Relationships between Specified and Underspecified Quantification by the Theory of Acyclic Recursion

Roussanka Loukanova
Department of Mathematics, Stockholm University, Sweden
rloukanova@gmail.com

KEYWORD

Recursion;
Semantics;
Underspecification

ABSTRACT

The paper introduces a technique for representing quantifier relations that can have different scope order depending on context. The technique is demonstrated by classes of terms denoting relations, where each of the arguments of a relation term is bound by a different quantifier. We represent a formalization of linking quantifiers with the corresponding argument slots that they bind, across lambda-abstractions and reduction steps. The purpose of the technique is to represent underspecified order of quantification, in the absence of a context and corresponding information about the order. Furthermore, it is used to represent subclasses of larger classes of relations depending on order of quantification or specific relations.

1. Introduction

The formal theory of the technique introduced in the paper is a generalization of the theories of recursion introduced by (Moschovakis, 1997; Moschovakis, 1989). The formal languages and their respective calculi include terms constructed by adding a recursion operator along with the typical λ-abstraction and application. The resulting theories serve as a powerful, computational formalization of the abstract notion of algorithm with full recursion, which, while operating over untyped functions and other entities, can lead to calculations without termination. The untyped languages of recursion were then extended to a higher-order theory of acyclic recursion L^{λ}_{ar}, see (Moschovakis, 2006), which is more expressive, by adding typed, functional objects. In another aspect, L^{λ}_{ar} is limited to computations that always close-off, by allowing only acyclic terms. I.e., the class of languages L^{λ}_{ar}, and their corresponding calculi, represent abstract, functional operations (algorithms) that terminate after finite number of computational steps. Such limitation is useful in many, if not most, practical applications. In particular, algorithmic semantics of human language can be among such applications, for which the simply-typed theory of acyclic recursion L^{λ}_{ar} was introduced in (Moschovakis, 2006).

In this paper, we use an extended formal language and theory of L^{λ}_{ar}, with respective calculi, that gives better possibilities for representation of underspecified scope distribution of higher-order quantifiers. Firstly, we use the extended reduction calculus of L^{λ}_{ar} introduced in (Loukanova, 2017), which employs an additional reduction rule, γ-rule, see (Loukanova, 2017). Secondly, we use restrictions over L^{λ}_{ar}-terms introduced in (Loukanova, 2013). This paper provides also a more general technique than in (Loukanova, 2013). Here we represent a formalization of linking quantifiers with the corresponding argument slots that they bind, across λ-abstractions and reduction steps. In addition, the technique presented here is applicable for any abstract, i.e., mathematical, n-ary argument-binding relations, $n \geq 2$, while we illustrate it with human language quantifiers.

Detailed introduction to the formal language L^{λ}_{ar} of Moschovakis acyclic recursion, its syntax, denotational and algorithmic semantics, and its theory, is given in (Moschovakis, 2006) and (Loukanova, 2017). The formal system L^{λ}_{ar} is a higher-order type theory, which is a proper extension of Gallin's TY_2, see (Gallin, 1975), and via

that, of Montague Intensional Logic (IL). Furthermore, L^{λ}_{ar} covers in much more adequately semantics of human language than the semantic representations in Montague's "The proper treatment of quantification in ordinary English" (PTQ). About Montague IL and PTQ, see (Montague, 1973; Montague, 1988; Thomason, 1974).

2. The Type Theory L^{λ}_{ar}

In this paper, we give a brief, informal introduction of L^{λ}_{ar}. For details, see (Moschovakis, 2006) and (Loukanova, 2017).

Notationally, we use the symbol $:\equiv$ in several ways. Firstly, we use it in the typical rules, given in Backus-Naur form (BNF), for the definitions of the expressions of the formal language L^{λ}_{ar}, as in (*Types*) and (4)–(5). By these notational variants of the formal definitions, we follow the tradition of a widely-spread notation in computer science. Then, we expand this usage of the symbol $:\equiv$, for convenience, as the syntactic operator of replacement of sub-expressions, e.g., variables and constants, with other expressions, e.g., in (18).

We use the symbol $\equiv$ for syntactic identity between expressions, and in syntactic definitions of abbreviations, e.g., as in (7) and (9b)–(9f). I.e., the symbols $:\equiv$ and $\equiv$ are meta-symbols, which are not in the vocabulary of the language L^{λ}_{ar}, and its extensions. On the other hand, the symbol $:=$ is an essential symbol in formal expressions, i.e., it occurs in some of the L^{λ}_{ar}-terms, essentially in the *recursion terms* of the form (5).

2.1 Syntax of L^{λ}_{ar}

Types of L^{λ}_{ar}: The set Types is the smallest set defined recursively

$$\tau :\equiv e \mid t \mid s \mid (\tau_1 \to \tau_2) \qquad (\textit{Types})$$

The vocabulary of L^{λ}_{ar} consists of pairwise disjoint sets of:

Typed Constants $K = \bigcup_{\tau \in \mathsf{Types}} K_\tau$; where, for each $\tau \in \mathsf{Types}$, K_τ is a denumerable set of constants:

$$K_\tau = \{\mathsf{c}_0, \mathsf{c}_1, \ldots, \mathsf{c}_{k_\tau}, \ldots\} \tag{1}$$

Note that one may choose a version of L^{λ}_{ar} having all of the sets K_τ to be finite.

Typed Variables L^{λ}_{ar} has two kinds of typed variables:

Pure Variables $\mathsf{PureVars} = \bigcup_{\tau \in \mathsf{Types}} \mathsf{PureVars}_\tau$, where for each $\tau \in \mathsf{Types}$, $\mathsf{PureVars}_\tau$ and $\mathsf{RecVars}_\tau$ are denumerable sets:

$$\mathsf{PureVars}_\tau = \{\mathsf{v}_0, \mathsf{v}_1, \ldots\} \tag{2}$$

Recursion Variables called also *locations*, $\mathsf{RecVars} = \bigcup_{\tau \in \mathsf{Types}} \mathsf{RecVars}_\tau$, where for each $\tau \in \mathsf{Types}$:

$$\mathsf{RecVars}_\tau = \{\mathsf{p}_0, \mathsf{p}_1, \ldots\} \tag{3}$$

The Terms of L^{λ}_{ar}: The language L^{λ}_{ar} extends the terms of the typical λ-calculi, by using the facility of the recursion (i.e., location) variables and a new operator for term construction. In addition to application $A(B)$ and λ-abstraction terms, the formal language L^{λ}_{ar} has specialized *recursion terms*. The recursion terms of L^{λ}_{ar} are formed by using the designated recursion operator, denoted by the constant where in infix notation. Here, by (5), we express the recursive rules for generating the set of L^{λ}_{ar}-terms by using a notational variant of extended, typed Backus-Naur form (TBNF), with the assumed types given as superscripts.

Definition 1. The set Terms consists of the expressions generated by the following rules:

$$A :\equiv \mathsf{c}^{\tau} : \tau \mid x^{\tau} : \tau \mid B^{(\sigma\to\tau)}(C^{\sigma}) : \tau \mid \lambda(v^{\sigma})(B^{\tau}) : (\sigma \to \tau) \tag{4}$$

$$\mid A_0^{\sigma} \text{ where } \{p_1^{\sigma_1} := A_1^{\sigma_1}, \ldots, p_n^{\sigma_n} := A_n^{\sigma_n}\} : \sigma \tag{5}$$

where $A_1 : \sigma_1, \ldots, A_n : \sigma_n$ are in Terms, and $p_1 : \sigma_1, \ldots, p_n : \sigma_n$ ($n \geq 0$), are pairwise different recursion variables, of matching types and such that $\{p_1^{\sigma_1} := A_1^{\sigma_1}, \ldots, p_n^{\sigma_n} := A_n^{\sigma_n}\}$ is a sequence of assignments that satisfies the following Acyclicity Constraint (AC):

Acyclicity Constraint (AC): For any terms $A_1 : \sigma_1, \ldots, A_n : \sigma_n$, and pairwise different recursion variables $p_1 : \sigma_1, \ldots, p_n : \sigma_n$ ($n \geq 0$), the sequence $\{p_1 := A_1, \ldots, p_n := A_n\}$ is *acyclic* iff there is a function $\mathsf{rank} : \{p_1, \ldots, p_n\} \longrightarrow \mathbb{N}$ such that, for all $p_i, p_j \in \{p_1, \ldots, p_n\}$,

$$\text{if } p_j \text{ occurs freely in } A_i \text{ then } \mathsf{rank}(p_j) < \mathsf{rank}(p_i).$$

By induction on term structure, it follows that the type assignment to the terms $A \in \mathsf{Terms}$ is unambiguous. Notationally, it can be skipped when it is understood or irrelevant. In addition, depending on convenience and clarity, we shall use two notational variants of type assignment, either by a superscript, A^{τ}, or after a colon sign, $A : \tau$, to express that A is a term of type τ.

The terms of the form (5) are called *recursion terms*. The terms of the form (4), without the distinction between pure and recursion (location) variables, provide the syntax of typical simply-typed λ-calculi. The language L_{ar}^{λ} and its calculi, including the Reduction Calculi, is a theory of typed recursion, which essentially extends λ-calculi to typed theory of recursion, i.e., theory of algorithms. That is achieved via adding the distinction between the two kinds of variables, and the new term constructs of the form (5). The recursion terms A_0^{σ} where $\{p_1^{\sigma_1} := A_1^{\sigma_1}, \ldots, p_n^{\sigma_n} := A_n^{\sigma_n}\}$ are formed by using the symbol where. Actually, where is an operator constant (like the typical logic operators for conjunction, disjunction, negation) of the language L_{ar}^{λ}, and its extension L_r^{λ} to full recursion, but where designates recursion calculations.

We shall skip the type assignments to the terms, e.g., as in (6b)–(6c), when this is appropriate and the types are clear. Sometimes, to increase readability, we use extra brackets, as in (6d).

$$A_0^{\sigma} \text{ where } \{p_1^{\sigma_1} := A_1^{\sigma_1}, \ldots, p_n^{\sigma_n} := A_n^{\sigma_n}\} : \sigma \tag{6a}$$

$$A_0^{\sigma} \text{ where } \{p_1^{\sigma_1} := A_1^{\sigma_1}, \ldots, p_n^{\sigma_n} := A_n^{\sigma_n}\} \tag{6b}$$

$$A_0 \text{ where } \{p_1 := A_1, \ldots, p_n := A_n\} \tag{6c}$$

$$[A_0^{\sigma} \text{ where } \{p_1^{\sigma_1} := A_1^{\sigma_1}, \ldots, p_n^{\sigma_n} := A_n^{\sigma_n}\}] : \sigma \tag{6d}$$

We often use the following notation for abbreviated sequences of mutually recursive assignments:

$$\overrightarrow{p} := \overrightarrow{A} \equiv p_1 := A_1, \ \ldots, \ p_n := A_n \quad (n \geq 0) \tag{7}$$

2.2 Two Kinds of Semantics of L_{ar}^{λ}

Denotational Semantics of L_{ar}^{λ}. The definition of the denotations of the terms follows the structure of the L_{ar}^{λ}-terms, in a compositional way. Intuitively, the denotation $\mathsf{den}(A)$ of a term A is computed algorithmically, by computing the denotations $\mathsf{den}(A_i)$ of the parts A_i and saving them in the corresponding recursion variable (i.e., location) p_i, step-by-step, according to recursive ranking $\mathsf{rank}(p_i)$.

The reduction calculi of L^{λ}_{ar} effectively reduces each term A to its canonical form $\mathsf{cf}(A)$: $A \Rightarrow_{\mathsf{cf}} \mathsf{cf}(A)$, which in general, is a recursion term:

$$\mathsf{cf}(A) \equiv A_0 \text{ where } \{p_1 := A_1, \ldots, p_n := A_n\} \qquad (n \geq 0) \tag{8}$$

For each A, its canonical form $\mathsf{cf}(A)$ is unique up to renaming bound variables and reordering the recursive assignments $\{p_1 := A_1, \ldots, p_n := A_n\}$. The order of the recursive assignments is unessential since the order of the algorithmic steps in computations of the denotations are determined by the $\mathsf{rank}(A_i)$, for $i = 1, \ldots, n$.

Algorithmic Semantics of L^{λ}_{ar}. The reduction calculi and the canonical forms of the terms play an essential role in the algorithmic semantics of L^{λ}_{ar}. The algorithm for computing the denotation $\mathsf{den}(A)$ of a meaningful L^{λ}_{ar}-term A, is determined by its canonical form. E.g., the sentence (9a) can be rendered into the L^{λ}_{ar}-term A, (9b), which then, by a sequence of reduction steps (not included here, for sake of space, and marked by $\Rightarrow \ldots$), is reduced to its canonical form $\mathsf{cf}(A)$, (9c).

$$\text{John likes Mary's father.} \tag{9a}$$

$$\xrightarrow{\text{render}} A \equiv [\mathit{like}\big(\mathit{father\_of}(\mathit{mary})\big)](\mathit{john}) \tag{9b}$$

(note: (9b) is an *explicit term*)

$$\equiv \mathit{like}\big(\mathit{father\_of}(\mathit{mary}), \mathit{john}\big) \ : \ \widetilde{\mathsf{t}} \tag{9c}$$

(note: (9c) is an *explicit term* in relational notation)

$$\Rightarrow \ldots \text{(by applying reduction rules of } L^{\lambda}_{ar}\text{)}$$

$$\Rightarrow_{\mathsf{cf}} \mathit{like}(f)(j) \text{ where } \{j := \mathit{john},\ m := \mathit{mary},\ f := \mathit{father\_of}(m)\} \tag{9d}$$

(note: (9d) is *a recursion term*)

$$\equiv \mathit{like}(f, j) \text{ where } \{j := \mathit{john},\ m := \mathit{mary},\ f := \mathit{father\_of}(m)\} \tag{9e}$$

(note: (9e) is a *recursion term* in relational notation)

$$\equiv \mathsf{cf}(A) \tag{9f}$$

There is a rank function for the term (9c), which satisfies the acyclicity condition. For each such rank function, $\mathsf{rank}(m) < \mathsf{rank}(f)$, since m occurs in the term-part $\mathit{father\_of}(m)$ of the assignments $f := \mathit{father\_of}(m)$. E.g., $\mathsf{rank}(j) = 0$, $\mathsf{rank}(m) = 1$, and $\mathsf{rank}(f) = 2$. And, the term, which is in canonical form, determines the algorithm for computing A:

Step 1. Compute:

$$\mathsf{den}(j) = \mathsf{den}(\mathit{john}) \tag{10}$$

Step 2. Compute:

$$\mathsf{den}(m) = \mathsf{den}(\mathit{mary}) \tag{11}$$

Step 3. Compute:

$$\mathsf{den}(f) = \mathsf{den}[\mathit{father\_of}(m)] \tag{12}$$

Step 4. Compute:

$$\mathsf{den}(A) = \mathsf{den}\big[[\mathit{like}(f)](j)\big] = \mathsf{den}\big[[\mathit{like}(\mathsf{den}(f))]\big](\mathsf{den}(j)) \tag{13}$$

3. Reduction Calculus of the Theory of Acyclic Recursion

For the reductions of terms to their canonical forms that are used in this paper, we need the extended γ-reduction, which uses the γ-rule introduced in (Loukanova, 2017). While the detailed reduction steps of the terms A to their canonical and γ-canonical forms are part of the computational attire, we do include only some of them here, where we find it good for the topic under consideration. We shall skip many of the sequential reduction steps for sake of space and long chains of computations. They are not essential for understanding the technique of underspecified semantic representation introduced in the paper. In our future work, we foresee development of a computerised systems for the reduction of the terms into their canonical and γ-canonical forms. However, the rules of the Reduction Calculus of L^{λ}_{ar} and its extension with the γ-rule are essential for the topic of this paper. For better understanding of the paper, we include the rules here.

About detailed formal coverage of the γ-calculus, see (Loukanova, 2017). The η-rule and η-reduction calculus, introduced in (Loukanova, 2011a) is more simple, while a weaker special case of the $gamma$-rule and γ-reduction. The η-rule could have been used for the reductions to canonical forms in some of the examples given in this paper, but for the cost of longer intermediate terms in the reduction sequences, and by choosing a special order of applications of the reduction rules.

In this section, we give the formal definitions of the reduction rules of the extended γ-reduction calculus of L^{λ}_{ar}, which includes the additional γ-rule. About more details and the Referential γ-Synonymy Theorem 3, see (Loukanova, 2017).

Reduction Rules.

Congruence: If $A \equiv_{\mathsf{c}} B$, then $A \Rightarrow B$ (cong)

Transitivity: If $A \Rightarrow B$ and $B \Rightarrow C$, then $A \Rightarrow C$ (trans)

Compositionality:

If $A \Rightarrow A'$ and $B \Rightarrow B'$, then $A(B) \Rightarrow A'(B')$ (comp-ap)

If $A \Rightarrow B$, then $\lambda(u)(A) \Rightarrow \lambda(u)(B)$ (comp-λ)

If $A_i \Rightarrow B_i$, for $i = 0, \ldots, n$, then (comp-rec)

A_0 where $\{\, p_1 := A_1, \ldots, p_n := A_n \,\} \Rightarrow B_0$ where $\{\, p_1 := B_1, \ldots, p_n := B_n \,\}$

Head rule:

$$\begin{aligned} &\big(A_0 \text{ where } \{\overrightarrow{p} := \overrightarrow{A}\}\big) \text{ where } \{\overrightarrow{q} := \overrightarrow{B}\} \\ &\Rightarrow A_0 \text{ where } \{\overrightarrow{p} := \overrightarrow{A},\ \overrightarrow{q} := \overrightarrow{B}\} \end{aligned} \tag{head}$$

given that no p_i occurs free in any B_j, for $i = 1, \ldots, n$, $j = 1, \ldots, m$.

Bekič-Scott rule:

$$\begin{aligned} &A_0 \text{ where } \{p := \big(B_0 \text{ where } \{\overrightarrow{q} := \overrightarrow{B}\}\big), \overrightarrow{p} := \overrightarrow{A}\} \\ &\Rightarrow A_0 \text{ where } \{p := B_0, \overrightarrow{q} := \overrightarrow{B},\ \overrightarrow{p} := \overrightarrow{A}\} \end{aligned} \tag{B-S}$$

given that no q_i occurs free in any A_j, for $i = 1, \ldots, n$, $j = 1, \ldots, m$.

Rrecursion-application rule:

$$(A_0 \text{ where } \{\overrightarrow{p} := \overrightarrow{A}\})(B) \Rightarrow A_0(B) \text{ where } \{\overrightarrow{p} := \overrightarrow{A}\} \tag{recap}$$

given that no p_i occurs free in B for $i = 1, \ldots, n$.

Application rule:

$$A(B) \Rightarrow A(p) \text{ where } \{p := B\} \tag{ap}$$

given that B is a proper term and p is a fresh location

λ-rule:

$$\begin{array}{l} \lambda(u)(A_0 \text{ where } \{\, p_1 := A_1, \ldots, p_n := A_n \,\}) \\ \Rightarrow \lambda(u)A'_0 \text{ where } \{\, p'_1 := \lambda(u)A'_1, \ldots, p'_n := \lambda(u)A'_n \,\} \end{array} \tag{λ}$$

where for all $i = 1, \ldots, n$, p'_i is a fresh location and A'_i is the result of the replacement of the free occurrences of $p_1, \ldots, p_n$ in A_i with $p'_1(u), \ldots, p'_n(u)$, respectively, i.e.:

$$\begin{array}{l} A'_i \equiv \; A_i\{p_1 :\equiv p'_1(u), \ldots, p_n :\equiv p'_n(u)\} \\ \quad \text{for all } i \in \{\, 1, \ldots, n \,\} \end{array} \tag{18}$$

γ-rule

$$\begin{array}{rl} A \equiv & A_0 \text{ where } \{\, \overrightarrow{a} := \overrightarrow{A},\; p := \lambda(v)P,\; \overrightarrow{b} := \overrightarrow{B} \,\} \\ \Rightarrow^*_\gamma & A'_0 \text{ where } \{\, \overrightarrow{a} := \overrightarrow{A'}, p' := P, \qquad \overrightarrow{b} := \overrightarrow{B'} \,\} \end{array} \tag{γ}$$

where

(a) The term $A \in \mathsf{Terms}$ satisfies the γ-condition (in Definition 2) for the assignment $p := \lambda(v)P$.

(b) $p' \in \mathsf{RecVars}_\tau$ is a fresh recursion variable.

(c) $\overrightarrow{A'} \equiv \overrightarrow{A}\{p(v) :\equiv p'\}$ is the result of the replacements $A_i\{p(v) :\equiv p'\}$ of all occurrences of $p(v)$ by p', in all parts A_i in $\overrightarrow{A}$ ($i \in \{0, \ldots, n\}$).

(d) $\overrightarrow{B'} \equiv \overrightarrow{B}\{p(v) :\equiv p'\}$ is the result of the replacements $B_j\{p(v) :\equiv p'\}$ of all occurrences of $p(v)$ by p', in all parts B_j in $\overrightarrow{B}$ ($j \in \{1, \ldots, k\}$).

Definition 2 (γ-condition)**.** We say that a term $A \in \mathsf{Terms}$ *satisfies the γ-condition for an assignment* $p := \lambda(v)P$ if and only if A is of the form:

$$A \equiv A_0 \text{ where } \{\, \overrightarrow{a} := \overrightarrow{A},\; p := \lambda(v)P,\; \overrightarrow{b} := \overrightarrow{B} \,\} \tag{20}$$

for some $v \in \mathsf{PureVars}_\sigma$, $p \in \mathsf{RecVars}_{(\sigma\to\tau)}$, $\lambda(v)P \in \mathsf{Terms}_{(\sigma\to\tau)}$, $\overrightarrow{A} \equiv A_1, \ldots, A_n \in \mathsf{Terms}$, $\overrightarrow{a} \equiv a_1, \ldots, a_n \in \mathsf{RecVars}$ ($n \geq 0$), $\overrightarrow{B} \equiv B_1, \ldots, B_k \in \mathsf{Terms}$, $\overrightarrow{b} \equiv b_1, \ldots, b_k \in \mathsf{RecVars}$ ($k \geq 0$), of corresponding types and such that the following clauses (1) and (2) hold:

1. The term $P \in \mathsf{Terms}_\tau$ does not have any (free) occurrences of v (and of p, by the acyclicity) in it, i.e., $v \notin \mathsf{FreeV}(P)$.

2. All the occurrences of p in A_0, $\overrightarrow{A}$ and $\overrightarrow{B}$ are occurrences in a sub-term $p(v)$ that are in the scope of $\lambda(v)$ (modulo congruence by renaming the scope variable v for each scope $\lambda(v)$).

In such a case, we also say that *the assignment $p := \lambda(v)P$ satisfies the γ-condition in the recursion term A* (20).

The *reduction relation* is the smallest relation, denoted by $\Rightarrow$, between terms that is closed under the reduction rules, without using the γ-rule. The *γ-reduction relation* is the smallest relation, denoted by $\Rightarrow^*_\gamma$, or simply by $\Rightarrow_\gamma$, between terms that is closed under the reduction rules, including the γ-rule.

Definition 3 (Term Irreducibility)**.** We say that a term $A \in \mathsf{Terms}$ is *irreducible* if and only if

$$\text{for all } B \in \mathsf{Terms}, \text{ if } A \Rightarrow B, \text{ then } A \equiv_{\mathsf{c}} B \tag{21}$$

The following theorems are major results that are essential for algorithmic semantics.

Theorem 1 (Canonical Form Theorem: existence and uniqueness of the canonical forms)**.** *(Moschovakis, 2006) For each term A, there is a unique, up to congruence, irreducible term C, denoted by $\mathsf{cf}(A)$ and called the canonical form of A, such that:*

1. $\mathsf{cf}(A) \equiv A_0$ where $\{\, p_1 := A_1, \ldots, p_n := A_n \,\}$,
   for some explicit, irreducible terms $A_1, \ldots, A_n$ ($n \geq 0$)

2. $A \Rightarrow \mathsf{cf}(A)$

3. *if $A \Rightarrow B$ and B is irreducible, then $B \equiv_c \mathsf{cf}(A)$, i.e., $\mathsf{cf}(A)$ is the unique, up to congruence, irreducible term to which A can be reduced.*

Similarly, for each L^λ_{ar}-term A, there is unique up to congruence term in γ-canonical form.

Theorem 2 (Simplified γ-Canonical Form Theorem)**.** *(Loukanova, 2017) For every $A \in \mathsf{Terms}$, there is a unique, up to congruence, γ-irreducible term C, denoted by $\mathsf{cf}_\gamma(A)$ and called the γ-canonical form of A, such that:*

1. $\mathsf{cf}_\gamma(A) \equiv A_0$ where $\{\, p_1 := A_1, \ldots, p_n := A_n \,\}$,
   for some explicit, irreducible $A_0, \ldots, A_n \in \mathsf{Terms}$ ($n \geq 0$)

2. $A \Rightarrow^*_\gamma \mathsf{cf}_\gamma(A)$

3. *if $A \Rightarrow^*_\gamma B$ and B is γ-irreducible, then $B \equiv_c \mathsf{cf}_\gamma(A)$, i.e., $\mathsf{cf}_\gamma(A)$ is unique, up to congruence, γ-irreducible term. We write*

$$A \Rightarrow_{\mathsf{gcf}} B \iff B \equiv_c \mathsf{cf}_\gamma(A) \tag{22a}$$

$$A \Rightarrow_{\mathsf{gcf}} \mathsf{cf}_\gamma(A) \tag{22b}$$

Theorem 3 (Referential γ-Synonymy Theorem)**.** *Two terms A, B are referentially γ-synonymous, $A \approx_\gamma B$, if and only if there are explicit, irreducible terms of corresponding types, $A_i : \sigma_i$, $B_i : \sigma_i$ ($i = 0, \ldots, n$), ($n \geq 0$), such that:*

$$A \Rightarrow_{\mathsf{gcf}} A_0 \text{ where } \{\, p_1 := A_1, \ldots, p_n := A_n \,\} \quad \textit{(γ-irreducible)} \tag{23a}$$

$$B \Rightarrow_{\mathsf{gcf}} B_0 \text{ where } \{\, p_1 := B_1, \ldots, p_n := B_n \,\} \quad \textit{(γ-irreducible)} \tag{23b}$$

and for all $i = 0, \ldots, n$,

$$\mathsf{den}(A_i)(g) = \mathsf{den}(B_i)(g), \quad \textit{for all } g \in G. \tag{24}$$

About more details on the L^λ_{ar} theories and their reduction systems, see (Loukanova, 2017).

4. Distributions of Multiple Quantifiers

4.1 Specific Instances of Quantifier Distributions

We use the relation $\xrightarrow{\mathsf{render}}$ between expressions of a human language (here, English) and formal terms, i.e., L^λ_{ar}-terms, to designate that the formal term is a corresponding semantic representation. E.g., if E is an expression of a human language, and T a L^λ_{ar}-term, we write $\mathsf{E} \xrightarrow{\mathsf{render}} T$, in case T is a semantic representation of E. In this paper, we are not concerned about how the rendering can be obtained. For a possibility of defining the rendering relation, via syntax-semantics interface in computational grammar, see (Loukanova, 2011a).

We represent the general problem with a sentence like (25) that represents a specific instance of a general problem. E.g., the sentence (25) is an instance of a whole class of human language sentences that have a head verb with syntactic arguments, which can be noun phrases interpreted as semantic quantifiers. In human language, such verbs are common, while verbs with more syntactic arguments are relatively limited. A verb similar to "give" denotes a relation with three semantic arguments. Each of these arguments can be filled up by a different quantifier. Furthermore, in general, each of the syntactic complements of the head verb in a sentence may have components that are also quantifiers, and thus, contribute to the combinatorial possibilities of scope distributions. In this paper, we do not consider such additional quantifiers, since that is not in its subject. We focus on quantifiers contributed directly by the major arguments of the head relation and their scope distributions. I.e., we limit our consideration to quantifiers that bind variables filling the argument slots of the relation denoted by the head verb. For a demonstration of the technique, we take the sentence (25).

$$S \equiv \text{Every professor gives some student two papers.} \tag{25}$$

$$S \xrightarrow{\mathsf{render}} T_1 \tag{26}$$

As in typical λ-calculi, one of the semantic interpretations of the sentence S can be represented by the L^λ_{ar}-term T_1 in (27a)–(27d). In a given, specific context, the speaker may intend an interpretation of the sentence S represented by the closed, i.e., *fully specified*, L^λ_{ar}-term T_1 with the scope distribution (27a)–(27d). This and other distributions of these quantifiers can be represented as well, and by other L^λ_{ar}-terms, as we shall see in the following sections.

$$T_1 \equiv every(professor) \tag{27a}$$

$$\big[_3 \lambda(x_3) some(student) \tag{27b}$$

$$\big[_1 \lambda(x_1) two(paper) \tag{27c}$$

$$\big[_2\lambda(x_2)give(x_1)(x_2)(x_3)]_2\big]_1\big]_3 \tag{27d}$$

In rendering the sentence S to a L^{λ}_{ar}-term, we render the verb "give" to the constant $give : (\widetilde{\mathsf{e}} \to (\widetilde{\mathsf{e}} \to (\widetilde{\mathsf{e}} \to \widetilde{\mathsf{t}})))$, which is the functional, i.e., currying, coding for the corresponding 3-argument relation. Note that we have taken the order of the applications of the constant $give$ to be, respectively, at first to the term for recipient, then for the present that is given, and at last for the presenter.

In this work, we use the typical for mathematics indexing $1, \ldots, n$, for the argument slots and the variables filling them, for n-argument curryed functional symbols. In what follows, this indexing is convenient tool for distinguishing argument slots and linking hem via λ-abstraction. Technically, the names of the variables $x_1, \ldots, x_n$ are irrelevant, as soon as they are set upon a conventional agreement. Here, we use notational names of the variables, so that x_i fills the i-th argument slot, $i = 1, \ldots, n$.

By using the reduction rules, we reduce the term T_1 to its canonical and γ-canonical forms (by suppressing the detailed, long, sequence of intermediate reductions). Note that, in the reductions and formulas, we use superscripts not only to distinguish variables, but also as counters of applications of (λ) and (γ) rules. The term (28h)–(28l) is obtained by three applications of the (γ) rule, once for $s^1 := \lambda(x_3)student$, and two times for $b^2 := \lambda(x_3)\lambda(x_1)paper$.

$$T_1 \Rightarrow \ldots \tag{28a}$$
$$\Rightarrow_{\mathsf{cf}} every(p)(R_3) \text{ where } \{ \tag{28b}$$
$$R_3 := \lambda(x_3)some(s^1(x_3))(R^1_1(x_3)), \tag{28c}$$
$$R^1_1 := \lambda(x_3)\lambda(x_1)two(b^2(x_3)(x_1))(R^2_2(x_3)(x_1)), \tag{28d}$$
$$R^2_2 := \lambda(x_3)\lambda(x_1)\lambda(x_2)give(x_1)(x_2)(x_3), \tag{28e}$$
$$b^2 := \lambda(x_3)\lambda(x_1)paper,\ s^1 := \lambda(x_3)student, \tag{28f}$$
$$p := professor\ \} \tag{28g}$$
$$\Rightarrow_{\gamma}{}^3\ every(p)(R_3) \text{ where } \{ \tag{28h}$$
$$R_3 := \lambda(x_3)some(s)(R^1_1(x_3)), \tag{28i}$$
$$R^1_1 := \lambda(x_3)\lambda(x_1)two(b)(R^2_2(x_3)(x_1)), \tag{28j}$$
$$R^2_2 := \lambda(x_3)\lambda(x_1)\lambda(x_2)give(x_1)(x_2)(x_3), \tag{28k}$$
$$b := paper,\ s := student,\ p := professor\ \} \tag{28l}$$
by applying 3 times (γ)

Instead of carrying out all reductions into one long sequence of reductions between the L^{λ}_{ar}-terms[1] (28a) and (28b)–(28f), we demonstrate the reduction process by working at first on reducing the sub-terms. The term (29a), which is the sub-term (27c)–(27d) of the term (27a)–(27d), is reduced to a canonical form as follows:

$$\big[_1\lambda(x_1)two(paper)[_2\lambda(x_2)give(x_1)(x_2)(x_3)]_2\big]_1 \tag{29a}$$
$$\Rightarrow\lambda(x_1)\Big[two(paper)(R_2) \text{ where } \{\ R_2 := \lambda(x_2)give(x_1)(x_2)(x_3)\ \}\Big] \tag{29b}$$
by (ap), (comp-λ)
$$\Rightarrow\lambda(x_1)\Big[two(p)(R_2) \text{ where } \{\ R_2 := \lambda(x_2)give(x_1)(x_2)(x_3), \tag{29c}$$

[1] Note that (28b)–(28f) is a single term, as well as (28h)–(28l).

$$b := \mathit{paper}\,\}\Big] \tag{29d}$$

by (ap), (comp-ap), (comp-rec) (head), (comp-λ)

$$\Rightarrow_{\mathsf{cf}} \lambda(x_1)\mathit{two}(b^1(x_1))(R_2^1(x_1)) \text{ where } \{ \tag{29e}$$

$$R_2^1 := \lambda(x_1)\lambda(x_2)\mathit{give}(x_1)(x_2)(x_3), \tag{29f}$$

$$b^1 := \lambda(x_1)\mathit{paper}\,\} \tag{29g}$$

by (λ)

$$\Rightarrow_{\gamma} \lambda(x_1)\mathit{two}(b)(R_2^1(x_1)) \text{ where } \{ \tag{29h}$$

$$R_2^1 := \lambda(x_1)\lambda(x_2)\mathit{give}(x_1)(x_2)(x_3), \tag{29i}$$

$$b := \mathit{paper}\,\} \tag{29j}$$

by (γ)

Furthermore, we have the following reductions, for the term (30a), which is the sub-term (27b)–(27d) of the term (27a)–(27d), is reduced to a canonical form as follows:

$$\big[_3\lambda(x_3)\Big[_{x_3} \mathit{some}(\mathit{student}) \big[_1\lambda(x_1)\mathit{two}(\mathit{paper})[_2\lambda(x_2)\mathit{give}(x_1)(x_2)(x_3)]_2\big]_1\Big]_{x_3}\big]_3 \tag{30a}$$

$$\Rightarrow \lambda(x_3)\Big[_{x_3} \mathit{some}(s)(R_1) \text{ where } \{ \tag{30b}$$

$$R_1 := \big[_1\lambda(x_1)\mathit{two}(\mathit{paper})[_2\lambda(x_2)\mathit{give}(x_1)(x_2)(x_3)]_2\big]_1, \tag{30c}$$

$$s := \mathit{student}\,\}\Big]_{x_3} \tag{30d}$$

by (ap), (comp-ap), (rec-ap), (ap), (head), (comp-λ)

$$\Rightarrow \lambda(x_3)\Big[_{x_3} \mathit{some}(s)(R_1) \text{ where } \{ \tag{30e}$$

$$R_1 := \big[_1\lambda(x_1)\mathit{two}(b^1(x_1))(R_2^1(x_1)) \text{ where } \{ \tag{30f}$$

$$R_2^1 := \lambda(x_1)\lambda(x_2)\mathit{give}(x_1)(x_2)(x_3), \tag{30g}$$

$$b^1 := \lambda(x_1)\mathit{paper}\,\}\big]_1, \tag{30h}$$

$$s := \mathit{student}\,\}\Big]_{x_3} \tag{30i}$$

by (29e)-(29g), (comp-rec), (comp-λ)

$$\Rightarrow \lambda(x_3)\Big[_{x_3} \mathit{some}(s)(R_1) \text{ where } \{ \tag{30j}$$

$$R_1 := \lambda(x_1)\mathit{two}(b^1(x_1))(R_2^1(x_1)), \tag{30k}$$

$$R_2^1 := \lambda(x_1)\lambda(x_2)\mathit{give}(x_1)(x_2)(x_3), \tag{30l}$$

$$b^1 := \lambda(x_1)\mathit{paper},\ s := \mathit{student}\,\}\Big]_{x_3} \tag{30m}$$

by (B-S), (comp-λ)

$$\Rightarrow \lambda(x_3)\mathit{some}(s^1(x_3))(R_1^1(x_3)) \text{ where } \{ \tag{30n}$$

$$R_1^1 := \lambda(x_3)\lambda(x_1)two(b^2(x_3)(x_1))(R_2^2(x_3)(x_1)), \tag{30o}$$
$$R_2^2 := \lambda(x_3)\lambda(x_1)\lambda(x_2)give(x_1)(x_2)(x_3), \tag{30p}$$
$$b^2 := \lambda(x_3)\lambda(x_1)paper,\ s^1 := \lambda(x_3)student\,\} \tag{30q}$$

by (λ)

Note that, in the reductions and the formulas, we use superscripts not only to distinguish variables, but also as counters of applications of the (λ) and (γ) rules. E.g., we use superscripts of fresh recursion variables ξ^k as counters of the number k of the applications of the λ-rule, by which these recursion variables are introduced as replacement of an initial variable ξ. Thus, R_2^2 in (30o)–(30p), is the result of two applications of the λ-rule. The first one replaces $R_2 :\equiv R_2^1(x_1)$ and, respectively $R_2 :\equiv R_2^1$ in the corresponding assignments, from (29c) to (29e)–(29f). Then, the second application of the λ-rule results in, $R_2^1 :\equiv R_2^2(x_3)$ from (30k) to (30o); and, respectively $R_2^1 :\equiv R_2^2$, from the assignment (30l) to its new form (30p). Now, the condensed reduction of L^{λ}_{ar}-term T_1 to its canonical form can be expanded by (31a)–(31z).

$$T_1 \Rightarrow every(professor)\Big[_3\lambda(x_3)some(s^1(x_3))(R_1^1(x_3))\ \text{where}\ \{ \tag{31a}$$
$$R_1^1 := \lambda(x_3)\lambda(x_1)two(b^2(x_3)(x_1))(R_2^2(x_3)(x_1)), \tag{31b}$$
$$R_2^2 := \lambda(x_3)\lambda(x_1)\lambda(x_2)give(x_1)(x_2)(x_3), \tag{31c}$$
$$b^1 := \lambda(x_3)\lambda(x_1)paper,\ s^1 := \lambda(x_3)student\,\}\Big]_3 \tag{31d}$$

by (27a)–(27d), (30l)-(30o), (comp-ap)

$$\Rightarrow [every(p)\ \text{where}\ \{\,p := professor\,\}] \tag{31e}$$
$$\Big[_3\lambda(x_3)some(s^1(x_3))(R_1^1(x_3))\ \text{where}\ \{ \tag{31f}$$
$$R_1^1 := \lambda(x_3)\lambda(x_1)two(b^2(x_3)(x_1))(R_2^2(x_3)(x_1)), \tag{31g}$$
$$R_2^2 := \lambda(x_3)\lambda(x_1)\lambda(x_2)give(x_1)(x_2)(x_3), \tag{31h}$$
$$b^2 := \lambda(x_3)\lambda(x_1)paper,\ s^1 := \lambda(x_3)student\,\}\Big]_3 \tag{31i}$$

by (ap), (comp-ap)

$$\Rightarrow every(p)(R_3)\ \text{where}\ \{ \tag{31j}$$
$$R_3 := \Big[_3\lambda(x_3)some(s^1(x_3))(R_1^1(x_3))\ \text{where}\ \{ \tag{31k}$$
$$R_1^1 := \lambda(x_3)\lambda(x_1)two(b^2(x_3)(x_1))(R_2^2(x_3)(x_1)), \tag{31l}$$
$$R_2^2 := \lambda(x_3)\lambda(x_1)\lambda(x_2)give(x_1)(x_2)(x_3), \tag{31m}$$
$$b^2 := \lambda(x_3)\lambda(x_1)paper,\ s^1 := \lambda(x_3)student\,\}\Big]_3, \tag{31n}$$
$$p := professor\,\} \tag{31o}$$

by ..., (rec-ap), (ap), (comp-rec), (head)

$$\Rightarrow_{\mathsf{cf}} every(p)(R_3)\ \text{where}\ \{ \tag{31p}$$
$$R_3 := \lambda(x_3)some(s^1(x_3))(R_1^1(x_3)), \tag{31q}$$
$$R_1^1 := \lambda(x_3)\lambda(x_1)two(b^2(x_3)(x_1))(R_2^2(x_3)(x_1)), \tag{31r}$$
$$R_2^2 := \lambda(x_3)\lambda(x_1)\lambda(x_2)give(x_1)(x_2)(x_3), \tag{31s}$$

$$b^2 := \lambda(x_3)\lambda(x_1)paper,\ s^1 := \lambda(x_3)student, \tag{31t}$$

$$p := professor\ \} \tag{31u}$$

by (B-S)

$$\Rightarrow_\gamma{}^3\ every(p)(R_3)\ \text{where}\ \{ \tag{31v}$$

$$R_3 := \lambda(x_3)some(s)(R^1_1(x_3)), \tag{31w}$$

$$R^1_1 := \lambda(x_3)\lambda(x_1)two(b)(R^2_2(x_3)(x_1)), \tag{31x}$$

$$R^2_2 := \lambda(x_3)\lambda(x_1)\lambda(x_2)give(x_1)(x_2)(x_3), \tag{31y}$$

$$b := paper,\ s := student,\ p := professor\ \} \tag{31z}$$

by 3 times (γ)

The term (31v)–(31z) is obtained by three applications of the (γ) rule, once for $s^1 := \lambda(x_3)student$, and two times for $b^2 := \lambda(x_3)\lambda(x_1)paper$. Note that (28h) –(28l), i.e., respectively, (31v) –(31z), is in canonical form, but it is $\mathsf{cf}_\gamma(T_1)$. While the L^λ_{ar}-term (28b) –(28g), i.e., respectively, (31p) –(31u), is $\mathsf{cf}(T_1)$.

We can apply the (γ) rule earlier in the reduction steps, as in (29h)–(29j), which simplifies the subsequent reduction steps. In the above reductions, we have chosen to apply the (γ) rule at the end, after the entire reduction of T_1, simply to demonstrate both the simplification utility of the (γ) rule and that, in general, it does not preserve the original canonical forms of L^λ_{ar}-terms. Without the application of the (γ) rule, e.g., as in (31p)–(31u), the term $\mathsf{cf}(T_1)$ represents extra algorithmic steps to be performed in the calculation of $\mathsf{den}(T_1)$, which are vacuous functional applications, e.g., in $b^2(x_3)(x_1)$ of the function denoted by b^2, which is $\lambda(x_3)\lambda(x_1)paper$ via the recursion assignment $b^2 := \lambda(x_3)\lambda(x_1)paper$.

The term $\mathsf{cf}_\gamma(T_1)$ in (28h)–(28l) is in γ-canonical form, which is also simply canonical. It has the same denotation $\mathsf{den}(T_1)$, and represents simpler calculations steps, which are the same as in $\mathsf{cf}(T_1)$, except avoiding the vacuous functional applications.

Similarly to the specified L^λ_{ar}-term T_1, (27a)–(27d), depending on context, the sentence S can be rendered to T_2, (32b)–(33e), with a different distribution of quantification.

$$S \xrightarrow{\mathsf{render}} T_2 \tag{32a}$$

$$T_2 \equiv some\ \ (student) \tag{32b}$$

$$\big[_1\lambda(x_1)every(professor) \tag{32c}$$

$$\big[_3\lambda(x_3)two(paper) \tag{32d}$$

$$\big[_2\lambda(x_2)give(x_1)(x_2)(x_3)\big]_2\big]_3\big]_1 \tag{32e}$$

$$\Rightarrow_{\mathsf{gcf}}\ some(s)(R_1)\ \text{where}\ \{ \tag{33a}$$

$$R_1 := \lambda(x_1)every(p)(R^1_3(x_1)), \tag{33b}$$

$$R^1_3 := \lambda(x_1)\lambda(x_3)two(b)(R^2_2(x_1)(x_3)), \tag{33c}$$

$$R^2_2 := \lambda(x_1)\lambda(x_3)\lambda(x_2)give(x_1)(x_2)(x_3), \tag{33d}$$

$$b := paper,\ s := student,\ p := professor\ \} \tag{33e}$$

by 3 times (γ)

Note that by using indexed variables corresponding to the order of the argument slots of the constant $give$, i.e., $give(x_1)(x_2)(x_3)$, we maintain expressing the order of the quantifiers that bind the corresponding variables

filling up those argument slots. Thus, the quantifier order is expressed by the order of the λ-abstracts in the recursion assignment for the constant *give* rendering the head verb of the sentence S in (25). In general, the variable names are irrelevant, in sense that we can rename them, as we wish, in the λ-sub-terms, without variable clashes. However, maintaining the corresponding indexes is not only simple mnemonics, since it represents quantifier order, and represents corresponding bindings. As we shall see in what follows, indexing facilitates the representation of respective bindings, which we will use in representing underspecified order of quantification.

Outside any context available, there may not be enough information to render an ambiguous sentence like (25) to a L^{λ}_{ar}-term with a single, specific, quantifier scope distribution. And even in a specific context, the scope distribution is still dependent on agents in it. From computational point, it is inefficient to render such a sentence to the set of all possible distributions of scopes. Even when impossible distributions of quantifier order are factored out, e.g., by lexical or other type incompatibilities, more complex sentences can have multiple, alternative quantifier scopes. This topic continues to be one of the major difficulties in computational semantics and language processing. Here, we present a formal approach to it by using the formal calculi of the typed theory of Acyclic Recursion, see (Moschovakis, 2006) and (Loukanova, 2017).

4.2 Generalized Quantifiers

We consider state-dependent generalized quantifiers that are special cases of n-argument relations Q between state-dependent sets of objects of types $(\mathsf{s} \to \sigma_1), \ldots, (\mathsf{s} \to \sigma_n)$, for $n \geq 1$. I.e., a state-dependent, generalized quantifier Q is a relation $Q \subseteq D_1 \times \cdots \times D_{(n-1)} \times D_n$, where, for $i \in \{1, \ldots, n\}$, $D_i = \mathbb{T}_{((\mathsf{s}\to\sigma_i)\to(\mathsf{s}\to\mathsf{t}))}$, in each state s, is a set of subsets of the domain of state-dependent objects of type σ_i. I.e., with the state dependence abbreviated as $\widetilde{\sigma}_i \equiv \mathsf{s} \to \sigma_i$, $D_i = \mathbb{T}_{(\widetilde{\sigma}_i \to \widetilde{\mathsf{t}})}$ $(i \in \{1, \ldots, n\})$. For quantifiers from human language, we sometimes call the set $D_1 \times \cdots \times D_{(n-1)}$ *the cartesian domain* of Q; D_i, for $i \in \{1, \ldots, (n-1)\}$, its *$i$-th domain*; and D_n *the range* of of Q. Instead of D_n, we often use the letters R and r (with or without indexes) instead of D_n as mnemonic for the range of Q. We also use upper and lower case letters, with and without indexes, Q and q for quantifiers, and D and d for domains. What properties a relation Q needs to satisfy to be qualified as a quantifier is not in the subject of this paper, and we take them as n-argument relations $Q \subseteq D_1 \times \cdots \times D_{(n-1)} \times D_n$, $D_i = \mathbb{T}_{(\widetilde{\sigma}_i \to \widetilde{\mathsf{t}})}$ $(i \in \{1, \ldots, n\})$. Furthermore, in this paper, we focus on the special case of $n = 2$, 2-argument generalized quantifiers, where $\sigma_i \equiv \mathsf{e}$, from which we can make generalization to n-argument quantifier relations between state-dependent sets of objects of state dependent types $\widetilde{\sigma}_i$, for any natural number $n \in \mathbb{N}$.

In L^{λ}_{ar}, in general and including in this paper, we use Curry coding of relations with unary functions and corresponding terms denoting them. We consider L^{λ}_{ar}-terms Q denoting quantifiers (instead of their denotations). A L^{λ}_{ar}-term D_i that denotes the i-th domain or the range of a quantifier denoted by a L^{λ}_{ar}-term Q is $\mathsf{D}_i : (\widetilde{\sigma}_i \to \widetilde{t})$. Thus, a L^{λ}_{ar}-term Q denoting an n-ary, generalized quantifier is of type (34a), and we consider the 2-argument quantifiers of type (34b).

$$\mathsf{Q} : \big((\widetilde{\sigma}_1 \to \widetilde{\mathsf{t}}) \to \cdots \to \big((\widetilde{\sigma}_n \to \widetilde{\mathsf{t}}) \to \widetilde{\mathsf{t}}\big)\big), \qquad \text{for } n \in \mathbb{N} \tag{34a}$$

$$\mathsf{Q} : \big((\widetilde{\mathsf{e}} \to \widetilde{\mathsf{t}}) \to \big((\widetilde{\mathsf{e}} \to \widetilde{\mathsf{t}}) \to \widetilde{\mathsf{t}}\big)\big) \tag{34b}$$

A L^{λ}_{ar}-term Q for a 2-argument, generalized quantifier between individuals of type $\widetilde{\mathsf{e}}$, e.g., a constant *some*, *every*, *two*, etc., is of type (34b), and denotes the characteristic function $\mathbb{T}_{\left((\widetilde{\mathsf{e}}\to\widetilde{\mathsf{t}})\to((\widetilde{\mathsf{e}}\to\widetilde{\mathsf{t}})\to\widetilde{\mathsf{t}})\right)}$ of a relation $\mathbb{T}_{\left((\widetilde{\mathsf{e}}\to\widetilde{\mathsf{t}})\times(\widetilde{\mathsf{e}}\to\widetilde{\mathsf{t}})\to\widetilde{\mathsf{t}}\right)}$ between properties of entities of the domain $\mathbb{T}_{\widetilde{\mathsf{e}}}$.

4.3 Combinatorial Permutations of Quantifier Scopes

In the major Section 5, we develop technique for representing multiple, alternative terms, each representing a specific quantifier distribution, by a single, underspecified L^{λ}_{ar}-term. Such an underspecified term has free recursion variables for quantifiers, that leaves the scope distributions open, to be specified when sufficient information is available, by context. Before that, in this section, we make general observations, with formal representations by L^{λ}_{ar}-terms of the shared patterns in specific quantifier distributions. By this, we formalize the linkage over the argument slots that are bound by the corresponding quantifiers. These formal linkages are exhibited formally by the λ-abstractions over corresponding applications and are maintained during reduction steps. We use permutation functions that represent the specific quantifier distributions. The canonical forms of the renderings in Section 4.1 represent the common pattern of the quantification structure.

From the above template examples of quantifier distribution in Section 4.1, we can conclude a general pattern. The general pattern provides instantiations to specific instances of:

(1) quantifiers, e.g., *every*, *some*, *one*, *two*, etc.;

(2) quantifier scope distribution;

(3) quantifier domains. e.g., *man*, *student*, *professor*, *paper*, etc.;

(4) quantifier range, which can be provided by rendering of a head verb, e.g., *give* in the examples in Section 4.1, or other syntactic head construction.

General Case: Given a permutation $\pi\colon \{\,1,\ldots,n\,\} \to \{\,1,\ldots,n\,\}$, for $n \geq 1$, we take recursion variables $Q_i, R_i, q_i, d_i, h \in \mathsf{RecVars}$ that satisfy, respectively, (35a), (35b), (35c), (35d), (35e), for $i \in \{\,1,\ldots,n\,\}$:

$$Q_i \in \mathsf{RecVars}_{\left((\widetilde{\sigma}_{(i,m_i)}\to\widetilde{\mathsf{t}})\to\widetilde{\mathsf{t}}\right)}, \tag{35a}$$

$$R_i \in \mathsf{RecVars}_{(\widetilde{\sigma}_{(i,m_i)}\to\widetilde{\mathsf{t}})}, \tag{35b}$$

$$q_i \in \mathsf{RecVars}_{\left((\widetilde{\sigma}_{(i,1)}\to\widetilde{\mathsf{t}})\to\cdots\to\left((\widetilde{\sigma}_{(i,m_i)}\to\widetilde{\mathsf{t}})\to\widetilde{\mathsf{t}}\right)\right)} \tag{35c}$$

$$d_{(i,j)} \in \mathsf{RecVars}_{(\widetilde{\sigma}_{(i,j)}\to\widetilde{\mathsf{t}})}, \quad \text{for } j = 1,\ldots,(m_i-1) \tag{35d}$$

$$h \in \mathsf{RecVars}_{\left(\widetilde{\sigma}_{(1,m_1)}\to\ldots\left((\widetilde{\sigma}_{(n,m_n)}\to\widetilde{\mathsf{t}})\to\widetilde{\mathsf{t}}\right)\right)} \tag{35e}$$

In general, $n, i, m_i \in \mathbb{N}$.

In this paper, we exemplify the introduction of the general technique by 3-argument relations represented by corresponding terms, and taking three quantifier terms filling up the argument slots of the corresponding 3-argument terms. This is a reasonable case, which can be generalized to more general cases with $n \geq 2$.

By taking 3-argument terms such as the constant *give*, we have $i = 3$. Any three quantifiers, which are appropriate to fill up the argument slots of *give* and the corresponding 3-argument terms, are of types $\sigma_{(i,j)} \equiv \mathsf{e}$.

Given a permutation $\pi\colon \{\,1,\ldots,n\,\} \to \{\,1,\ldots,n\,\}$, for $n \geq 1$, we take recursion variables $Q_i, R_i, q_i, d_i, h \in \mathsf{RecVars}$ that are appropriately typed, i.e., such that satisfy, respectively, (36a), (36b), (36c), for $i \in \{\,1,\ldots,n\,\}$, e.g. with $n = 3$:

$$Q_i \in \mathsf{RecVars}_{\left((\widetilde{\mathsf{e}}\to\widetilde{\mathsf{t}})\to\widetilde{\mathsf{t}}\right)}, \qquad R_i \in \mathsf{RecVars}_{(\widetilde{\mathsf{e}}\to\widetilde{\mathsf{t}})}, \tag{36a}$$

$$q_i \in \mathsf{RecVars}_{\left((\widetilde{\mathsf{e}}\to\widetilde{\mathsf{t}})\to\left((\widetilde{\mathsf{e}}\to\widetilde{\mathsf{t}})\to\widetilde{\mathsf{t}}\right)\right)}, \qquad d_i \in \mathsf{RecVars}_{(\widetilde{\mathsf{e}}\to\widetilde{\mathsf{t}})}, \tag{36b}$$

$$h \in \mathsf{RecVars}_{\left(\widetilde{\sigma}_1 \to \ldots (\widetilde{\sigma}_n \to \widetilde{\mathsf{t}})\right)}, \qquad \text{for } i \in \{\, 1, \ldots, n \,\} \tag{36c}$$

The term in (37a)–(37g) represents each of the special cases of distribution of quantifier scopes in Section 4.1, for a specific permutation π. The essential differences are: (1) The head parts of the terms in Section 4.1 are moved into the recursion system, via a respective assignment (37d) to the recursion variable $R_{n+1} \in \mathsf{RecVars}_{\widetilde{\mathsf{t}}}$. (2) The L^{λ}_{ar} constant *give* is moved into an assignment, in (37g). (3) Quantifiers are moved into assignments, in (37e), (37f). The term in (37a)–(37g) is an instantiation of a more general pattern of rendering quantifier relations.

$$\begin{aligned}
A \equiv R_4 \text{ where } \{\, R^2_{\pi(3)} &:= \lambda(x_{\pi(1)})\lambda(x_{\pi(2)})\lambda(x_{\pi(3)})h(x_1)(x_2)(x_3), && (37a)\\
R^1_{\pi(2)} &:= \lambda(x_{\pi(1)})\lambda(x_{\pi(2)})Q_{\pi(3)}[R^2_{\pi(3)}(x_{\pi(1)})(x_{\pi(2)})], && (37b)\\
R_{\pi(1)} &:= \lambda(x_{\pi(1)})Q_{\pi(2)}[R^1_{\pi(2)}(x_{\pi(1)})], && (37c)\\
R_4 &:= Q_{\pi(1)}[R_{\pi(1)}], && (37d)\\
Q_1 := q_1(s),\ Q_2 &:= q_2(b),\ Q_3 := q_3(p), && (37e)\\
q_1 := \mathit{some},\ q_2 &:= \mathit{two},\ q_3 := \mathit{every}, && (37f)\\
h := \mathit{give},\ b &:= \mathit{paper},\ s := \mathit{student},\ p := \mathit{professor}\,\} && (37g)
\end{aligned}$$

By the extended γ-reduction, using the γ-rule given on p. 24 in Section 3, the Referential Synonymy Theorem 1, and the Referential γ-Synonymy Theorem 3, the (patterns of) terms for general quantifiers, like (28h)–(28l) and (33e)–(33a), can be reduced to the term Q, (42a)–(42k). In a brief summary, the term Q has congruent canonical forms with respect to renaming the pure variables in the λ-abstracts, as well as the recursion variables bound by the constant where. However, we shall maintain the indexes corresponding to the order of the argument slots, because this style provides visualization of linking the quantifier bindings.

The terms in (37a)–(37g) and (38a)–(38h) are algorithmically γ-synonymous, i.e., $A \approx_{\gamma} B$.

$$\begin{aligned}
B \equiv R_4 \text{ where } \{\, R^2_{\pi(3)} &:= \lambda(x_{\pi(1)})\lambda(x_{\pi(2)})\lambda(x_{\pi(3)})h(x_1)(x_2)(x_3), && (38a)\\
R^1_{\pi(2)} &:= \lambda(x_{\pi(1)})\lambda(x_{\pi(2)})Q_{\pi(3)}[&& (38b)\\
&\qquad \lambda(x_{\pi(3)})R^2_{\pi(3)}(x_{\pi(1)})(x_{\pi(2)})(x_{\pi(3)})], && (38c)\\
R_{\pi(1)} &:= \lambda(x_{\pi(1)})Q_{\pi(2)}[\lambda(x_{\pi(2)})R^1_{\pi(2)}(x_{\pi(1)})(x_{\pi(2)})], && (38d)\\
R_4 &:= Q_{\pi(1)}[\lambda(x_{\pi(1)})R_{\pi(1)}(x_{\pi(1)})], && (38e)\\
Q_1 := q_1(s),\ Q_2 &:= q_2(b),\ Q_3 := q_3(p), && (38f)\\
q_1 := \mathit{some},\ q_2 &:= \mathit{two},\ q_3 := \mathit{every}, && (38g)\\
h := \mathit{give},\ b &:= \mathit{paper},\ s := \mathit{student},\ p := \mathit{professor}\,\} && (38h)
\end{aligned}$$

The terms in (37a)–(37g) and (38a)–(38h) have congruent forms, with respect to renaming the pure variables in the λ-abstracts, as well as the recursion variables bound by the recursion operator where. The order of the assignments in the scope of where is also irrelevant. Recall that where designates mutual recursion. We use variable names that correspond to the argument indexing. I.e., the argument slots of the recursion variable h are indexed with the natural numbers $1, 2, 3$, corresponding to the currying order of applications. The indexing is useful in representing the permutation possibilities in the order of quantifier scopes, while maintaining the links between the pure variables filling argument slots and the corresponding quantifiers binding those variables. In the terms above, we maintain linking the sub-terms via indexes $i \in \{\, 1, \ldots, n \,\}$, which represents that (1) x_i fills

up the i-th argument slot of h, (2) Q_i λ-binds the i-th argument of h via R_i, It is handy to use pure variables correspondingly indexed, i.e., x_i, across the λ-abstracts as well $\lambda(x_i)$. While the variable names are irrelevant, because bound variables can be properly renamed, using the indexes visualizes the abstract links created by successive λ-abstractions and the corresponding applications.

By the indexing, we represent the following conditions that represent the quantification order, and also which quantifiers bind which variables (contributed by the respective argument slots).

1. π is a permutation $\pi\colon \{1,\ldots,n\} \to \{1,\ldots,n\}$, which represents a specific, alternative binding order: $\langle Q_{\pi(1)},\ldots,Q_{\pi(n)}\rangle$, with $Q_{\pi(n)}$ the innermost, $Q_{\pi(1)}$ the outermost scope. In this paper, we exemplify the special case, $n = 3$.

2. The order of quantification $\langle Q_{\pi(1)},\ldots,Q_{\pi(n)}\rangle$ in the terms (37a)–(37g) and (38a)–(38h) corresponds to the order of the λ-abstractions in the assignment

$$R^{(n-1)}_{\pi(n)} := \lambda(x_{\pi(1)})\ldots\lambda(x_{\pi(n)})h(x_1)\ldots(x_n) \tag{39}$$

3. The order of the λ-abstraction in (40), i.e., (42c)–(42d), corresponds to the order of the applications in $R^{j}_{\pi(j+1)}(x_{\pi(1)})\ldots(x_{\pi(j)})(x_{\pi(j+1)})$, for $j = (n-1),\ldots,1$. This maintains the binding linking across the sub-terms. The quantifier $Q_{\pi(j+1)}$ binds the variable filling up the $\pi(j+1)$-th argument slot of h. The linking is visualized and maintained by the index of the pure variable $x_{\pi(j+1)}$, in (40), i.e., (42c)–(42d).

$$\begin{aligned} R^{(j-1)}_{\pi(j)} := {} & \lambda(x_{\pi(1)})\ldots\lambda(x_{\pi(j)})Q_{\pi(j+1)}[\\ & \lambda(x_{\pi(j+1)})R^{j}_{\pi(j+1)}(x_{\pi(1)})\ldots(x_{\pi(j)})(x_{\pi(j+1)})] \\ \text{for } j = {} & (n-1),\ldots,1. \end{aligned} \tag{40}$$

4. The outermost quantifier $Q_{\pi(1)}$ binds the variable filling the $\pi(1)$-the argument slot of h, visualized by the pure variable $x_{\pi(1)}$, in (41), and respectively, in (42i).

$$R_{(n+1)} := Q_{\pi(1)}[\lambda(x_{\pi(1)})R_{\pi(1)}(x_{\pi(1)})] \tag{41}$$

The term B in (38a)–(38h) is a special case of the term Q, (42a)–(42k). For any arbitrarily fixed, but specific permutation, $\pi\colon \{1,\ldots,n\} \to \{1,\ldots,n\}$, the term Q, in (42a)–(42k), represents a general pattern for n-argument relations encoded by curryed functions h, and n quantifiers q_i, $i = 1,\ldots,n$, binding the pairwise different variables filling the argument slots of h, via $Q_i := q_i(d_i)$. This term is underspecified in two aspects. Firstly, the recursion variables h, q_i, d_i are free in Q, i.e., $h, q_i, d_i \in \mathsf{FreeV}(Q)$. Secondly, the permutation π is arbitrary. Of course, in sentences of human language, some permutations can have no reasonable interpretations. Excluding them, can be handled by lexical restrictions, syntactically and semantically, which are not in the subject of this paper.

$$Q \equiv R_{(n+1)} \text{ where } \{ \tag{42a}$$

$$R^{(n-1)}_{\pi(n)} := \lambda(x_{\pi(1)})\ldots\lambda(x_{\pi(n)})h(x_1)\ldots(x_n) \tag{42b}$$

$$R^{(j-1)}_{\pi(j)} := \lambda(x_{\pi(1)})\ldots\lambda(x_{\pi(j)})Q_{\pi(j+1)}[\tag{42c}$$

$$\lambda(x_{\pi(j+1)})R^{j}_{\pi(j+1)}(x_{\pi(1)})\ldots(x_{\pi(j)})(x_{\pi(j+1)})] \tag{42d}$$

$$(\text{for } j = (n-1),\ldots,1)$$

$$R^{2}_{\pi(3)} := \lambda(x_{\pi(1)})\lambda(x_{\pi(2)})\lambda(x_{\pi(3)})Q_{\pi(4)}[\tag{42e}$$

$$\lambda(x_{\pi(4)})R^{j}_{\pi(4)}(x_{\pi(1)})(x_{\pi(2)})(x_{\pi(3)})] \tag{42f}$$

$$R^{1}_{\pi(2)} := \lambda(x_{\pi(1)})\lambda(x_{\pi(2)})Q_{\pi(3)}[\tag{42g}$$

$$\lambda(x_{\pi(3)})R^{j}_{\pi(3)}(x_{\pi(1)})(x_{\pi(2)})(x_{\pi(3)})] \tag{42h}$$

$$R_{\pi(1)} := \lambda(x_{\pi(1)})Q_{\pi(2)}[\lambda(x_{\pi(2)})R^{1}_{\pi(2)}(x_{\pi(1)})(x_{\pi(2)})], \tag{42i}$$

$$R_{(n+1)} := Q_{\pi(1)}[\lambda(x_{\pi(1)})R_{\pi(1)}(x_{\pi(1)})], \tag{42j}$$

$$Q_i := q_i(d_i) \qquad (\text{for } i = 1, \ldots, n; n \geq 1)\,\} \tag{42k}$$

Note that $Q_i, R_i, q_i, d_i, h \in \mathsf{RecVars}$ in (42a)–(42k), are of types given in (36a), (36b), (36c), for $i \in \{\, 1, \ldots, n \,\}$. The terms in (37a)–(37g) and (38a)–(38h), and in Section 4.1, are special cases for $n = 3$, with instantiations, i.e. specifications, of the recursion variables $q_i, d_i, h \in \mathsf{RecVars}$, by adding recursion assignments.

While the term Q in (42a)–(42k) is underspecified with respect to the free recursion variables $h, q_i, d_i \in \mathsf{FreeV}(Q)$, the order of the scopes of Q_i, for $i = 1, \ldots, n$, is specified by any given, specific permutation π, $\pi\colon \{\, 1, \ldots, n \,\} \to \{\, 1, \ldots, n \,\}$. One way to represent the underspecified quantification order could be to leave the permutation function π underspecified, i.e., without being instantiated. However, then the underspecified π is at meta-theoretical level, outside of L^{λ}_{ar}. It is possible to represent underspecified scope distribution by using a term like (42a)–(42k), with possible specification of $q_i, d_i, h \in \mathsf{RecVars}$, and by leaving the permutation function π underspecified, except suitable restrictions over π depending on the specifications of $q_i, d_i, h \in \mathsf{RecVars}$. However by using underspecified function constant π, the expression (42a)–(42k) is not anymore L^{λ}_{ar}-term. It is a schemata, i.e., a pattern, in a meta-language, external to L^{λ}_{ar}, from which L^{λ}_{ar}-terms can be obtained by specific instances of permutations π. There is a better technique, presented in the next section, which provides specific cases for π. It also is flexible with respect to imposing constraints on excluding some of the permutations π. Such constraints depend on specifications of the recursion variables $q_i, d_i, h \in \mathsf{RecVars}$ with specific relations. Such restrictions are not in the subject of this paper. Typically, they depend on the semantic properties of the relations, but also on lexical classifications of human languages.

5. Underspecified Scope Distribution

The expression (42a)–(42k) implicitly carries a pattern for underspecified L^{λ}_{ar}-term that represents underspecified scope of the relations Q_i. In this section, we introduce a technique for underspecified quantification in the case $n = 3$, which ten can be generalized to $n \in \mathbb{N}$. We bring again, temporarily the specifications of $q_i, d_i, h \in \mathsf{RecVars}$ as in Section 4.1 to illustrate the technique. Note that we use extended terms with additional sub-expressions (43e) that add constraints over free recursion variables, as introduced in (Loukanova, 2013). The technique introduced here uses the formal representation of the links that maintain the binding argument slots corresponding to quantification across λ-abstractions and reductions to canonical forms, visualized via indexing.

The formal definition of the constraints that Q_i λ-binds the i-th argument of h via R_i, (for $i = 1, \ldots, 3$), in (43e) is rather technical and spacious and we leave it outside the subject of this paper, for an extended paper. Here we note that the definition formalizes the linking of each quantifier Q_i with the variable x_i that it binds, i.e, with the corresponding i-th argument slot filled up by x_i, by avoiding explicit usage of metalanguage symbols $Q_{\pi(i)}$ with a unspecified permutation $\pi\colon \{\, 1, \ldots, n \,\} \to \{\, 1, \ldots, n \,\}$. (Loukanova, 2013) uses another kind of constraints, and the relation between them and the constraints in (43e) is also outside the subject of this paper. Here we only mention that the choice between them is open and depends on possible applications of the quantifier underspecification. An important difference is that the technique presented here is more general and applicable for any abstract, i.e., mathematical, n-ary quantifier relations, $n \geq 2$. Such quantifiers are abstract mathematical

objects, in syntax and semantics of formal languages, not only those originating in human language NPs and sentences.

$$
\begin{aligned}
U \equiv R_4 \text{ where } \{\, & l_1 := Q_1(R_1),\, l_2 := Q_2(R_2),\, l_3 := Q_3(R_3), && \text{(43a)}\\
& Q_1 := q_1(d_1),\, Q_2 := q_2(d_2),\, Q_3 := q_3(d_3), && \text{(43b)}\\
& q_1 := \mathit{some},\, q_2 := \mathit{two},\, q_3 := \mathit{every}, && \text{(43c)}\\
& d_1 := \mathit{student},\, d_2 := \mathit{paper},\, d_3 := \mathit{professor},\, h := \mathit{give}\,\} && \text{(43d)}\\
\text{s.t. } \{\, & Q_i \ \lambda\text{-binds the } i\text{-th argument of } h \text{ via } R_i, && \text{(43e)}\\
& R_4 \text{ is assigned to a closed subterm with} \\
& \text{fully scope-specified } Q_i && \text{(43f)}\\
& (\text{for } i = 1, \ldots, 3),\, \}
\end{aligned}
$$

Now, from the underspecified (43a)–(43f), we derive one of the possible closed L^{λ}_{ar}-terms, (44a)–(44j), having fully specified quantifier scopes:

Note:

(1) The λ-abstractions are the tool for linking the quantifiers with the respective argument slots that they bind, i.e., in satisfying the constraints (43e)–(43f).

(2) The λ-abstracts are nested within the where-scopes, accordingly, by the dependencies.

$$
\begin{aligned}
U_{321} \equiv R_4 \text{ where } \{ & && \text{(44a)}\\
& R_4 := l_3,\, l_3 := Q_3(R_3),\, Q_3 := q_3(d_3), && \text{(44b)}\\
& q_3 := \mathit{every},\, d_3 := \mathit{professor}, && \text{(44c)}\\
& R_3 := \lambda(x_3)\Big[_3\, l_2 \text{ where } \{_3\, l_2 := Q_2(R_2),\, Q_2 := q_2(d_2), && \text{(44d)}\\
& \qquad q_2 := \mathit{two},\, d_2 := \mathit{paper}, && \text{(44e)}\\
& \qquad R_2 := \lambda(x_2)\big[_2 l_1 \text{ where } \{_2\, l_1 := Q_1(R_1), && \text{(44f)}\\
& \qquad\qquad Q_1 := q_1(d_1), && \text{(44g)}\\
& \qquad\qquad q_1 := \mathit{some},\, d_1 := \mathit{student}, && \text{(44h)}\\
& \qquad\qquad R_1 := \lambda(x_1)h(x_1)(x_2)(x_3), && \text{(44i)}\\
& \qquad\qquad h := \mathit{give}\,\}_2\big]_2 && \text{(44j)}\\
& \qquad \}_3\Big]_3\, \}\\
\text{s.t. } \{\, & Q_i \ \lambda\text{-binds the } i\text{-th argument of } h \text{ via } R_i, && \text{(44k)}\\
& R_4 \text{ is assigned to a closed subterm with} && \text{(44l)}\\
& \text{fully scope-specified } Q_i \\
& (\text{for } i = 1, \ldots, 3),\, \}
\end{aligned}
$$

By using reductions, including the important (λ) and (γ) rules, similarly to the ones in Section 4.1, we reduce the term U_{321} in (44a)–(44j), to the γ-canonical form in (45a)–(45j). Note that these reductions use more applications

of the (γ) rule (11 times), due to the additional assignments in the scope of the λ-abstractions, which are subject to the (λ) rule.

$$\mathsf{cf}_\gamma(U_{321}) \equiv R_4 \text{ where } \{ \tag{45a}$$
$$R_4 := l_3,\ l_3 := Q_3(R_3), \tag{45b}$$
$$Q_3 := q_3(d_3), \tag{45c}$$
$$R_3 := \lambda(x_3)l_2^1(x_3),\ l_2^1 := \lambda(x_3)Q_2(R_2^1(x_3)), \tag{45d}$$
$$Q_2 := q_2(d_2), \tag{45e}$$
$$R_2^1 := \lambda(x_3)\lambda(x_2)l_1^2(x_3)(x_2),\ l_1^2 := \lambda(x_3)\lambda(x_2)Q_1(R_1^2(x_3)(x_2)), \tag{45f}$$
$$Q_1 := q_1(d_1), \tag{45g}$$
$$R_1^2 := \lambda(x_3)\lambda(x_2)\lambda(x_1)h(x_1)(x_2)(x_3), \tag{45h}$$
$$q_3 := \mathit{every},\ d_3 := \mathit{professor},\ q_2 := \mathit{two},\ d_2 := \mathit{paper}, \tag{45i}$$
$$q_1 := \mathit{some},\ d_1 := \mathit{student},\ h := \mathit{give}\,\} \tag{45j}$$

Now, we can simplify the term (45a)–(45j) by preserving its essential algorithmic steps. Each pair of the first two assignments in (45b), (45d), and (45f) can be merged. Formally, this merging is via extending the reduction calculi of L_{ar}^{λ} by adding suitable reduction rules, which is not in the subject of this paper. The result is the term S_{321}, (46a)–(46j), that is not algorithmically (step-by-step) equivalent to the terms U_{321}, (44a)–(44j), and $\mathsf{cf}_\gamma(U_{321})$, (45a)–(45j). While U_{321} and $\mathsf{cf}_\gamma(U_{321})$ are algorithmically equivalent, i.e., $U_{321} \approx \mathsf{cf}_\gamma(U_{321})$. However, the term S_{321}, (46a)–(46j), is more simple, by avoiding the unnecessary computations denoted by the merged assignments. Otherwise, U_{321}, (44a)–(44j), preserves all other computational steps, represented by the assignments.

$$S_{321} \equiv R_4 \text{ where } \{ \tag{46a}$$
$$R_4 := Q_3(R_3), \tag{46b}$$
$$Q_3 := q_3(d_3), \tag{46c}$$
$$R_3 := \lambda(x_3)Q_2(R_2^1(x_3)), \tag{46d}$$
$$Q_2 := q_2(d_2), \tag{46e}$$
$$R_2^1 := \lambda(x_3)\lambda(x_2)Q_1(R_1^2(x_3)(x_2)), \tag{46f}$$
$$Q_1 := q_1(d_1), \tag{46g}$$
$$R_1^2 := \lambda(x_3)\lambda(x_2)\lambda(x_1)h(x_1)(x_2)(x_3), \tag{46h}$$
$$q_3 := \mathit{every},\ d_3 := \mathit{professor},\ q_2 := \mathit{two},\ d_2 := \mathit{paper}, \tag{46i}$$
$$q_1 := \mathit{some},\ d_1 := \mathit{student},\ h := \mathit{give}\,\} \tag{46j}$$

6. Further Generalizations

Next, we can remove the assignments for the specific constants *every*, *some*, *two*, *professor*, *student*, *paper*, *give*, from each of the above terms, U, $\mathsf{cf}_\gamma(U_{321})$, and S_{321}, and get further generalizations of the patterns in them, i.e., V, G, G_{321}, respectively. By removing the specific assignments (43c)–(43d) from the term (43a)–(43f), and dropping out $h := \mathit{give}$, we obtain an underspecified term (47a)–(47e), i.e., a term with free recursion variables, which is a general computational pattern for computing denotations of an entire class $\mathcal{C}$ of 3-argument relations h, such as *give*, etc., where each of three 2-argument, unknown quantifiers q_i binds the respective

i-th argument slot of h, but with underspecified order of binding. Note that h is a free recursion variable of (47a)–(47e), by (47d), $h \in \mathsf{RecVars}_{\left(\widetilde{\mathsf{e}}\to(\widetilde{\mathsf{e}}\to(\widetilde{\mathsf{e}}\to\widetilde{\mathsf{t}}))\right)}$.

$$
\begin{aligned}
V \equiv R_4 \text{ where } \{\, & l_1 := Q_1(R_1),\ l_2 := Q_2(R_2),\ l_3 := Q_3(R_3), && \text{(47a)}\\
& Q_1 := q_1(d_1),\ Q_2 := q_2(d_2),\ Q_3 := q_3(d_3),\ \} && \text{(47b)}\\
\text{s.t. } \{\, & Q_i \ \lambda\text{-binds the } i\text{-th argument of } h \text{ via } R_i && \text{(47c)}\\
& h : \left(\widetilde{\mathsf{e}} \to (\widetilde{\mathsf{e}} \to (\widetilde{\mathsf{e}} \to \widetilde{\mathsf{t}}))\right), && \text{(47d)}\\
& R_4 \text{ is assigned to a closed subterm with} && \text{(47e)}\\
& \text{fully scope specified } Q_i\\
& R_4 \text{ dominates each } Q_i \text{ (for } i = 1, \ldots, 3)\,\}
\end{aligned}
$$

An additional assignment for h may be useful, even if still underspecified, for example, by replacing $h := give$ with $h := \lambda(x_1)\lambda(x_2)\lambda(x_3)g(x_1)(x_2)(x_3)$, is unnecessary. Furthermore, we have that

$$
\begin{aligned}
&\lambda(x_1)\lambda(x_2)\lambda(x_3)g(x_1)(x_2)(x_3) \approx g\\
&\text{for } g \in \mathsf{RecVars}_{\left(\widetilde{\mathsf{e}}\to(\widetilde{\mathsf{e}}\to(\widetilde{\mathsf{e}}\to\widetilde{\mathsf{t}}))\right)}
\end{aligned}
\tag{48}
$$

The type constraint (47d), along with (47c), is sufficient. Strictly speaking, the set of the constraints is a good place to include type assignments to all variables occurring in a term, for which the type can not be derived, in a full formalization of the constraint sub-expressions.

Similarly, (44a)–(44l) is generalized to:

$$
\begin{aligned}
G \equiv R_4 \text{ where } \{ & && \text{(49a)}\\
& R_4 := Q_3(R_3),\ Q_3 := q_3(d_3), && \text{(49b)}\\
& R_3 := \lambda(x_3)\Big[_3\, l_2 \text{ where } \{_3\, l_2 := Q_2(R_2),\ Q_2 := q_2(d_2), && \text{(49c)}\\
& \qquad R_2 := \lambda(x_2)\big[_2 l_1 \text{ where } \{_2\, l_1 := Q_1(R_1), && \text{(49d)}\\
& \qquad\quad Q_1 := q_1(d_1), && \text{(49e)}\\
& \qquad\quad R_1 := \lambda(x_1)h(x_1)(x_2)(x_3)\ \}_2\big]_2\ \}_3\Big]_3\ \} && \text{(49f)}\\
\text{s.t. } \{ & \ Q_i \ \lambda\text{-binds the } i\text{-th argument of } h \text{ via } R_i, && \text{(49g)}\\
& \quad \left.\begin{array}{l} R_4 \text{ is assigned to a closed subterm with}\\ \text{fully scope specified } Q_i \text{ (for } i = 1, \ldots, 3), \end{array}\right\} && \text{(49h)}
\end{aligned}
$$

By removing (46i)–(46j) from the term (46a)–(46j), we obtain a new, underspecified term, (50a)–(50e), which represents the algorithmic semantics of the subclass $\mathcal{C}_{321} \subseteq \mathcal{C}$, where the unspecified (unknown) quantifiers Q_i, $i = 1, 2, 3$, bind in the specific order $\langle Q_3, Q_2, Q_1\rangle$: Q_1 is the innermost, Q_3 is the topmost. Respectively, the binary quantifiers q_i, $i = 1, 2, 3$ bind in the specific order $\langle q_3, q_2, q_1\rangle$: q_1 is the innermost, q_3 is the topmost.

$$
\begin{aligned}
G_{321} \equiv R_4 \text{ where } \{ & && \text{(50a)}\\
& R_4 := Q_3(R_3),\ Q_3 := q_3(d_3), && \text{(50b)}\\
& R_3 := \lambda(x_3)Q_2(R_2^1(x_3)),\ Q_2 := q_2(d_2), && \text{(50c)}
\end{aligned}
$$

$$R^1_2 := \lambda(x_3)\lambda(x_2)Q_1(R^2_1(x_3)(x_2)),\ Q_1 := q_1(d_1), \tag{50d}$$

$$R^2_1 := \lambda(x_3)\lambda(x_2)\lambda(x_1)h(x_1)(x_2)(x_3)\,\} \tag{50e}$$

Similarly, for any permutation $\pi\colon \{1,\ldots,n\} \to \{1,\ldots,n\}$, we get the specification of the quantifier order in (51a)–(51f), while the actual quantifiers, their domains and ranges, are underspecified. The quantifiers Q_i, $i = 1,2,3$, bind in the specific order $\langle Q_{\pi(3)}, Q_{\pi(2)}, Q_{\pi(1)}\rangle$: $Q_{\pi(1)}$ is the innermost, $Q_{\pi(3)}$ is the topmost. This is represented semantically by the term (51a)–(51f) Respectively, the binary quantifiers q_i, $i = 1,2,3$ bind in the specific order $\langle q_{\pi(3)}, q_{\pi(2)}, q_{\pi(1)}\rangle$: $q_{\pi(1)}$ is the innermost, $q_{\pi(3)}$ is the topmost.

$$G_{\pi(123)} \equiv R_4 \text{ where } \{ \tag{51a}$$

$$R_4 := Q_{\pi(3)}(R_{\pi(3)}),\ Q_{\pi(3)} := q_{\pi(3)}(d_{\pi(3)}), \tag{51b}$$

$$R_{\pi(3)} := \lambda(x_{\pi(3)})Q_{\pi(2)}(R^1_{\pi(2)}(x_{\pi(3)})),\ Q_{\pi(2)} := q_{\pi(2)}(d_{\pi(2)}), \tag{51c}$$

$$R^1_{\pi(2)} := \lambda(x_{\pi(3)})\lambda(x_{\pi(2)})Q_{\pi(1)}(R^2_{\pi(1)}(x_{\pi(3)})(x_{\pi(2)})), \tag{51d}$$

$$Q_{\pi(1)} := q_{\pi(1)}(d_{\pi(1)}), \tag{51e}$$

$$R^2_{\pi(1)} := \lambda(x_{\pi(3)})\lambda(x_{\pi(2)})\lambda(x_{\pi(1)})h(x_{\pi(1)})(x_{\pi(2)})(x_{\pi(3)})\,\} \tag{51f}$$

Also, for any permutation $\pi\colon \{1,\ldots,n\} \to \{1,\ldots,n\}$, we get (37a)–(37g), where, in addition, we have specified instantiations to the constants. Note that according to the instantiations by the assignments in (37a)–(37g), the quantifiers Q_i, $i = 1,2,3$ bind in the specific order $\langle Q_{\pi(1)}, Q_{\pi(2)}, Q_{\pi(3)}\rangle$: $Q_{\pi(3)}$ is the innermost, $Q_{\pi(1)}$ is the topmost. Respectively, the binary quantifiers q_i, $i = 1,2,3$ bind in the specific order $\langle q_{\pi(1)}, q_{\pi(2)}, q_{\pi(3)}\rangle$: $q_{\pi(3)}$ is the innermost, $q_{\pi(1)}$ is the topmost.

7. Conclusions

7.1 Discussion

In this paper, we have introduced a technique of underspecified, acyclic recursion, for representation of a class of relations, belonging to the same class as quantifiers, that can bind arguments by multiple, ambiguous binding scope. Several, e.g., n, quantifiers, can interact and bind the arguments of n-arguments relations ($n \geq 2$), in alternative orders depending on context and agents in context. The technique gives possibilities for leaving the order of quantifiers underspecified, in the absence of relevant information.

The order of the quantifier scopes, i.e., the order in which several quantifiers bind arguments of a relation, or a function, having n-arguments ($n \geq 2$), is typically dependent on specific contexts and agents. The quantifiers, and the relations whose argument slots they bind, can also be underspecified. Thus, the term Q, (42a)–(42k), is a computational pattern that represents a wider class of binding relations that can bind in alternative orders represented by permutation function π. It is not computationally efficient to generate the set of all possible alternatives π for binding orders, without context, and even in specific context without sufficient information. This is not also rational from general considerations, e.g., cognition, and how information should be presented and processed efficiently.

The formal theory L^{λ}_{ar} provides highly expressive computational utilities, including for representation of algorithmic semantics that is underspecified, while maintaining algorithmic structure that can be expanded and specified when more information is available. E.g., without context and sufficient information, the semantic information carried by a sentence like "Every professor gives some student two papers", does not need to be represented by the set of all alternatives, i.e., both scope distributions T_1, (27a)–(27d), and T_2, (32b)–(33e).

It is more efficient and rational to render the common information that is carried by both of these specific interpretations, in an underspecified term U in (43a)–(43f). The L^{λ}_{ar}-term U is in canonical form, i.e., it represents algorithmic instructions for computing the denotations of U depending on context. The algorithmic instructions that are available in U contain available computational structure and facts, in their most basic forms, because U is in a canonical form. In a given context, with available information, an agent (which can be a computational system embedded in a device) can specify U, e.g., to the term U_{321} in (44a)–(44j) by instantiating the binding scope of the quantifiers. Furthermore, the agent can derive, from U_{321}, the more simple term S_{321}, (46a)–(46j).

7.2 Futute Work

Here, we briefly overview several areas of application of the computational technique introduced in this paper, which in the same time are subject of future work and developments.

Computational Semantics. A primary application is to computational semantics of human language. As we described and exemplified in Section 4, human language is abundant of ambiguities that present the major difficulty to computerized processing. Ambiguities and underspecification, typically can be resolved by context. Quantifiers in human language are among the major contributors of ambiguities. Expansion of multiple semantic representations have been avoid by the technique of semantic storage, e.g., see (Loukanova, 2002). While such techniques are successful, they involve meta-theoretic means and are specialized for quantifiers. The technique here has the superiority of using the facilities of the type theory of recursion L^{λ}_{ar} at its object level. In addition, it is applicable to more general relations.

The technique of Minimal Recursion Semantics (MRS), see (Copestake et al., 2005), has been useful for underspecified semantic representation of multiple semantic scopes. MRS has been implemented and used very successfully in large scale grammars, e.g., DELPH-IN (DELPH-IN,) and CSLI Linguistic Grammars Online (CSLI LinGO,). MRS lacks strict logical formalization, and our work provides such via currying encoding of relations. Further work is due for direct, relational formalization, without currying, for semantic representation in large scale grammars, and in computational grammar in general.

Computational Syntax-Semantics Interface. (Loukanova, 2011b; Loukanova, 2012) introduces a technique for syntax-semantics interface in computational grammar, which uses L^{λ}_{ar} for semantic representations, in compositional mode. While that work represents syntactic phrases that include NP quantifiers, quantifier scope ambiguities are not covered. Our upcoming work includes incorporation of the technique for underspecified semantic scopes, introduced in this paper, in computational syntax-semantics interface. The work (Loukanova and Jiménez-López, 2012) can be extended by the introduced technique for underspecified scopes.

Other Applications. We envisage that the formal theory introduced here has many potential applications, where covering semantic information that depends on context and information is important and includes relations that have scope binding. E.g.: (1) type-theoretic foundations of: (a) semantics of programming languages (b) formalization of algorithm specifications, e.g., by higher-order type theory of algorithms L^{λ}_{ar}, L^{λ}_{r}, or their extended, or adapted versions (c) compilers and techniques for converting recursion into tail-recursion and iteration (2) information representation systems, e.g., in: (a) data basis (b) health and medical systems (c) medical sciences (d) legal systems (e) administration.

Many of these areas include and depend on semantic processing of human language. Some of them include semantic data with quantifiers, or other relations having multiple scope binding. In particular, we consider that, for a better success, it is important to develop new approaches in the areas of Machine Learning and Information

Retrieval that use techniques for integration of the quantitative methods (e.g., from mathematical statistics), which, typically, are used in these areas, with logic methods for semantic representations. We consider that L^{λ}_{ar} and its extended versions, by including the technique from this paper, can be very fruitful in such developments.

Mathematical Quantifiers. The technique introduced in this paper is used to represent not only quantifier relations, but also classes of relations that, similarly to quantifiers, have scope dependent arguments, where the scopes depend on order of binding the corresponding arguments. While we illustrate the formalization by examples from human language, it is useful for abstract, mathematical quantifier relations having n-arguments ($n \geq 2$), and for applications in areas with domains consisting of relations between sets of objects. Such applications are subject to future work.

Extending the Formalization. A more immediate future line of work is to provide details of the formalization of the constraints (43e)–(43f) for linking the quantifiers to the respective argument slots they bind. Another line is to relate the constraints on λ-binding variables to the work in (Loukanova, 2013).

Furthermore, the simplifications of the term (45a)–(45j) to the term (46a)–(46j), by reducing "chain" assignments, require extending the reduction calculus.

Implementation. One of the values of the Reduction Calculus of L^{λ}_{ar} is that it is effective, i.e., each term can be reduced to its canonical form and to its γ-canonical form after finite number of steps. Our future work includes implementing it into a computerised system for reducing terms to their canonical forms.

8. References

Copestake, A., Flickinger, D., Pollard, C., and Sag, I., 2005. Minimal recursion semantics: an introduction. *Research on Language and Computation*, 3:281–332. doi:DOI10.1007/s11168-006-6327-9.

CSLI LinGO, Oct 2016 (last accessed). CSLI Linguistic Grammars Online (LinGO) Lab at Stanford University. `http://lingo.stanford.edu`.

DELPH-IN, Oct 2016 (last accessed). Deep Linguistic Processing with HPSG (DELPH-IN). `http://moin.delph-in.net`.

Gallin, D., 1975. *Intensional and Higher-Order Modal Logic*. North-Holland.

Loukanova, R., 2002. Generalized Quantification in Situation Semantics. In Gelbukh, A., editor, *Computational Linguistics and Intelligent Text Processing*, volume 2276 of *Lecture Notes in Computer Science*, pages 46–57. Springer Berlin / Heidelberg. ISBN 3-540-43219-1. doi:10.1007/3-540-45715-1_4.

Loukanova, R., 2011a. Semantics with the Language of Acyclic Recursion in Constraint-Based Grammar. In Bel-Enguix, G. and Jiménez-López, M. D., editors, *Bio-Inspired Models for Natural and Formal Languages*, pages 103–134. Cambridge Scholars Publishing. ISBN (10): 1-4438-2725-8, (13): 978-1-4438-2725-6.

Loukanova, R., 2011b. Syntax-Semantics Interface for Lexical Inflection with the Language of Acyclic Recursion. In Bel-Enguix, G., Dahl, V., and Jiménez-López, M. D., editors, *Biology, Computation and Linguistics — New Interdisciplinary Paradigms*, volume 228 of *Frontiers in Artificial Intelligence and Applications*, pages 215–236. IOS Press, Amsterdam; Berlin; Tokyo; Washington, DC. ISBN 978-1-60750-761-1 (print), 978-1-60750-762-8 (online).

Loukanova, R., 2012. Semantic Information with Type Theory of Acyclic Recursion. In Huang, R., Ghorbani, A. A., Pasi, G., Yamaguchi, T., Yen, N. Y., and Jin, B., editors, *Active Media Technology - 8th International Conference, AMT 2012, Macau, China, December 4-7, 2012. Proceedings*, volume 7669 of *Lecture Notes in Computer Science*, pages 387–398. Springer. ISBN 978-3-642-35235-5.

Loukanova, R., 2013. Algorithmic Granularity with Constraints. In Imamura, K., Usui, S., Shirao, T., Kasamatsu, T., Schwabe, L., and Zhong, N., editors, *Brain and Health Informatics*, volume 8211 of *Lecture Notes in Computer Science*, pages 399–408. Springer International Publishing. ISBN 978-3-319-02752-4. doi: 10.1007/978-3-319-02753-1_40.

Loukanova, R., 2017. γ-Reduction in Type Theory of Acyclic Recursion. (to appear).

Loukanova, R. and Jiménez-López, M. D., 2012. On the Syntax-Semantics Interface of Argument Marking Prepositional Phrases. In Pérez, J. B., Sánchez, M. A., Mathieu, P., Rodríguez, J. M. C., Adam, E., Ortega, A., Moreno, M. N., Navarro, E., Hirsch, B., Lopes-Cardoso, H., and Julián, V., editors, *Highlights on Practical Applications of Agents and Multi-Agent Systems*, volume 156 of *Advances in Intelligent and Soft Computing*, pages 53–60. Springer Berlin / Heidelberg. ISBN 978-3-642-28761-9.

Montague, R., 1973. The proper treatment of quantification in ordinary English. In Hintikka, J., Moravcsik, J., and Suppes, P., editors, *Approaches to Natural Language*, pages 221–242. D. Reidel Publishing Co., Dordrecht, Holland. Reprinted in (Thomason, 1974, pp. 247–270).

Montague, R., 1988. The proper treatment of quantification in ordinary English. In *Philosophy, Language, and Artificial Intelligence*, pages 141–162. Springer.

Moschovakis, Y. N., 1989. The formal language of recursion. *The Journal of Symbolic Logic*, 54(04):1216–1252.

Moschovakis, Y. N., 1997. The logic of functional recursion. In *Logic and Scientific Methods*, pages 179–207. Kluwer Academic Publishers. Springer.

Moschovakis, Y. N., 2006. A logical calculus of meaning and synonymy. *Linguistics and Philosophy*, 29:27–89.

Thomason, R. H., editor, 1974. *Formal Philosophy: Selected Papers of Richard Montague*. Yale University Press, New Haven, Connecticut.