X_0}{(\ldots, \Gamma, b{:}A\backslash B, \ldots \vdash X{:}C;\tau) \longrightarrow (\ldots, b \bullet a{:}B, \ldots \vdash X{:}C;\tau)}$$

(Receive-Right) $(/L)$

$$\frac{(\Gamma \vdash Y{:}A;[]) \longrightarrow\!\!\!\!\!\rightarrow (\mathbf{suc};\tau') \quad a = Y \bullet \tau' \quad b \bullet a \text{ is defined} \quad Y \not\equiv X_0}{(\ldots, b{:}B/A, \Gamma, \ldots \vdash X{:}C;\tau) \longrightarrow (\ldots, b \bullet a{:}B, \ldots \vdash X{:}C;\tau)}$$

(Zoom-In) $(\Uparrow L)$

$$\frac{(\Gamma_1, x{:}A, \Gamma_2 \vdash Y{:}B;[]) \longrightarrow\!\!\!\!\!\rightarrow (\mathbf{suc};\tau') \quad a \bullet x = Y \bullet \tau' \quad b \bullet a \text{ is defined} \quad Y \not\equiv X_0}{(\ldots, \Gamma_1, b{:}A \Uparrow B, \Gamma_2, \ldots \vdash X{:}C;\tau) \longrightarrow (\ldots, b \bullet a{:}B, \ldots \vdash X{:}C;\tau)} \ (*)$$

(No-Scope) $(\Uparrow R)$

$$\frac{}{(\ldots \vdash X{:}A \Uparrow B;\tau) \longrightarrow (\ldots \vdash Y{:}A;\tau \circ [\lambda x.xY/X])} \ (*)(**)$$

(Success) $(Identity)$

$$\frac{}{(a{:}A \vdash X{:}A;\tau) \longrightarrow (\mathbf{suc};\tau \circ [a/X])}$$

(One-Step) $\dfrac{\Phi_1 \longrightarrow \Phi_2}{\Phi_1 \longrightarrow\!\!\!\!\!\rightarrow \Phi_2}$

(Reflexivity) $\dfrac{}{\Phi_1 \longrightarrow\!\!\!\!\!\rightarrow \Phi_1}$

(Transitivity) $\dfrac{\Phi_1 \longrightarrow\!\!\!\!\!\rightarrow \Phi_2 \quad \Phi_2 \longrightarrow\!\!\!\!\!\rightarrow \Phi_3}{\Phi_1 \longrightarrow\!\!\!\!\!\rightarrow \Phi_3}$

(*) x a fresh λ variable which does not occur in the antecedent configuration.
(**) $Y \notin Dom(\tau) \cup \{X\}$ and $Y \not\equiv X_0$.

Figure 2: Operational Rules for **Lq**

"Suppose that we have configuration of the form $(\Delta_1, \Gamma, b : A\backslash B, \Delta_2 \vdash X : C; \tau)$. Then we may invoke a sub-computation starting from $(\Gamma \vdash Y : A; [])$, and if it reaches $(\mathbf{suc}; \tau')$, $a = Y \bullet \tau'$ and $b \bullet a$ is defined, then stop the sub-computation and continue the main computation at configuration $(\Delta_1, b \bullet a : B, \Delta_2 \vdash X : C; \tau)$".

2. Note that (Introduce-RM) is only applied to configurations with initial meta-variables X_0. This means that we can use (Introduce-RM) only in the main computation and only before (Hypothesize-Left), (Hypothesize-Right) and (No-Scope) are used (because we choose a fresh meta-variable to invoke a subcomputation or to use the above three rules).

3. As a special case of (Receive-Left), we have the one which has already stated in Section 2;

$(\Delta_1, a : A, b : A\backslash B, \Delta_2 \vdash X : C; \tau) \longrightarrow (\Delta_1, b \bullet a : B, \Delta_2 \vdash X : C; \tau)$
if $b \bullet a$ is defined;

because it is always true that $(a : A \vdash Y : A; []) \longrightarrow (\mathbf{suc}; [a/Y])$ and $Y[a/Y] = a$. Similarly we have

$(\Delta_1, b : B/A, a : A, \Delta_2 \vdash X : C; \tau) \longrightarrow (\Delta_1, b \bullet a : B, \Delta_2 \vdash X : C; \tau)$
if $b \bullet a$ is defined.

These derived rules are also referred to as (Receive-Left) and (Receive-Right), respectively.

4. We may add the following operational rule for $\Uparrow$, which is also referred to as (Zoom-In);

$(\ldots, b : A \Uparrow B, \ldots \vdash X : B; \tau) \longrightarrow (\ldots, x : A, \ldots \vdash Y : B; \tau \circ [b(\lambda x.Y)/X])$
where x is a fresh variable which does not occur in the antecedent configuration.

It can be shown that adding this rule does not change the reachability relation. The reason for this is that it corresponds to the following special case of inference rule $\Uparrow L$ of **Lq** sequent calculus;

$$\frac{\Gamma_1, x : A, \Gamma_2 \vdash a \bullet x : B \quad b \bullet a : B \vdash b \bullet a : B}{\Gamma_1, b : A \Uparrow B, \Gamma_2 \vdash b \bullet a : B} \Uparrow L(*)$$

In practice, we usually use this new rule rather than the original one because this rule captures the original intention of scoping operator $\Uparrow$ more clearly.

The following theorem states the exact correspondence between the **Lq** inference rules and the operational rules.

Theorem 2 (Completeness of Operational Semantics for Lq) *Let A be a quantifier-free category. $\Gamma \mapsto a : A$ if and only if $\Gamma \vdash a : \exists \mathbf{x_1} \ldots \exists \mathbf{x_n} A$ is provable in* **Lq** *for some marker variables $\mathbf{x_1} \ldots \mathbf{x_n}$.*

The theorem is proved in Appendix. Here it should be emphasized that the proof relies on the cut elimination theorem (Theorem 1) very crucially.

6 Examples

Having set up the calculus and investigated its formal properties, let us now turn to the examination of the usefulness and the expressivity of our calculus in practice. We shall show some examples of DRS construction. A close look at the actual process of DRS construction reveals several interesting aspects of our calculus. Our basic lexicon is given in Figure 3. Our list is, however, incomplete, and we occasionally use lexical entries which are not found in the list.

The first example is concerned with the use of the scoping operator. It best illustrates how the storage mechanism contributes to the DRS construction as a whole.

Consider the sentence (16).

(16) A student read every book.

Let us write Γ_0 to denote the list;

$$\begin{array}{ll} \lambda PQ.(\mathbf{x};P(\mathbf{x});Q(\mathbf{x})) & :\exists\mathbf{x}(np \Uparrow s)/n \\ student & :n \\ read & :(np\backslash s)/np \\ \lambda PQ.(\mathbf{x};P(\mathbf{x})) \Rightarrow Q(\mathbf{x}) & :\exists\mathbf{x}(np \Uparrow s)/n \\ book & :n \end{array}$$

We begin with initial configuration $(\Gamma_0 \vdash X_0 : s;[])$. Since (16) consists of a single sentence, we may take s as the goal category (of course, txt would do as well). Then, the derivation in Figure 4 yields the following DRS (17) corresponding to a reading in which the object takes wide scope over the subject;

(17) $(\mathbf{v};book(\mathbf{v})) \Rightarrow (\mathbf{u};student(\mathbf{u});read(\mathbf{v})(\mathbf{u}))$.

For the moment, we shall focus our attention to the transition from (d) to (e) by (Zoom-In) in Figure 4[3]. Using box format, the transition may be written as follows;

$$\text{(d)} \left(\begin{array}{ll} \lambda Q.(\mathbf{u};student(\mathbf{u});Q(\mathbf{u})) & :np \Uparrow s \\ read & :(np\backslash s)/np \\ \lambda Q.\boxed{\begin{array}{c}\mathbf{v}\\ book(\mathbf{v})\end{array}} \Rightarrow \boxed{\mathrm{Q}(\mathbf{v})} & :\underline{np \Uparrow s} \end{array} \vdash X_0 : s;[] \right) \longrightarrow \text{(Zoom-In)}$$

$$\text{(e)} \left(\begin{array}{ll} \underline{\lambda Q.(\mathbf{u};student(\mathbf{u});Q(\mathbf{u}))} & :\underline{np \Uparrow s} \\ read & :(np\backslash s)/np \\ x & :np \end{array} \vdash Y : s;[\lambda Q.\boxed{\begin{array}{c}\mathbf{v}\\ book(\mathbf{v})\end{array}} \Rightarrow \boxed{\mathrm{Q}(\mathbf{v})}(\lambda x.Y)/X_0] \right)$$

By (Zoom-In) the unit $\lambda Q.\boxed{\begin{array}{c}\mathbf{v}\\ book(\mathbf{v})\end{array}} \Rightarrow \boxed{\mathrm{Q}(\mathbf{v})} : np \Uparrow s$ sends its content (label) into the store (meta-substitution) and then becomes $x : np$. Correspondingly the store changes its state from the empty $[]$ to $[\lambda Q.\boxed{\begin{array}{c}\mathbf{v}\\ book(\mathbf{v})\end{array}} \Rightarrow \boxed{\mathrm{Q}(\mathbf{v})}(\lambda x.Y)/X_0]$. It looks as if we *entered* the box $\boxed{\mathrm{Q}(\mathbf{v})}$ by (Zoom-In) while the outside DRS were kept in the store. Indeed, it may be seen that the rest of computation is devoted to the construction of Q, and hence it proceeds completely in that box. It may also be observed that it is (Success) rule that allows us to *leave* the box. In this way, the scoping operator may be regarded as providing the *box-entering* mechanism from the viewpoint of DRT. It should be emphasized that it is our computational interpretation of the connective (operational rules with configurations) that makes it possible to view the operator from this new perspective.

Using the notion of reachability, the result of computation may be described as

(18) $\Gamma_0 \mapsto (\mathbf{v};book(\mathbf{v})) \Rightarrow (\mathbf{u};student(\mathbf{u});read(\mathbf{v})(\mathbf{u})) : s$.

Therefore, by Theorem 2,

(19) $\Gamma_0 \vdash (\mathbf{v};book(\mathbf{v})) \Rightarrow (\mathbf{u};student(\mathbf{u});read(\mathbf{v})(\mathbf{u})) : \exists_{\mathbf{x_1}} \ldots \exists\mathbf{x_n} s$

is provable for some $\mathbf{x_1},\ldots,\mathbf{x_n}$ in $\mathbf{Lq}$. It is easily observed from the proof of Theorem 2 (in Appendix) that what we have actually is

(20) $\Gamma_0 \vdash (\mathbf{v};book(\mathbf{v})) \Rightarrow (\mathbf{u};student(\mathbf{u});read(\mathbf{v})(\mathbf{u})) : \exists_{\mathbf{v}}\exists\mathbf{u} s$.

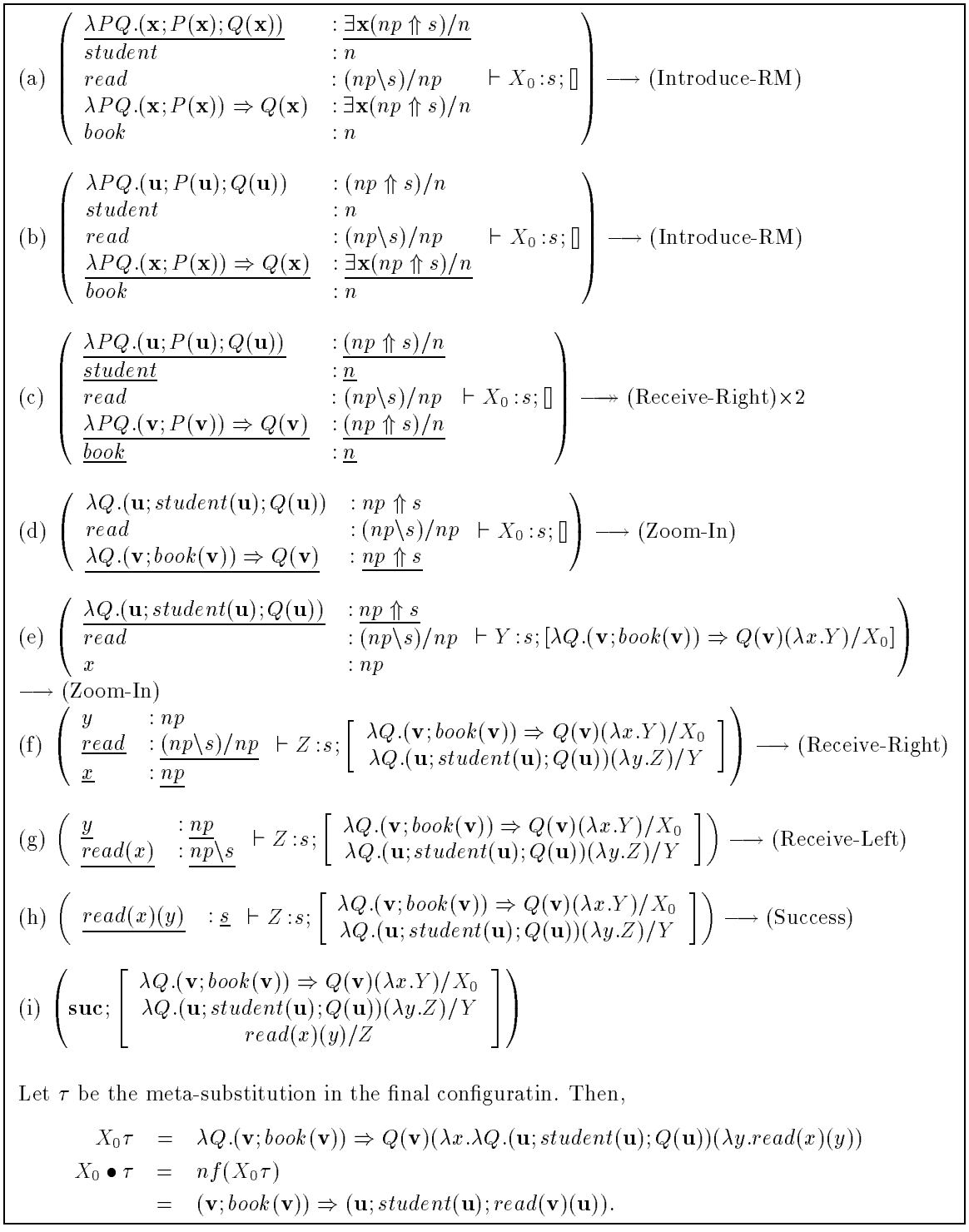
We also obtain another reading of (16)

(21) $\mathbf{u};student(\mathbf{u});(\mathbf{v};book(\mathbf{v})) \Rightarrow (read(\mathbf{v})(\mathbf{u}))$

[3] In the sequel, we only use (Zoom-In) in the extended sense of Remark 3(4) in the previous section.

a	$\mapsto$	$\lambda PQ.(\mathbf{x}; P(\mathbf{x}); Q(\mathbf{x}))$	$:\exists\mathbf{x}(np \Uparrow s)/n$
every	$\mapsto$	$\lambda PQ.(\mathbf{x}; P(\mathbf{x})) \Rightarrow Q(\mathbf{x})$	$:\exists\mathbf{x}(np \Uparrow s)/n$
no	$\mapsto$	$\lambda PQ.\neg(\mathbf{x}; P(\mathbf{x}); Q(\mathbf{x}))$	$:\exists\mathbf{x}(np \Uparrow s)/n$
the	$\mapsto$	$\lambda PQ(\mathbf{x}; \mathbf{x}\dot{=}\mathbf{y}; P(\mathbf{x}); Q(\mathbf{x}))$	$:\forall\mathbf{y}\exists\mathbf{x}(np \Uparrow s)/n$
another	$\mapsto$	$\lambda PQ(\mathbf{x}; \mathbf{x} \neq \mathbf{y}; P(\mathbf{x}); Q(\mathbf{x}))$	$:\forall\mathbf{y}\exists\mathbf{x}(np \Uparrow s)/n$
he	$\mapsto$	$\mathbf{x}$	$:\forall\mathbf{x}np$
him	$\mapsto$	$\mathbf{x}$	$:\forall\mathbf{x}np$
his	$\mapsto$	$\lambda PQ.(\mathbf{x}; poss(\mathbf{y}, \mathbf{x}); P(\mathbf{x}); Q(\mathbf{x}))$	$:\forall\mathbf{y}\exists\mathbf{x}(np \Uparrow s)/n$
Bill	$\mapsto$	$\mathbf{b}$	$:np$
who	$\mapsto$	$\lambda PQv.(Q(v); P(v))$	$:(n\backslash n)/(np\backslash s)$
who	$\mapsto$	$\lambda PQv.(Q(v); P(v))$	$:(n\backslash n)/(s/np)$
man	$\mapsto$	man	$:n$
smiles	$\mapsto$	$smiles$	$:np\backslash s$
loves	$\mapsto$	$loves$	$:(np\backslash s)/np$
doesn't	$\mapsto$	$\lambda Pv.\neg(P(v))$	$:(np\backslash s)/(np\backslash s)$
if	$\mapsto$	$\lambda pq.(p \Rightarrow q)$	$:(s/s)/s$
.	$\mapsto$	$\lambda pq.(p; q)$	$:s\backslash(txt/s)$
.	$\mapsto$	$\lambda pq.(p; q)$	$:txt\backslash(txt/s)$
.	$\mapsto$	$\lambda p.p$	$:s\backslash txt$
and	$\mapsto$	$\lambda pq.(p; q)$	$:s\backslash(s/s)$
and	$\mapsto$	$\lambda PQv.(P(v); Q(v))$	$:(s/np)\backslash((s/np)/(s/np))$
and	$\mapsto$	$\lambda PQv.(P(v); Q(v))$	$:(np\backslash s)\backslash((np\backslash s)/(np\backslash s))$
or	$\mapsto$	$\lambda pq.(p \vee q)$	$:s\backslash(s/s)$
or	$\mapsto$	$\lambda PQv.(P(v)) \vee (Q(v))$	$:(s/np)\backslash((s/np)/(s/np))$
or	$\mapsto$	$\lambda PQv.(P(v)) \vee (Q(v))$	$:(np\backslash s)\backslash((np\backslash s)/(np\backslash s))$

Figure 3: Basic Lexicon



Let τ be the meta-substitution in the final configuratin. Then,

$$\begin{aligned} X_0\tau &= \lambda Q.(\mathbf{v};book(\mathbf{v})) \Rightarrow Q(\mathbf{v})(\lambda x.\lambda Q.(\mathbf{u};student(\mathbf{u});Q(\mathbf{u}))(\lambda y.read(x)(y)) \\ X_0 \bullet \tau &= nf(X_0\tau) \\ &= (\mathbf{v};book(\mathbf{v})) \Rightarrow (\mathbf{u};student(\mathbf{u});read(\mathbf{v})(\mathbf{u})). \end{aligned}$$

Figure 4: DRS construction for (16)

by changing the order of two applications of (Zoom-In) rule in the above derivation.

We restricted our language of labels to strict DRSs, rather than proto-DRSs or proper DRSs of [vEK97]. Now we shall see the importance of this restriction through the following examples. They show that our way of combining DRT with type-logical grammar is really fruitful, not only for DRT, but also for type-logical grammar; the former provides a theory of indefinites, which has not been developed very well in the latter. Consider sentences (22) and (23).

(22) Every man walks or talks.

(23) A man walks and talks.

(22) has two readings, depending on whether the disjunction takes wide-scope over the universal quantifier or not. On the other hand, the reading of (23) according to which the conjunction takes wide-scope over the indefinite (i.e., there are two man involved, one walks and the other talks) is infelicitous, and impossible if the sentence is read with an accent on 'A'. Hence it may safely be assumed that (23) has only one reading usually.

It is difficult to explain the difference between (22) and (23) in the framework of the standard type-logical grammar, because traditionally indefinites are interpreted by means of existential quantifiers, and the treatment of 'a' and that of 'every' are much alike (cf. [Mor94]). On the other hand, our calculus clearly explains the difference between (22) and (23) because DRT provides a proper way to treat indefinites. Here the non-destructiveness condition is crucial. For (22), our **Lq** correctly computes the following two DRSs;

$$(\mathbf{u}; man(\mathbf{u})) \Rightarrow (walks(\mathbf{u}) \vee talks(\mathbf{u}))$$

$$((\mathbf{u}; man(\mathbf{u})) \Rightarrow walks(\mathbf{u})) \vee ((\mathbf{u}; man(\mathbf{u})) \Rightarrow talks(\mathbf{u}))$$

The first DRS corresponds to the reading in which 'every' takes wide scope over 'or', while the second corresponds to one in which 'or' takes wide scope over 'every'.

For (23), along the same lines, the following two might be expected;

$$\mathbf{u}; man(\mathbf{u}); walks(\mathbf{u}); talks(\mathbf{u})$$

$$\mathbf{u}; man(\mathbf{u}); walks(\mathbf{u}); \mathbf{u}; man(\mathbf{u}); talks(\mathbf{u})$$

The first one is perfectly a strict DRS. Hence we have this reading. But the second one violates the non-destructiveness condition, hence it cannot be regarded as a strict DRS, therefore we (successfully) rule out the reading in which 'and' takes wide scope over 'a'.

In order to see another importance of the non-destructiveness condition, consider the following sentence;

(24) Bill and Sue own a donkey.

(24) has two readings, one collective and the other distributive. For the present, we shall confine our attention to its distributive reading. According to Muskens[Mus96], the reading obtains the following DRS in his extended sense;

(25) $[\mathbf{u_3} | donkey\ \mathbf{u_3}; Bill\ owns\ \mathbf{u_3}]; [\mathbf{u_3} | donkey\ \mathbf{u_3}; Sue\ owns\ \mathbf{u_3}]$

which may be written in our notation as

(26) $\mathbf{u_3}; donkey(\mathbf{u_3}); owns(\mathbf{u_3})(\mathbf{b}); \mathbf{u_3}; donkey(\mathbf{u_3}); owns(\mathbf{u_3})(\mathbf{s})$.

Muskens claims that (25) is a correct DRS, which gives the right predictions about truth conditions and anaphora. However, (25) does not give the right contextual effect, since if (25) were correct the latter occurrence of $\mathbf{u_3}$, which represents the donkey Sue owns, would be accessible from the succeeding discourse, but it obviously is not the case:

(27) Bill and Sue own a donkey. It is pretty.

In (27), if we read the first sentence distributively, 'a donkey' cannot antecede 'it': Neither Bill's donkey nor Sue's donkey can be referred to by 'it' in the second sentence.

In our framework, on the other hand, there is no difficulty. (26) is not a strict DRS, because it violates the non-destructiveness condition. The distributive reading of (24) should be recovered by means of mechanisms dealing with plurals (See the next section).

Finally, let us see that our calculus embodies certain sentential grammar principles very naturally. The grammar of English does not allow a circular coindexation. Consider the following noun phrase with the indicated indices;

(28) *[her$_1$ mother]^1

It may be regarded as a special case of *i-within-i effects*. Our framework explains why the above coindexation fails very naturally; We have

her $\mapsto$ $\lambda PQ.(\mathbf{x}; P(\mathbf{y}, \mathbf{x}); Q(\mathbf{x})) : \forall \mathbf{y} \exists \mathbf{x}(np \Uparrow s)/(n/np)$
mother $\mapsto$ $mother : n/np$[4]

as the lexical entries for 'her' and 'mother'. Observe that, in every attempt to interpret (28), $\mathbf{y}$ is instantiated by the RM to which 'her' refers, while $\mathbf{x}$ is instantiated by a new RM which represents the entity the whole noun phrase denotes. Since $\forall \mathbf{y}$ precedes $\exists \mathbf{x}$, (Catch-RM) for $\mathbf{y}$ should take place before (Introduce-RM) for $\mathbf{x}$, and since the latter operational rule always introduces a fresh RM, the RM introduced cannot be identical with the RM to which $\mathbf{y}$ is resolved. Hence $\mathbf{x}$ and $\mathbf{y}$ are always taken to be distinct.

More interestingly, our **Lq** correctly predicts that the coindexations in (29) and (30) are possible, while the one in (31) is not (unless a RM which represents either 'his wife' or 'her husband' has been introduced in the context independently of sentence (31));

(29) Mary^1 loves her$_1$ husband.

(30) Her$_1$ husband loves Mary^1.

(31) *[His$_1$ wife]^2 loves [her$_2$ husband]^1.

So far, so good. But is it possible to extend this explanation into the general theory of i-within-i effects? More generally, is it possible to develop a categorial binding theory in our framework? The author has made some attempts, but they are still work in progress. (For other categorial approach to i-within-i effects, see [Jac94].)

7 Plural Anaphora: An Illustration

It is not always the case that an antecedent discourse referent of a pronoun is represented by a single lexical entry. A typical example is the case of plural anaphora. In order to make an appropriate anaphoric link between a plural pronoun and its antecedent discourse referent, we sometimes have to *create* a new RM from RMs and DRSs which have been already constructed. It is not so easy to do so in the existing frameworks of compositional DRT, such as [Zee89], [Mus94] and [vEK97], because they assume that anaphoric links should be established in advance of DRS construction. On the other hand, we do not make such assumptions, hence it is realistic to consider such a RM-creation mechanism in our framework. In this section, we shall illustrate how to extend our system to deal with plural anaphora.

[4] Since 'mother' is a relational noun, we have to assign category n/np, rather than n, to it. Correspondingly, we need to assign to the possessive pronoun a lexical entry different from that given in Figure 3.

$$\frac{\Gamma, a:!A, b:B, \Delta \vdash c:C}{\Gamma, b:B, a:!A, \Delta \vdash c:C}\ !Er \qquad \frac{\Gamma, b:B, a:!A, \Delta \vdash c:C}{\Gamma, a:!A, b:B, \Delta \vdash c:C}\ !El \qquad \frac{\Gamma, a:!A, a:!A, \Delta \vdash c:C}{\Gamma, a:!A, \Delta \vdash c:C}\ !C$$

$$\frac{\Gamma, \Delta \vdash c:C}{\Gamma, a:!A, \Delta \vdash c:C}\ !W \qquad \frac{\Gamma, a:A, \Delta \vdash c:C}{\Gamma, a:!A, \Delta \vdash c:C}\ !D \qquad \frac{!\Gamma \vdash c:C}{!\Gamma \vdash c:!C}\ !R$$

Figure 5: Inference Rules for Linear Logic Modality

(Move-Left) $(\ldots, b:B, a:!A, \ldots \vdash X:C;\tau) \longrightarrow (\ldots, a:!A, b:B, \ldots \vdash X:C;\tau)$;

(Move-Right) $(\ldots, a:!A, b:B, \ldots \vdash X:C;\tau) \longrightarrow (\ldots, b:B, a:!A, \ldots \vdash X:C;\tau)$;

(Copy) $(\ldots, a:!A, \ldots \vdash X:C;\tau) \longrightarrow (\ldots, a:!A, a:!A, \ldots \vdash X:C;\tau)$;

(Touch-Down) $(\ldots, a:!A, \ldots \vdash X:C;\tau) \longrightarrow (\ldots, a:A, \ldots \vdash X:C;\tau)$;

(Suicide) $(\ldots, a:!A, \ldots \vdash X:C;\tau) \longrightarrow (\ldots, \ldots \vdash X:C;\tau)$.

Figure 6: Operational Rules for Linear Logic Modality

In [KR93] suitable antecedent discourse referents of plural pronouns are created essentially by means of two principles, that is, *Summation* and *Abstraction*. It is not our aim to examine whether these principles are appropriate or not. Our aim is rather to show that these rules are basically available in our type-logical framework, so that it makes sense to discuss the appropriateness of these principles in the context of type-logical grammar[5] These mechanisms are represented as a sort of reactive processes, called *daemons* after the daemon-processes in Unix systems. Daemons should be reusable as many times as we like (including 0 time). To express the reusability of daemons we exploit the modality of (noncommutative) linear logic (see [Abr90], [Gir95]).

First we extend our categories so that $!A$ is also a category if A is a category in the sense of Section 4 (i.e. if A does not contain a modality). The inference rules for the modality are given in Figure 5, and the operational rules in Figure 6 (we assume that for each configuration $(\Gamma \vdash X:C;\tau)$, C should be modality-free). The cut-elimination theorem (Theorem 1) and the operational completeness theorem (Theorem 2) extend to this new calculus without any difficulty.

We extend our lexicon by adding those entries in Figure 7. Note that daemons are included in the lexicon as if they were lexical entries. Three constants $\oplus$, $\Sigma_ : (_)$ and $\in$ are newly introduced, which designate summation operator, abstraction operator and set-membership relation, respectively[6]. The intended meanings of these constants are found in [KR93].

Remark 4 For Σ operator, we make the following stipulation; expression $\Sigma\mathbf{u}:(D)$ is *legal* only if D is a *duplex condition* ([KR93], [vEK97]) of the form $Q_{\mathbf{X}}(D_1)(D_2)$ where Q is a generalized quantifier, and $\mathbf{u} \in intro(D_1) \cup intro(D_2) \cup \{\mathbf{x}\}$. Otherwise $\Sigma\mathbf{u}:(D)$ is illegal and a λpDRS containing $\Sigma\mathbf{u}:(D)$ is excluded from the set of labels $\mathbf{L}$.

[5] We say 'basically' because we ignore the intricacies in the treatment of plurals (see, e.g., [KR93]).

[6] We assume that the types of these constants are naturally understood and the definitions of pDRSs, strict DRSs $\lambda pDRSs$ and legal $\lambda pDRSs$ are suitably modified.

they	$\mapsto$	$\mathbf{x}$	$: \forall \mathbf{x} np$
and	$\mapsto$	$\lambda xyP.(\mathbf{z}; \mathbf{z} \doteq x \oplus y; P(\mathbf{z}))$	$: \exists \mathbf{x} np \backslash ((s/(np\backslash s))/np)$
Summation Daemon	$\mapsto$	$\lambda p.(\mathbf{z}; \mathbf{z} \doteq \mathbf{x} \oplus \mathbf{y}; p)$	$: !\forall \mathbf{x}\mathbf{y}\exists \mathbf{z} s/s$
Abstraction Daemon	$\mapsto$	$\lambda p.(p; \mathbf{y}; \mathbf{y} \doteq \Sigma \mathbf{x} : (p))$	$: !\forall \mathbf{x}\exists \mathbf{y} s/s$
Distribution Daemon	$\mapsto$	$\lambda xP.(\mathbf{y}; \mathbf{y} \in x) \Rightarrow P(\mathbf{y})$	$: !\exists \mathbf{y} np \backslash (np \Uparrow s)$

Figure 7: Lexicon (and Daemons) for Plurals

When $\Sigma\mathbf{u}{:}(D)$ is legal, it is (semantically) interpreted as

$$\Sigma\mathbf{u}\ \boxed{\begin{array}{|c|}\hline intro(D_1)\cup intro(D_2)\\ \hline cond(D_1)\cup cond(D_2)\\ \hline\end{array}}$$

in the framework of [KR93]. It might not be an elegant solution, but the point is that at least it works!

In order to show the use of Summation Daemon, we shall consider the following example;

(32) Mary hit John. He hit Bill. They cried.

Let us confine our attention to the reading according to which the antecedent of 'They' is the set which consists of Mary, John and Bill. In order to get a suitable RM, we need to use Summation Daemon. Hence we put Summation Daemon into the initial configuration. It is irrelevant where to put the daemon in the configuration, because it may move freely by (Move-Left) and (Move-Right). In the derivation, we make two copies of Summation Daemon, then by using (Touch-Down), (Catch-RM) and (Introduce-RM) several times, produce the following two equations;

$$\mathbf{u} = \mathbf{j} \oplus \mathbf{m}$$
$$\mathbf{v} = \mathbf{u} \oplus \mathbf{b}$$

Here, RM $\mathbf{j}$ represents John, $\mathbf{m}$ Mary, and $\mathbf{b}$ Bill. RM $\mathbf{v}$ serves as the suitable antecedent of 'They'. Therefore, we obtain the intended DRS for (32);

(33) $hit(\mathbf{j})(\mathbf{m});hit(\mathbf{b})(\mathbf{j});\mathbf{u};\mathbf{u}=\mathbf{j}\oplus\mathbf{m};\mathbf{v};\mathbf{v}=\mathbf{u}\oplus\mathbf{b};cried(\mathbf{v}):txt.$

The whole process of the DRS construction is depicted in Figure 8.

In Section 6, we have explained that the distributive reading of

(24) Bill and Sue own a donkey.

should not be represented as

(26) $\mathbf{u_3};donkey(\mathbf{u_3});owns(\mathbf{u_3})(\mathbf{b});\mathbf{u_3};donkey(\mathbf{u_3});owns(\mathbf{u_3})(\mathbf{s}).$

because it does not reflect the right contextual effect, and that our framework successfully rule out it by the non-destructiveness criterion. The proper construction for the distributive reading of (24) makes use of the non-boolean conjunction 'and' and Distribution Daemon, and yields the following DRS;

(34) $\mathbf{u};\mathbf{u}\dot{=}\mathbf{b}\oplus\mathbf{s};(\mathbf{v};\mathbf{v}\in\mathbf{u})\Rightarrow(\mathbf{w};donkey(\mathbf{w});owns(\mathbf{w})(\mathbf{v}))$

It should be observed that RM $\mathbf{w}$ is classically bound, hence that it is not accessible from the subsequent discourse. In this respect our system works very well.

There still remains a problem, however. The following sentence has a distributive reading, too;

(35) Bill likes and Sue dislikes a class.

(i.e., there are two classes involved, one Bill likes and the other Sue dislikes.) But we have no way to construct the right DRS representing the reading, because here Distribution Daemon does not do. Probably some other mechanisms are needed to handle sentences like above.

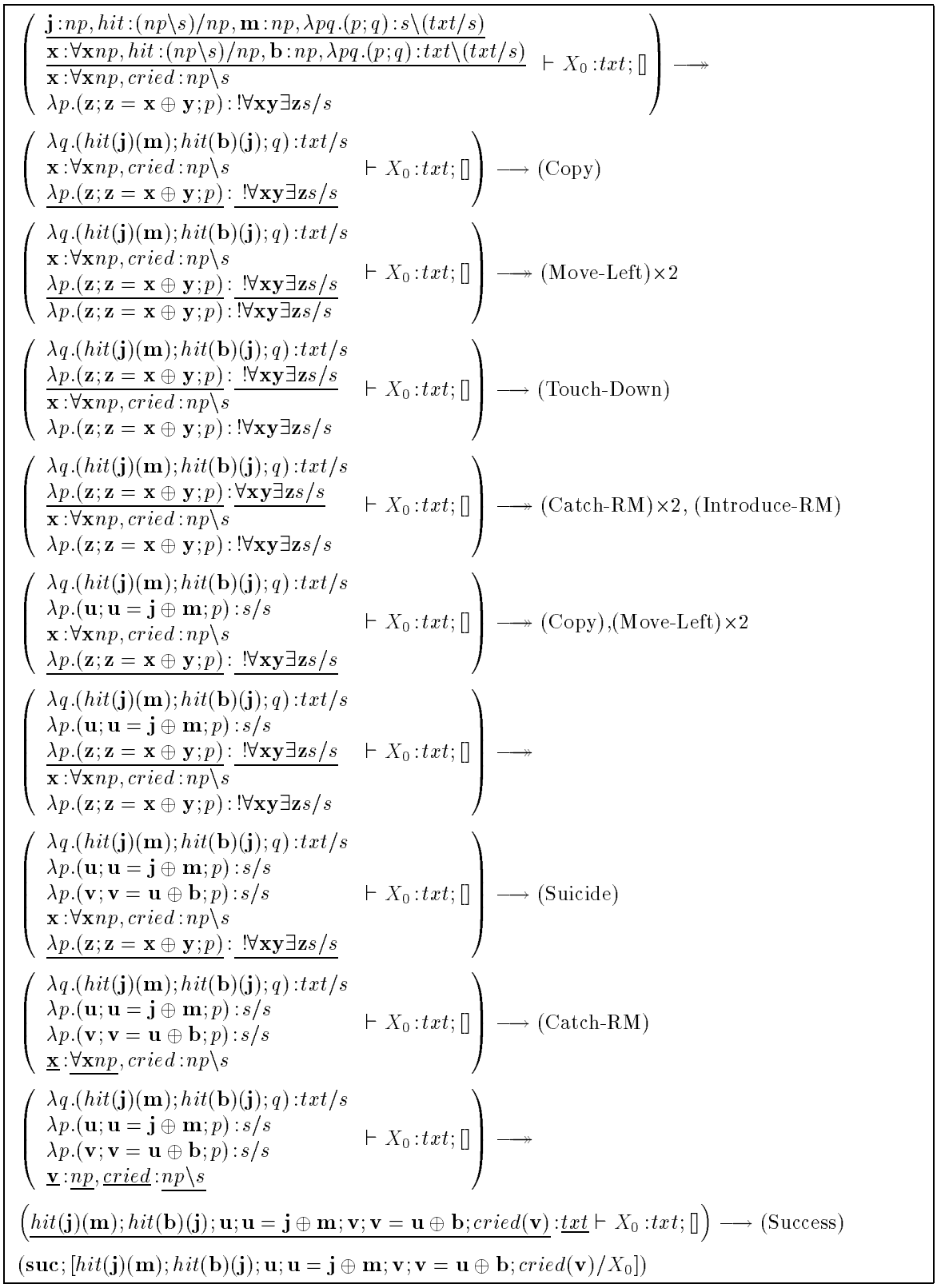


Figure 8: DRS construction for (32)

8 Conclusion and Future Work

In this paper, we have proposed a new type-logical framework of DRT, in which anaphoric linking takes place during the DRS construction. Our framework is based on sequent calculus proof search of labelled Lambek calculus with quantifier types (**Lq**). Our attempt is distinguished from the existing ones in that it does not assume that anaphoric coindexation is given in advance of DRS construction. In particular, it allows us to represent mechanisms creating new RMs such as Summation and Abstraction type-logically.

The fact that our calculus does not admit so-called *Curry-Howard correspondence* (see, e.g., [vB91]) might be regarded as a serious drawback of our proposal. The Curry-Howard correspondence reveals a deep relationship between proofs and meanings (readings) in view of categorial type logics. It seems that the practical importance of the correspondence essentially lies in the following fact: in logics having the correspondence, a proof construction procedure for those logics automatically offers a meaning composition mechanism. Although our calculus does not admit such a correspondence due to the unusual use of quantifier types, we compensated the lack of the correspondence by giving our logic a concrete computational interpretation in terms of operational semantics. The semantics allows us to explicitly describe how to compute the meaning for a given text, and Theorem 2 assures that the computation still retains a tight correspondence with the logic. Hence we claim that, at least from a practical point of view, the absence of the Curry-Howard correspondence is not a serious difficulty of our framework.

A related question that might be raised is whether our framework can be safely said to be compositional. It is surely true that the use of quantification over labels and the use of daemon processes make it less clear in what sense our framework is compositional. However, at least we can safely claim that our framework is based on a *bottom-up* mechanism which computes the whole meaning (DRS) of a text from its components' meanings by the inference rules of Lambek type logical calculus.

We leave the following problems for future work;

Process Equivalence As suggested in the introduction, each unit (in particular, a lexical entry) can be seen as a process in the sense of concurrent process calculi ([Hoa85], [Mil89]). Then a natural question arises: what is the identity of a process? In other word, what makes two processes A and B identical? It has been a central problem in the area of concurrent process calculi to establish a suitable notion of process equivalence. It is important for our work as well, because a proper equivalence notion allows us to translate a complex lexical entry into a simpler one. One might expect the mutual derivability ($A \vdash B$ and $B \vdash A$) would provide such an equivalence. However, it should be carefully examined (cf. [KY93]).

Semantics Is it possible to give a suitable (model-theoretic) semantics to **Lq**? Apparently, we could give our labels (DRSs) a semantics in terms of dynamic semantics (see, e.g., [vEK97]) could give our categories a monoid-based or relational semantics (see [vB91]), (but, of course, completeness is another problem). However, in our setting, labels and categories are mutually interacting, hence a separate treatment of labels and categories would miss the point. It would be interesting if we could describe this intricate situation by purely semantic means.

Application of scoping ($\Uparrow$) in discourse So far the use of the scoping operator $\Uparrow$ has been limited to inter-sentential uses. Since the operator provides interesting mechanisms of box-entering (Zoom-In) and box-leaving (Success) in terms of DRT, further applications may be expected. Using categories something like $A \Uparrow txt$, we can freely go into and out of various discourse boxes. Are there any interesting applications?

Exploring Daemonism further We have used several daemons, hence our standpoint may be called *Daemonism*. It should be contrasted with the *Radical Lexicalism* of usual type-logical grammars (but [Car98] uses mechanisms similar to ours to deal with plurals). Our framework is more flexible in that we are not confined to the strict function-argument structure, and still remain in the realm of logic. Daemons are quite useful, and it is expected that other useful daemons will be found ultimately.

Skolem functions representing anaphora dependency Our system is, as it stands, not very efficient due to the blind-instantiation nature of (Catch-RM). Hence it is natural to incorporate Skolemization mechanism into our setting, something like

$$\frac{a[\mathbf{u}/\mathbf{x}]:A[\mathbf{u}/\mathbf{x}],\Gamma\vdash C}{a:\forall\mathbf{x}A[\mathbf{x}],\Gamma\vdash c:C}\ \mathbf{u}:fresh$$

$$\frac{a[\mathbf{f}(\mathbf{u_1},\ldots,\mathbf{u_n})/\mathbf{x}]:A[\mathbf{f}(\mathbf{u_1},\ldots,\mathbf{u_n})/\mathbf{x}],\Gamma\vdash C}{a:\exists\mathbf{x}A[\mathbf{x}],\Gamma\vdash c:C}$$

where $\mathbf{f}$ is a new function symbol and $\mathbf{u_1},\ldots,\mathbf{u_n}$ are all RMs occurring in the lower sequent.

Beside the efficiency problem, pre-unified units themselves are of particular interest; since our quantifiers bind RMs, Skolem functions would represent constraints on RM-resolution, i.e., *anaphora dependency*. Hence, if Skolemization technique is successfully implemented, it would naturally lead us to a type-logical theory of underspecification (for the Skolemization technique in resource logics, see, e.g., [LS94]).

References

[Abr90] V. Michele Abrusci. Non-commutative intuitionistic linear propositional logic. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 36:297–318, 1990.

[AP91] Jean-Marc Andreoli and Remo Pareschi. Linear objects: Logical processes with built-in inheritance. *New Generation Computing*, 9:445–473, 1991.

[Ash93] N. Asher. *Reference to Abstract Objects in Discourse*. Kluwer, Dordrecht, 1993.

[Bar92] H. P. Barendregt. Lambda calculi with types. In S. Abramsky, Dov M. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science, Volume 2*, pages 117–309. Oxford University Press, 1992.

[Car98] Bob Carpenter. *Type-Logical Semantics*. MIT Press, 1998.

[Gab96] Dov. M. Gabbay. *Labelled Deductive Systems*. Oxford: Oxford University Press, 1996.

[Gir87] Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.

[Gir95] Jean-Yves Girard. Linear logic: Its syntax and semantics. In J.-Y. Girard, Y. Lafont, and L. Regnier, editors, *Advances in Linear Logic*, pages 1–42. Cambridge University Press, 1995. Proceedings of the Workshop on Linear Logic, Ithaca, New York, June 1993.

[Hei82] I. Heim. *The Semantics of Definite and Indefinite Noun Phrases*. PhD thesis, University of Massachusetts, Amherst, 1982.

[HM94] Joshua S. Hodas and Dale Miller. Logic programming in a fragment of intuitionistic linear logic. *Information and Computation*, 110(2):327–365, 1994. Extended abstract in the Proceedings of the Sixth Annual Symposium on Logic in Computer Science, Amsterdam, July 15–18, 1991.

[Hoa85] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, Englewood Cliffs, NJ, 1985.

[Jac94] P. Jacobson. i-within-i effects in a variable free semantics and a categorial syntax. In P. Dekker et al., editors, *Proceedings of the 9th Amsterdam Colloquium*, 1994.

[Kam81] Kamp. A theory of truth and semantic representation. In J. Groenendijk, Th. Jansen, and M. Stokhof, editors, *Formal Methods in the Study of Language*, pages 277–322. Methematisch Centrum, Amsterdam, 1981.

[KKP95] M. Kohlhase, S. Kuschert, and M. Pinkal. A type-theoretic semantics for λ-DRT. In P. Dekker et al., editors, *Proceedings of the 10th Amsterdam Colloquium*, pages 479–498, 1995.

[KR93] Hans Kamp and Uwe Reyle. *From Discourse to Logic*. Kluwer, Dordrecht, 1993.

[KY93] Naoki Kobayashi and Akinori Yonezawa. Logical, testing, and observation equivalence for processes in a linear logic programming. Preprint (presented at Linear Logic Workshop, Cornell University), 1993.

[KY95] Naoki Kobayashi and Akinori Yonezawa. Asynchronous communication model based on linear logic. *Formal Aspects of Computing*, 7:113–149, 1995. Short version appeared in Joint International Conference and Symposium on Logic Programming, Washington, DC, November 1992, Workshop on Linear Logic and Logic Programming.

[Lam58] J. Lambek. The mathematics of sentence structure. *American Mathematical Monthly*, 65(3):154–170, 1958.

[LS94] Patrick Lincoln and Natarajan Shankar. Proof search in first-order linear logic and other cut-free sequent calculi. In S. Abramsky, editor, *Ninth Annual Symposium on Logic in Computer Science*, pages 282–291, Paris, France, 1994. IEEE Computer Society Press.

[Mil89] R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.

[MNPS91] D. Miller, G. Nadathur, F. Pfenning, and A. Scedrov. Uniform proofs as a foundation for logic programming. *Anals of Pure and Applied Logic*, 51:125–157, 1991.

[Moo88] M. Moortgat. *Categorial Investigations*. Dordrecht: Foris, 1988.

[Moo91] M. Moortgat. Generalized quantification and discontinuous type constructors. Technical report, Institute for Language and Speech, University of Utrecht, 1991.

[Moo97] M. Moortgat. Categorial type logics. In Johan van Benthem and Alice ter Meulen, editors, *Handbook of Logic and Language*, chapter 2. Elsevier Science B. V., 1997.

[Mor94] Glyn Morrill. *Type Logical Grammar: Categorial Logic of Signs*. Kluwer, Dordrecht, 1994.

[Mus94] Reinhard Muskens. Categorial grammar and discourse representation theory. In *Proceedings COLING'94*, pages 508–514, 1994.

[Mus96] Reinhard Muskens. Combining montague semantics and discourse representation. *Linguistics and Philosophy*, 19:143–186, 1996.

[vB91] J. van Benthem. *Language in Action: Categories, Lambdas and Dynamic Logic*. (Studies in Logic 130). North-Holland, Amsterdam, 1991.

[vEK97] Jan van Eijck and Hans Kamp. Representing discourse in context. In Johan van Benthem and Alice ter Meulen, editors, *Handbook of Logic and Language*, chapter 3. Elsevier Science B. V., 1997.

[Zee89] H. Zeevat. A compositional approach to discourse representation theory. *Linguistics and Philosophy*, 12:95–131, 1989.

Appendix: Proof of Theorem 1 and Theorem 2

• *Proof of Theorem 1* (*Cut Elimination Theorem for* **Lq**)

For a proof π of **Lq**, let $len(\pi)$ be the number of occurrences of the inference rules in π (an axiom counts as inference). Without mentioning, we will frequently use the fact that $nf(nf(MN)[K/x]) = nf(nf(M[K/x])nf(N[K/x]))$, so $(M \bullet N)[K/\bullet x] = (M[K/\bullet x]) \bullet (N[K/\bullet x])$ if either side of the equation is defined (in **L**). First we prove several lemmas.

Lemma 3 *Let $\Gamma \vdash c : B$ be a sequent where B is quantifier-free, z be a λ variable which occurs only in quantifier-free units (i.e. units whose formula-parts are quantifier-free) in $\Gamma \vdash c : B$, and d be a label such that $c[d/\bullet z] \in \mathbf{L}$. Suppose that $\Gamma \vdash c : B$ has a cut-free proof π. Then $\Gamma[d/\bullet z] \vdash c[d/\bullet z] : B$ has a cut-free proof π' such that $len(\pi') = len(\pi)$.*

Proof. By induction on the length of the proof π.
(Case 1) π consists of an axiom *Identity*; Trivial.

(Case 2) The last inference of π is $\backslash R$ of the form;

$$\frac{x:A, \Gamma \vdash a \bullet x : B}{\Gamma \vdash a : A \backslash B} \backslash R$$

Choose a λ variable x' such that $x' \notin FV(d)$, $x' \neq z$ and x' does not occur in the lower sequent. Then, $(a \bullet x)[x'/\bullet x] \equiv a \bullet x'$ since $x \notin FV(a)$, and $a \bullet x' \in \mathbf{L}$ by Lemma 1(1), so by the induction hypothesis $x' : A, \Gamma \vdash a \bullet x' : B$ has a cut-free proof π_0 such that $len(\pi_0) = len(\pi) - 1$. Since we are assuming $a[d/z] \in \mathbf{L}$, $(a \bullet x')[d/\bullet z] \equiv (a[d/\bullet z]) \bullet x' \in \mathbf{L}$, by Lemma 1(1). Hence, again by the induction hypothesis, $x' : A, \Gamma[d/\bullet z] \vdash a[d/\bullet z] \bullet x' : B$ has a cut-free proof π_1 such that $len(\pi_1) = len(\pi) - 1$. provable. Therefore we have a cut-free proof π' of $\Gamma[d/\bullet z] \vdash a[d/\bullet z] : A \backslash B$ such that $len(\pi') = len(\pi)$.

(Case 3) The last inference of π is $\backslash L$ of the form;

$$\frac{\Gamma \vdash a : A \quad \Delta_1, b \bullet a : B, \Delta_2 \vdash c : C}{\Delta_1, \Gamma, b : A \backslash B, \Delta_2 \vdash c : C} \backslash L$$

Assume $c[d/\bullet z] \in \mathbf{L}$. Then $\Delta_1[d/\bullet z], (b \bullet a)[d/\bullet z] : B, \Delta_2[d/\bullet z] \vdash c[d/\bullet z] : C$ is cut-free provable by the induction hypothesis. This means $(b[d/\bullet z]) \bullet (a[d/\bullet z]) \equiv (b \bullet a)[d/\bullet z] \in \mathbf{L}$, and $a[d/\bullet z] \in \mathbf{L}$ by Lemma 1(3). Therefore by the induction hypothesis, $\Gamma[d/\bullet z] \vdash a[d/\bullet z] : A$ is also cut-free provable. Thus our claim holds by $\backslash L$.

(Case 4) The last inference of π is $/L$ or $/R$; Similar to the above cases.

(Case 5) The last inference of π is $\exists LR$ or $\forall L$ or $\forall R$; Immediate from our assumption that the succedent category B is quantifier-free and z occurs only in quantifier-free units.

(Case 6) The last inference of π is $\Uparrow L$ of the form;

$$\frac{\Gamma_1, x : A, \Gamma_2 \vdash a \bullet x : B \quad \Delta_1, b \bullet a : B, \Delta_2 \vdash c : C}{\Delta_1, \Gamma_1, b : A \Uparrow B, \Gamma_2, \Delta_2 \vdash c : C} \Uparrow L(*)$$

Assume $c[d/\bullet z] \in \mathbf{L}$. Then $\Delta_1[d/\bullet z], (b \bullet a)[d/\bullet z] : B, \Delta_2[d/\bullet z] \vdash c[d/\bullet z] : C$ is cut-free provable by the induction hypothesis. This means $(b[d/\bullet z]) \bullet (a[d/\bullet z]) \equiv (b \bullet a)[d/\bullet z] \in \mathbf{L}$, and $a[d/\bullet z]$ is in **L** by Lemma 1(3).

We may safely assume that $x \notin FV(d)$ and $x \neq z$ (otherwise replace x with x' satisfying the condition as in (Case 2)). Then $(a \bullet x)[d/\bullet z] \equiv a[d/\bullet z] \bullet x \in \mathbf{L}$ by Lemma 1(1). Therefore by the induction hypothesis, $\Gamma_1[d/\bullet z], x : A, \Gamma_2[d/bulletz] \vdash a[d/\bullet z] : B$ is also cut-free provable. Thus our claim holds by $\backslash L$. ■

Lemma 4 *Let $\Gamma \vdash c:B$ a sequent and and θ be a marker renaming such that $Dom(\theta) \cup Range(\theta)$ is disjoint with the set of bound marker variables occurring in $\Gamma \vdash c:B$. Suppose that $c\theta \in \mathbf{L}$. If $\Gamma \vdash c:B$ has a cut-free proof π, then $\Gamma\theta \vdash c\theta:B\theta$ has a cut-free proof π' such that $len(\pi') = len(\pi)$.*

Proof. By induction on the length of the proof π.

(Case 1) π consists of an axiom $Identity$; Trivial.

(Case 2) The last inference of π is $\backslash L$, $\backslash R$, $/L$, $/R$, or $\Uparrow L$; The proof is similar to that of Lemma 3 (this time use Lemma 2 instead of Lemma 1).

(Case 3) The last inference of π is $\forall R$ of the form;

$$\frac{\Gamma \vdash a[\mathbf{u}/\mathbf{x}]:A[\mathbf{u}/\mathbf{x}]}{\Gamma \vdash a:\forall \mathbf{x} A}\ \forall R$$

Choose a RM $\mathbf{u}'$ such that $\mathbf{u}' \notin Range(\theta) \cup Dom(\theta)$ and $\mathbf{u}'$ does not occur in the lower sequent. Since $a[\mathbf{u}/\mathbf{x}][\mathbf{u}'/\mathbf{u}] \equiv a[\mathbf{u}'/\mathbf{x}] \in \mathbf{L}$ by Lemma 2(1), we have a cut-free proof π' of $\Gamma \vdash a[\mathbf{u}'/\mathbf{x}] : A[\mathbf{u}'/\mathbf{x}]$ such that $len(\pi') = len(\pi) - 1$. Now $a[\mathbf{u}'/\mathbf{x}]\theta \equiv a\theta[\mathbf{u}'/\mathbf{x}]$ and $A[\mathbf{u}'/\mathbf{x}]\theta \equiv A\theta[\mathbf{u}'/\mathbf{x}]$ by our choice of $\mathbf{u}'$ and the assumption $\mathbf{x} \notin Dom(\theta) \cup Range(\theta)$, and $a\theta[\mathbf{u}'/\mathbf{x}] \in \mathbf{L}$ since it is a variant of $a\theta$ (by Lemma 2(1)). Hence, applying the induction hypothesis, we have a cut-free proof of $\Gamma\theta \vdash a\theta[\mathbf{u}'/\mathbf{x}]:A\theta[\mathbf{u}'/\mathbf{x}]$. Therefore our claim holds by $\forall R$.

(Case 4) The last inference of π is $\forall L$; Immediate.

(Case 5) The last inference of π is $\exists LR$; Similar to (Case 4). ■

Lemma 5 *Let π be a proof*

$$\frac{\begin{array}{cc}\pi_1 & \pi_2 \\ \vdots & \vdots \\ \Gamma \vdash a:A & \Delta_1, a:A, \Delta_2 \vdash c:C\end{array}}{\Delta_1, \Gamma, \Delta_2 \vdash c:C}\ Cut$$

where π_1 and π_2 are cut-free. Then $\Delta_1, \Gamma, \Delta_2 \vdash c:C$ has a cut-free provable $\tilde{\pi}$ such that $len(\tilde{\pi}) < len(\pi)$.

Proof. By induction on $length(\pi_1) + length(\pi_2)$.

(Case 1) π_1 consists of an axiom $a : A \vdash a : A$; In this case π_2 itself is a cut-free proof of $\Delta_1, \Gamma, \Delta_2 \vdash c:C$. Clearly $len(\pi_2) < len(\pi)$.

(Case 2) π_2 consists of an axiom $a:A \vdash a:A$; Similar to (Case 1).

In what follows we assume that $len(\pi_1) > 1$ and $len(\pi_2) > 1$.

(Case 3) $a:A$ is not the principal formula of the last inference of π_2 (i.e., the upper sequent of the inference also contains $a:A$); We describe only a few cases.
Case(3-1) The last inference of π_2 is $\backslash R$; in this case π is the form

$$\frac{\begin{array}{cc}\pi_1 & \dfrac{\begin{array}{c}\pi_2' \\ \vdots \\ x:B, \Delta_1, a:A, \Delta_2 \vdash b \bullet x:C\end{array}}{\Delta_1, a:A, \Delta_2 \vdash b:B\backslash C}\ \backslash R \\ \vdots & \\ \Gamma \vdash a:A & \end{array}}{\Delta_1, \Gamma, \Delta_2 \vdash b:B\backslash C}$$

We may assume that x does not occur in Γ (otherwise replace x with a fresh variable using Lemma 3). Construct the following proof;

$$\dfrac{\begin{array}{cc}\overset{\pi_1}{\vdots} & \overset{\pi'_2}{\vdots} \\ \Gamma \vdash a:A & x:B, \Delta_1, a:A, \Delta_2 \vdash b \bullet x:C\end{array}}{x:B, \Delta_1, \Gamma, \Delta_2 \vdash b \bullet x:C}$$

By the induction hypothesis, the lower sequent has a cut-free proof the length of which is less than $len(\pi) - 1$. Hence, $\Delta_1, \Gamma, \Delta_2 \vdash b:B\backslash C$ has a cut-free proof the length of which is less than $len(\pi)$.

(Case 3-2) The last inference of π_2 is $\forall R$; Similar to (Case 3-1), but this time use Lemma 4 instead of Lemma 3.

(Case 4) $a:A$ is not the principal formula of the last inference of π_1; Similar to (Case 3).

Henceforth we assume that $a:A$ is a principal formula of the last inferences of π_1 and π_2.

(Case 5) A is of the form $B\backslash D$; then the last part of π is of the form

$$\dfrac{\dfrac{\overset{\pi'_1}{\vdots}}{x:B, \Gamma \vdash a \bullet x:D} \quad \dfrac{\begin{array}{cc}\overset{\pi'_{21}}{\vdots} & \overset{\pi'_{22}}{\vdots} \\ \Pi \vdash b:B & \Delta_1, a \bullet b:D, \Delta_2 \vdash c:C\end{array}}{\Delta_1, \Pi, a:B\backslash D, \Delta_2 \vdash c:C}}{\Gamma \vdash a:B\backslash D} \quad \dfrac{}{\Delta_1, \Pi, \Gamma, \Delta_2 \vdash c:C}$$

Since $(a \bullet x)[b/\bullet x] \equiv a \bullet b \in \mathbf{L}$, we have a cut-free proof $\overline{\pi'_1}$ of $b:B, \Gamma \vdash a \bullet b:D$ such that $len(\overline{\pi'_1}) = len(\pi'_1)$ by Lemma 3. Hence $\Pi, \Gamma \vdash a \bullet b:D$ has proof

$$\dfrac{\begin{array}{cc}\overset{\pi'_{21}}{\vdots} & \overset{\overline{\pi'_1}}{\vdots} \\ \Pi \vdash b:B & b:B, \Gamma \vdash a \bullet b:D\end{array}}{\Pi, \Gamma \vdash a \bullet b:D}.$$

Since $len(\pi'_2) + len(\pi'_1) < len(\pi_1) + len(\pi_2)$, we may apply the induction hypothesis, and therefore we have a cut-free proof $\tilde{\pi}$ of $\Pi, \Gamma \vdash a \bullet b:D$ such that $len(\tilde{\pi}) < len(\pi'_{21} + len(\pi'_1) + 1$. Hence $\Delta_1, \Pi, \Gamma, \Delta_2 \vdash c:$has proof

$$\dfrac{\begin{array}{cc}\overset{\tilde{\pi}}{\vdots} & \overset{\pi'_{22}}{\vdots} \\ \Pi, \Gamma \vdash a \bullet b:D & \Delta_1, a \bullet b:D, \Delta_2 \vdash c:C\end{array}}{\Delta_1, \Pi, \Gamma, \Delta_2 \vdash c:C}$$

and $len(\tilde{\pi}) + len(\pi'_{22}) < len(\pi'_{21}) + len(\pi'_1) + 1 + len(\pi'_{22}) < len(\pi_1) + len(\pi_2)$, we may apply the induction hypothesis and we have a cut-free proof of $\Delta_1, \Pi, \Gamma, \Delta_2 \vdash c:$the length of which is less than $len(\tilde{\pi}) + len(\pi'_{22}) < len(\pi)$.

(Case 6) A is of the form $\exists \mathbf{x} B$; then the last part of π is of the form

$$\dfrac{\dfrac{\begin{array}{cc}\overset{\pi'_1}{\vdots} & \overset{\pi'_1}{\vdots} \\ a[\mathbf{u}/\mathbf{x}]:C[\mathbf{u}/\mathbf{x}], \Gamma \vdash b[\mathbf{u}/\mathbf{x}]:B[\mathbf{u}/\mathbf{x}] & b[\mathbf{v}/\mathbf{x}]:B[\mathbf{v}/\mathbf{x}], \Delta \vdash b[\mathbf{v}/\mathbf{x}]:D[\mathbf{v}/\mathbf{x}]\end{array}}{a:\exists \mathbf{x} C, \Gamma \vdash b:\exists \mathbf{x} B \quad b:\exists \mathbf{x} B, \Delta \vdash c:\exists \mathbf{x} D}}{a:\exists \mathbf{x} C, \Gamma, \Delta \vdash c:\exists \mathbf{x} D}$$

Choose a RM $\mathbf{w}$ which does not occur freely in the lowest sequent. Since $b[\mathbf{w}/\mathbf{x}]$ is a variant of b, $b[\mathbf{u}/\mathbf{x}][\mathbf{w}/\mathbf{u}] \equiv b[\mathbf{w}/\mathbf{x}] \in \mathbf{L}$ by Lemma 2(1). Therefore, $a[\mathbf{w}/\mathbf{x}]:C[\mathbf{w}/\mathbf{x}], \Gamma \vdash b[\mathbf{w}/\mathbf{x}]:B[\mathbf{w}/\mathbf{x}]$ has a cut-free proof of the same length by Lemma 4. Similarly, $b[\mathbf{w}/\mathbf{x}]:B[\mathbf{w}/\mathbf{x}], \Delta \vdash b[\mathbf{w}/\mathbf{x}]:D[\mathbf{w}/\mathbf{x}]$ has a cut-free proof of the same length. Hence we have

$$\frac{a[\mathbf{w}/\mathbf{x}]:C[\mathbf{w}/\mathbf{x}], \Gamma \vdash b[\mathbf{w}/\mathbf{x}]:B[\mathbf{w}/\mathbf{x}] \quad b[\mathbf{w}/\mathbf{x}]:B[\mathbf{w}/\mathbf{x}], \Delta \vdash b[\mathbf{w}/\mathbf{x}]:D[\mathbf{w}/\mathbf{x}]}{a[\mathbf{w}/\mathbf{x}]:C[\mathbf{w}/\mathbf{x}], \Gamma, \Delta \vdash c[\mathbf{w}/\mathbf{x}]:D[\mathbf{w}/\mathbf{x}]}$$

and by the induction hypothesis, the lower sequent has a cut-free proof the length of which is less than $len(\pi'_1) + len(\pi'_2) + 1 = len(\pi) - 1$. Therefore, $a:\exists \mathbf{x} C, \Gamma, \Delta \vdash c:\exists \mathbf{x} D$ has a cut-free proof the length of which is less than $len(\pi)$.

(Case 7) A is of the form $\forall \mathbf{x} B$; then the last part of π is of the form;

$$\dfrac{\dfrac{\begin{array}{c}\pi'_1\\ \vdots\\ \Gamma \vdash a[\mathbf{u}/\mathbf{x}]:B[\mathbf{u}/\mathbf{x}]\end{array}}{\Gamma \vdash a:\forall \mathbf{x} B} \quad \dfrac{\begin{array}{c}\pi'_2\\ \vdots\\ \Delta_1, a[\mathbf{t}/\mathbf{x}]:B[\mathbf{t}/\mathbf{x}], \Delta_2 \vdash c:C\end{array}}{\Delta_1, a:\forall \mathbf{x} B, \Delta_2 \vdash c:C}}{\Delta_1, \Gamma, \Delta_2 \vdash c:C}$$

Since $a[\mathbf{t}/\mathbf{x}] \in \mathbf{L}$, $\Gamma \vdash a[\mathbf{t}/\mathbf{x}]:B[\mathbf{t}/\mathbf{x}]$ also has a cut-free proof of the same length by Lemma 4. Hence we have

$$\dfrac{\begin{array}{c}\pi'_1\\ \vdots\\ \Gamma \vdash a[\mathbf{t}/\mathbf{x}]:B[\mathbf{t}/\mathbf{x}]\end{array} \quad \begin{array}{c}\pi'_2\\ \vdots\\ \Delta_1, a[\mathbf{t}/\mathbf{x}]:B[\mathbf{t}/\mathbf{x}], \Delta_2 \vdash c:C\end{array}}{\Delta_1, \Gamma, \Delta_2 \vdash c:C}$$

and by the induction hypothesis, our claim holds.

(Case 8) A is of the form $B \Uparrow D$; then the last part of π is of the form

$$\dfrac{\dfrac{\begin{array}{c}\pi'_1\\ \vdots\\ \Gamma \vdash a:B\end{array}}{\Gamma \vdash \lambda x \bullet xa:B \Uparrow D} \quad \dfrac{\begin{array}{c}\pi'_2\\ \vdots\\ \Pi_1, x:B, \Pi_2 \vdash b \bullet x:D\end{array} \quad \begin{array}{c}\pi''_2\\ \vdots\\ \Delta_1, (\lambda x \bullet xa) \bullet b:D, \Delta_2 \vdash c:C\end{array}}{\Delta_1, \Pi_1, \lambda x \bullet xa:B \Uparrow D, \Pi_2, \Delta_2 \vdash c:C}}{\Delta_1, \Pi_1, \Gamma, \Pi_2, \Delta_2 \vdash c:C}$$

The proof is similar to (Case 5). See the following proof figure;

$$\dfrac{\dfrac{\begin{array}{c}\pi''_1\\ \vdots\\ \Gamma \vdash a:B\end{array} \quad \begin{array}{c}\pi'_2\\ \vdots\\ \Pi_1, a:B, \Pi_2 \vdash b \bullet a:D\end{array}}{\Pi_1, \Gamma, \Pi_2 \vdash b \bullet a:D} \quad \begin{array}{c}\pi''_2\\ \vdots\\ \Delta_1, (\lambda x \bullet xa) \bullet b:D, \Delta_2 \vdash c:C\end{array}}{\Delta_1, \Pi_1, \Gamma, \Pi_2, \Delta_2 \vdash c:C}$$

(Note that $(\lambda x \bullet xa) \bullet b \equiv b \bullet a$.) ■

Theorem 1 (Cut Elimination Theorem for Lq) *If S is provable in* **Lq**, *then S has a cut-free proof in* **Lq**.

Proof. By induction on the number of cuts occurring in the proof of S. Use the previous lemma. ■

• *Proof of Theorem 2* (*Operational Completeness Theorem*)

Lemma 6 *Suppose that* $\Gamma \vdash c : C$ *is cut-free provable where* C *does not contain a universal quantifier* $\forall$*. If* Γ *contains a free occurrence of RM* $\mathbf{u}$*, then* $c{:}C$ *also contains a free occurrence of* $\mathbf{u}$*.*

Proof. It is easily shown by induction on the length of the proof. ■

An (instance of) inference rule $\exists LR$ is said to be *canonical* if it is of the form;

$$\frac{\Gamma_1, a[\mathbf{u}/\mathbf{x}]{:}A[\mathbf{u}/\mathbf{x}], \Gamma_2 \vdash b{:}B}{\Gamma_1, a{:}\exists \mathbf{x} A, \Gamma_2 \vdash b{:}\exists \mathbf{u} B}\ \exists LR(**),$$

i.e., the label of the succedent unit does not change after the application of the inference.

Lemma 7 *If* $\Gamma \vdash c{:}C$ *has a cut-free proof, then it also has a cut-free proof in which all instances of* $\exists LR$ *are canonical.*

Proof. By induction on the length of the proof. We only consider the case when the last inference is $\exists LR$;

$$\frac{\Gamma_1, a[\mathbf{u}/\mathbf{x}]{:}A[\mathbf{u}/\mathbf{x}], \Gamma_2 \vdash b[\mathbf{u}/\mathbf{y}]{:}B[\mathbf{u}/\mathbf{y}]}{\Gamma_1, a{:}\exists \mathbf{x} A, \Gamma_2 \vdash b{:}\exists \mathbf{y} B}\ \exists LR(**)$$

Since $b[\mathbf{u}/\mathbf{y}]$ does not contain $\mathbf{y}$, $\Gamma_1, a[\mathbf{u}/\mathbf{x}] : A[\mathbf{u}/\mathbf{x}], \Gamma_2$ does not contain $\mathbf{y}$ by Lemma 6. Since $b[\mathbf{u}/\mathbf{y}][\mathbf{y}/\mathbf{u}] = b \in \mathbf{L}$, $\Gamma_1, a[\mathbf{y}/\mathbf{x}] : A[\mathbf{y}/\mathbf{x}], \Gamma_2 \vdash b : B$ has a proof of the same length with that of $\Gamma_1, a[\mathbf{u}/\mathbf{x}] : A[\mathbf{u}/\mathbf{x}], \Gamma_2 \vdash b[\mathbf{u}/\mathbf{y}] : B[\mathbf{u}/\mathbf{y}]$ by Lemma 4 (Note that $a[\mathbf{u}/\mathbf{x}][\mathbf{y}/\mathbf{u}] = a[\mathbf{y}/\mathbf{x}]$). By the induction hypothesis the sequent has a proof in which all instances of $\exists LR$ are canonical. Moreover,

$$\frac{\Gamma_1, a[\mathbf{y}/\mathbf{x}]{:}A[\mathbf{y}/\mathbf{x}], \Gamma_2 \vdash b{:}B}{\Gamma_1, a{:}\exists \mathbf{x} A, \Gamma_2 \vdash b{:}\exists \mathbf{y} B}$$

is canonical. Therefore our claim holds. ■

Theorem 2 (Completeness of Operational Semantics for Lq) *Let* A *be a quantifier-free category. Then* $\Gamma \mapsto a : A$ *if and only if* $\Gamma \vdash a : \exists \mathbf{x_1} \ldots \exists \mathbf{x_n} A$ *is provable in* **Lq** *for some marker variables* $\mathbf{x_1} \ldots \mathbf{x_n}$*.*

Proof. ($\Rightarrow$) Let $R \in \{\longrightarrow, \longrightarrow\!\!\!\!\!\rightarrow\}$. We prove the following four statements by induction on the generation of $\Phi_1 R \Phi_2$. Then statement (b) proves the theorem in one direction.

(a) Suppose that $\Phi_1 \equiv (\Gamma_1 \vdash X{:}C_1; \tau_1)$ and $\Phi_2 \equiv (\mathbf{suc}; \tau_2)$ and $X \not\equiv X_0$. Then $\Gamma_1 \vdash X \bullet \tau_2 : C_1$ is provable.

(b) Suppose that $\Phi_1 \equiv (\Gamma_1 \vdash X_0 : C_1; \tau_1)$ and $\Phi_2 \equiv (\mathbf{suc}; \tau_2)$. Then $\Gamma_1 \vdash X_0 \bullet \tau_2 : \exists \mathbf{x_1} \ldots \exists \mathbf{x_n} C_1$ is provable for some $\mathbf{x_1} \ldots \mathbf{x_n}$ $(n \geq 0)$.

(c) Suppose that $\Phi_1 \equiv (\Gamma_1 \vdash X{:}C_1; \tau_1)$, $\Phi_2 \equiv (\Gamma_2 \vdash Y{:}C_2; \tau_2)$ and $Y \not\equiv X_0$. If $\Gamma_2 \vdash Y \bullet (\tau_2 \circ \tau') : C_2$ is provable for a meta-substitution τ', then $\Gamma_1 \vdash X \bullet (\tau_2 \circ \tau') : \exists \mathbf{x_1} \ldots \mathbf{x_n} C_1$ is provable for some $\mathbf{x_1} \ldots \mathbf{x_n}$ $(n \geq 0)$. In particular, if $X \not\equiv X_0$, then we can take $n = 0$.

(d) Suppose that $\Phi_1 \equiv (\Gamma_1 \vdash X_0 : C_1; \tau_1)$ and $\Phi_2 \equiv (\Gamma_2 \vdash X_0 : C_2; \tau_2)$. If $\Gamma_2 \vdash X_0 \bullet (\tau_2 \circ \tau') : \exists \mathbf{y_1} \ldots \exists \mathbf{y_m} C_2$ is provable for a meta-substitution τ' and marker variables $\mathbf{y_1} \ldots \mathbf{y_m}$. Then $\Gamma_1 \vdash X_0 \bullet (\tau_2 \circ \tau') : \exists \mathbf{x_1} \ldots \mathbf{x_n} \exists \mathbf{y_1} \ldots \exists \mathbf{y_m} C_1$ is provable for some $\mathbf{x_1} \ldots \mathbf{x_n}$ $(n \geq 0)$.

We describe several cases.
(Case 1) Success

$$\overline{(a : A \vdash X : A; \tau) \longrightarrow (\mathbf{suc}; \tau \circ [a/X])}$$

(case (a) or (b)) Since $X \bullet (\tau \circ [a/X]) = a$ and $a\!:\!A \vdash a\!:\!A$ is an axiom, our claim holds.

(Case 2) Introduce-RM

$$\frac{\mathbf{u} \text{ does not occur freely in } (\Delta_1, a\!:\!\exists \mathbf{x} A, \Delta_2 \vdash X_0\!:\!C; \tau)}{(\Delta_1, a\!:\!\exists \mathbf{x} A, \Delta_2 \vdash X_0\!:\!C; \tau) \longrightarrow (\Delta_1, a[\mathbf{u/x}]\!:\!A[\mathbf{u/x}], \Delta_2 \vdash X_0\!:\!C; \tau)}$$

(case (d)) Suppose that $\Delta_1, a[\mathbf{u/x}]\!:\!A[\mathbf{u/x}], \Delta_2 \vdash b\!:\!\exists \mathbf{y_1} \ldots \mathbf{y_m} C$ is provable, where $b = X_0 \bullet (\tau \circ \tau')$. Then,

$$\frac{\Delta_1, a[\mathbf{u/x}]\!:\!A[\mathbf{u/x}], \Delta_2 \vdash b\!:\!\exists \mathbf{y_1} \ldots \mathbf{y_m} C}{\Delta_1, a\!:\!\exists \mathbf{x} A, \Delta_2 \vdash b\!:\!\exists \mathbf{u} \exists \mathbf{y_1} \ldots \mathbf{y_m} C} .$$

So our claim holds.

(Case 3) Hypothesize-Left

$$\frac{}{(\Delta \vdash X\!:\!A \backslash B; \tau) \longrightarrow (x\!:\!A, \Delta \vdash Y\!:\!B; \tau \circ [\lambda x.Y/X])}$$

where $Y \not\equiv X_0$. (case (c)) Suppose that $x\!:\!A, \Delta \vdash b\!:\!B$ is provable, where $b = Y \bullet (\tau \circ [\lambda x.Y/X] \circ \tau')$. We can easily show that b contains a free occurrence of x, so $\lambda x \bullet b$ is defined by Lemma 1(2). Hence,

$$\frac{x\!:\!A, \Delta \vdash b\!:\!B}{\Delta \vdash \lambda x \bullet b\!:\!A \backslash B}$$

(Note that $b = (\lambda x \bullet b) \bullet x$.) Since $X \bullet (\tau \circ [\lambda x.Y/X] \circ \tau') = (\lambda x.Y) \bullet \tau') = \lambda x \bullet b$, our claim holds.

(Case 4) Receive-Left

$$\frac{(\Gamma \vdash Y\!:\!A; []) \longrightarrow\!\!\!\!\!\rightarrow (\mathbf{suc}; \tau') \quad a = Y \bullet \tau' \quad b \bullet a \text{ is defined} \quad Y \not\equiv X_0}{(\Delta_1, \Gamma, b\!:\!A \backslash B, \Delta_2 \vdash X\!:\!C; \tau) \longrightarrow (\Delta_1, b \bullet a\!:\!B, \Delta_2 \vdash X\!:\!C; \tau)}$$

(case (c) or (d)) By the induction hypothesis $\Gamma \vdash a : A$ is provable. So if $\Delta_1, b \bullet a : B, \Delta_2 \vdash X \bullet (\tau \circ \tau')\!:\!\exists \mathbf{y_1} \ldots \mathbf{y_m} C$ $(m \geq 0)$ is provable, $\Delta_1, \Gamma, b\!:\!A \backslash B, \Delta_2 \vdash X \bullet (\tau \circ \tau')\!:\!\exists \mathbf{y_1} \ldots \mathbf{y_m} C$ is also provable.

(Case 5) Transitivity; We only consider the following case (other cases are easier);

$$\frac{(\Gamma_1 \vdash X_0\!:\!C_1; \tau_1) \longrightarrow\!\!\!\!\!\rightarrow (\Gamma_2 \vdash X_0\!:\!C_2; \tau_2) \quad (\Gamma_2 \vdash X_0\!:\!C_2; \tau_2) \longrightarrow\!\!\!\!\!\rightarrow (\Gamma_3 \vdash X_0\!:\!C_3; \tau_3)}{(\Gamma_1 \vdash X_0\!:\!C_1; \tau_1) \longrightarrow\!\!\!\!\!\rightarrow (\Gamma_3 \vdash X_0\!:\!C_3; \tau_3)}$$

(case (d)) Suppose that $\Gamma_3 \vdash X_0 \bullet (\tau_3 \circ \tau')\!:\!\exists \mathbf{y_1} \ldots \mathbf{y_m} C_3$ is provable. Then, by the induction hypothesis $\Gamma_2 \vdash X_0 \bullet (\tau_3 \circ \tau')\!:\!\exists \mathbf{z_1} \ldots \mathbf{z_k} \mathbf{y_1} \ldots \mathbf{y_m} C_2$ is provable for some $\mathbf{z_1} \ldots \mathbf{z_k}$. It is easy to show that $\tau_3 = \tau_2 \circ \tau''$ for some τ'', hence $\tau_3 \circ \tau' = \tau_2 \circ \tau'' \circ \tau'$. So, again by the induction hypothesis, $\Gamma_1 \vdash X_0 \bullet (\tau_3 \circ \tau')\!:\!\exists \mathbf{x_1} \ldots \mathbf{x_n} \mathbf{z_1} \ldots \mathbf{z_k} \mathbf{y_1} \ldots \mathbf{y_m} C_1$ is provable for some $\mathbf{x_1} \ldots \mathbf{x_n}$.

($\Leftarrow$) Suppose that $\Gamma \vdash a\!:\!\exists \mathbf{x_1} \ldots \exists \mathbf{x_n} A$ is provable in **Lq** for some $\mathbf{x_1} \ldots \mathbf{x_n}$. Then, by Theorem 1, it has a cut-free proof, and by Lemma 7 it has a cut-free proof π in which all instances of $\exists LR$ are canonical. We can easily show that $\Gamma \mapsto a\!:\!A$ by induction on $len(\pi)$. ■