Underspecified Quantification by the Theory of Acyclic Recursion

Roussanka Loukanova

Abstract The paper introduces a technique for representing quantifier relations that can have different scope order depending on context and agents. 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 λ-abstractions. The purpose of the technique is to represent underspecified order of quantification, for computationally efficient and adequate representation of scope ambiguity in the absence of context and corresponding information about the order. Furthermore, the technique is used to represent subclasses of larger classes of relations depending on order of quantification or specific relations.

Keywords Recursion · Type-theory · Semantics · Algorithms · Denotation · Reduction · Quantifiers · Underspecification

1 Background

The formal theory of the technique introduced in the paper is a generalization of the theories of recursion introduced by Moschovakis [12, 13]. 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 $\mathrm{L}_{\mathrm{ar}}^{\lambda}$, see Moschovakis [14], which is more expressive, by adding typed, functional objects. In another aspect, $\mathrm{L}_{\mathrm{ar}}^{\lambda}$ is

R. Loukanova(✉)
Department of Mathematics, Stockholm University, Stockholm, Sweden
e-mail: rloukanova@gmail.com

limited to computations that always close-off, by allowing only acyclic terms. I.e., the class of languages $\mathrm{L}^{\lambda}_{\mathrm{ar}}$, 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 $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ was introduced in Moschovakis [14].

In this paper, we use an extended formal language and theory of $\mathrm{L}^{\lambda}_{\mathrm{ar}}$, 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 $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ introduced in Loukanova [6], which employs an additional reduction rule, γ-rule, see Loukanova [6]. Secondly, we use restrictions over $\mathrm{L}^{\lambda}_{\mathrm{ar}}$-terms introduced in Loukanova [10]. This paper provides also a more general technique than Loukanova [10]. 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 $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ of Moschovakis acyclic recursion, its syntax, denotational and algorithmic semantics, and its theory, is given in Moschovakis [14] and Loukanova [6]. The formal system $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ is a higher-order type theory, which is a proper extension of Gallin's TY_2, see Gallin [4], and thus, of Montague's Intensional Logic (IL), see Montague [15].

2 Brief Introduction to the Type Theory $\mathbf{L}^{\lambda}_{\mathbf{ar}}$

In this paper, we only give brief, informal introduction of $\mathrm{L}^{\lambda}_{\mathrm{ar}}$, for sake of space. For details, see Moschovakis [14] and Loukanova [6].

2.1 *Syntax of* $\mathbf{L}^{\lambda}_{\mathbf{ar}}$

Types of $\mathrm{L}^{\lambda}_{\mathrm{ar}}$: The set Types is the smallest set defined recursively (using a wide-spread notation in computer science): $\tau :\equiv e \mid t \mid s \mid (\tau_1 \to \tau_2)$.

The vocabulary of $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ consists of pairwise disjoint sets of: typed constants, $K = \bigcup_{\tau \in \mathsf{Types}} K_\tau$; typed pure variables, $\mathsf{PureVars} = \bigcup_{\tau \in \mathsf{Types}} \mathsf{PureVars}_\tau$; and typed recursion variables (called also locations), $\mathsf{RecVars} = \bigcup_{\tau \in \mathsf{Types}} \mathsf{RecVars}_\tau$.

The Terms of $\mathrm{L}^{\lambda}_{\mathrm{ar}}$: In addition to application and λ-abstraction terms, $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ has recursion terms that are formed by using a designated recursion operator, denoted by the constant where in infix notation. The recursive rules for the set of $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ terms can be expressed by using a notational variant of "typed" BNF, with the assumed types given as superscripts:

$$A :\equiv \mathsf{c}^\tau : \tau \mid x^\tau : \tau \mid B^{(\sigma\to\tau)}(C^\sigma) : \tau \mid \lambda(v^\sigma)(B^\tau) : (\sigma \to \tau)$$
$$\mid A_0^\sigma \text{ where } \{p_1^{\sigma_1} := A_1^{\sigma_1}, \ldots, p_n^{\sigma_n} := A_n^{\sigma_n}\} : \sigma$$

where $\{p_1^{\sigma_1} := A_1^{\sigma_1}, \ldots, p_n^{\sigma_n} := A_n^{\sigma_n}\}$ is a sequence of assignments that satisfies the following *acyclicity condition*:

Acyclic System of Assignments: 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 an *acyclic system of assignments* iff there is a function $\mathsf{rank} : \{p_1, \ldots, p_n\} \longrightarrow \mathbb{N}$ such that, for all $p_i, p_j \in \{p_1, \ldots, p_n\}$, if p_j occurs freely in A_i then $\mathsf{rank}(p_j) < \mathsf{rank}(p_i)$.

The terms of the form A_0^σ where $\{p_1^{\sigma_1} := A_1^{\sigma_1}, \ldots, p_n^{\sigma_n} := A_n^{\sigma_n}\}$ are called *recursion terms*. We shall skip the type assignments when the types are clear.

2.2 Two Kinds of Semantics of $\mathrm{L}_{\mathrm{ar}}^{\lambda}$

Denotational Semantics of $\mathrm{L}_{\mathrm{ar}}^{\lambda}$. The definition of the denotations of the terms follows the structure of the $\mathrm{L}_{\mathrm{ar}}^{\lambda}$-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 $\mathrm{L}_{\mathrm{ar}}^{\lambda}$ 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:

$$\mathsf{cf}(A) \equiv A_0 \text{ where } \{p_1 := A_1, \ldots, p_n := A_n\} \quad (n \geq 0) \tag{1}$$

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 $\mathrm{L}_{\mathrm{ar}}^{\lambda}$. The reduction calculi and the canonical forms of the terms play an essential role in the algorithmic semantics of $\mathrm{L}_{\mathrm{ar}}^{\lambda}$. The algorithm for computing the denotation $\mathsf{den}(A)$ of a meaningful $\mathrm{L}_{\mathrm{ar}}^{\lambda}$-term A, is determined by its canonical form[1]. E.g., the sentence (2a) can be rendered into the $\mathrm{L}_{\mathrm{ar}}^{\lambda}$-term A, (2b), 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 $\mathsf{cf}(A)$, (2c).

[1] The symbol "$\equiv$" is a meta-symbol, which is not per se in the vocabulary of $\mathrm{L}_{\mathrm{ar}}^{\lambda}$. We use it to specify orthographical identity between expressions of $\mathrm{L}_{\mathrm{ar}}^{\lambda}$ and definitional notations.

$$\text{John likes Mary's father.} \tag{2a}$$

$$\xrightarrow{\text{render}} A \equiv [like(father\_of(mary))](john) \Rightarrow \ldots \tag{2b}$$

$$\Rightarrow_{\mathsf{cf}} like(f)(j) \text{ where } \{j := john,\ m := mary,\ f := father\_of(m)\} \tag{2c}$$

$$\equiv \mathsf{cf}(A) \tag{2d}$$

There is a rank function for the term (2c), which satisfies the acyclicity condition. For each such rank function, $\mathsf{rank}\, m < \mathsf{rank}(f)$, since m occurs in the term-part $father\_of(m)$ of the assignments $f := father\_of(m)$. 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 $\mathsf{den}(j) = \mathsf{den}(john)$.
Step 2: Compute $\mathsf{den}(m) = \mathsf{den}(mary)$.
Step 3: Compute $\mathsf{den}(f) = \mathsf{den}[father\_of(m)]$.
Step 4: Compute $\mathsf{den}(A) = \mathsf{den}\big[[like(f)](j)\big] = \mathsf{den}\big[[like(\mathsf{den}(f))]\big](\mathsf{den}(j))$

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 [6]. While the detailed reduction steps of the terms A to their canonical and γ-canonical forms are part of the computational attire, we do not include them here, for sake of space limits. They are not essential for understanding the technique of underspecified semantic representation introduced in the paper.

3 Distributions of Multiple Quantifiers

3.1 *Specific Instances of Quantifier Distributions*

We represent the general problem with a sentence like (3) that represents a specific instance of a general problem. E.g., the sentence (3) 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 and corresponding binding of variables filling the argument slots of the relation denoted by the head verb. In a given, specific context, the speaker may intend an interpretation of the sentence S represented by the closed, i.e., *fully specified*, $\mathrm{L}^{\lambda}_{\mathrm{ar}}$-term T_1, with the scope distribution (4b)–(4e).

$$S \equiv \text{Every professor gives some student two papers.} \tag{3}$$

$$S \xrightarrow{\text{render}} T_1 \tag{4a}$$

$$T_1 \equiv every \quad (professor) \tag{4b}$$

$$\big[_3\, \lambda(x_3) some(student) \tag{4c}$$

$$\big[_1\, \lambda(x_1) two(paper) \tag{4d}$$

$$[_2 \lambda(x_2) give(x_1)(x_2)(x_3)]_2\big]_1\big]_3 \tag{4e}$$

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 (5g)–(5k) is obtained by three applications of the (γ) rule, once for $s^1 := \lambda(x_3) student$, and two times for $b^2 := \lambda(x_3)\, \lambda(x_1) paper$.

$$T_1 \Rightarrow_{\mathsf{cf}} every(p)(R_3) \text{ where } \{ \tag{5a}$$

$$R_3 := \lambda(x_3) some(s^1(x_3))(R^1_1(x_3)), \tag{5b}$$

$$R^1_1 := \lambda(x_3)\, \lambda(x_1) two(b^2(x_3)(x_1))(R^2_2(x_3)(x_1)), \tag{5c}$$

$$R^2_2 := \lambda(x_3)\, \lambda(x_1)\, \lambda(x_2) give(x_1)(x_2)(x_3), \tag{5d}$$

$$b^2 := \lambda(x_3)\, \lambda(x_1) paper,\ s^1 := \lambda(x_3) student, \tag{5e}$$

$$p := professor\,\} \qquad \text{by (B-S)} \tag{5f}$$

$$\Rightarrow_{\gamma}{}^{3}\ every(p)(R_3) \text{ where } \{ \tag{5g}$$

$$R_3 := \lambda(x_3) some(s)(R^1_1(x_3)), \tag{5h}$$

$$R^1_1 := \lambda(x_3)\, \lambda(x_1) two(b)(R^2_2(x_3)(x_1)), \tag{5i}$$

$$R^2_2 := \lambda(x_3)\, \lambda(x_1)\, \lambda(x_2) give(x_1)(x_2)(x_3), \tag{5j}$$

$$b := paper,\ s := student,\ p := professor\,\} \tag{5k}$$

Similarly to the specified T_1, (4b)–(4e), depending on context, the sentence S can be rendered to T_2, (6b)–(7e), with a different distribution of quantification.

$$S \xrightarrow{\text{render}} T_2 \tag{6a}$$

$$T_2 \equiv some \quad (student) \tag{6b}$$

$$\big[_1\, \lambda(x_1) every(professor) \tag{6c}$$

$$\big[_3\, \lambda(x_3) two(paper) \tag{6d}$$

$$[_2 \lambda(x_2) give(x_1)(x_2)(x_3)]_2\big]_3\big]_1 \tag{6e}$$

$$
\begin{aligned}
\Rightarrow_{\mathsf{gcf}}\ & \mathit{some}(s)(R_1) \text{ where } \{ && \text{(7a)}\\
& R_1 := \lambda(x_1)\mathit{every}(p)(R_3^1(x_1)), && \text{(7b)}\\
& R_3^1 := \lambda(x_1)\,\lambda(x_3)\mathit{two}(b)(R_2^2(x_1)(x_3)), && \text{(7c)}\\
& R_2^2 := \lambda(x_1)\,\lambda(x_3)\,\lambda(x_2)\mathit{give}(x_1)(x_2)(x_3), && \text{(7d)}\\
& b := \mathit{paper},\ s := \mathit{student},\ p := \mathit{professor}\,\} \quad \text{by 3 times } (\gamma) && \text{(7e)}
\end{aligned}
$$

Note that by using indexed variables corresponding to the order of the argument slots of the constant *give*, i.e., $\mathit{give}(x_1)(x_2)(x_3)$, 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 (3). 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 (3) to a $\mathrm{L}_{\mathrm{ar}}^{\lambda}$-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. For notorious examples, see, e.g., Hobbs and Shieber [5]. This topic continues to be one of the major difficulties in computational semantics and language processing, and here we present a formal approach to it.

3.2 Combinatorial Permutations of Quantifier Scopes

In the major Section 4, we develop technique for representing multiple, alternative terms, each representing a specific quantifier distribution, by a single, underspecified $\mathrm{L}_{\mathrm{ar}}^{\lambda}$-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 $\mathrm{L}_{\mathrm{ar}}^{\lambda}$-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 above two rendering represent the common pattern of the quantificational structure.

Here, we will 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 $\mathrm{L}^{\lambda}_{\mathrm{ar}}$, and in this paper, we use Curry coding of relations with unary functions and corresponding terms denoting them. A $\mathrm{L}^{\lambda}_{\mathrm{ar}}$-term Q denoting an n-ary, generalized quantifier is of type (8a), and we consider the 2-argument quantifiers of type (8b).

$$Q : \big((\widetilde{\sigma}_1 \to \widetilde{\mathsf{t}}) \to \cdots \to \big((\widetilde{\sigma}_n \to \widetilde{\mathsf{t}}) \to \widetilde{\mathsf{t}}\big)\big), \qquad \text{for } n \in \mathbb{N} \tag{8a}$$

$$Q : \big((\widetilde{\mathsf{e}} \to \widetilde{\mathsf{t}}) \to \big((\widetilde{\mathsf{e}} \to \widetilde{\mathsf{t}}) \to \widetilde{\mathsf{t}}\big)\big) \tag{8b}$$

A $\mathrm{L}^{\lambda}_{\mathrm{ar}}$-term Q for a 2-argument, generalized quantifier between individuals of type $\widetilde{\mathsf{e}}$, e.g., a constant *some*, *every*, *two*, etc., 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 $\mathbb{T}_{\widetilde{\mathsf{e}}}$. From the above template examples of quantifier distribution in Section 3.1, 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 3.1, or other syntactic head construction.

Given a permutation $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$, we take recursion variables $Q_i, R_i, q_i, d_i, h \in \mathsf{RecVars}$ that are appropriately typed.

By the extended γ-reduction, see Loukanova [6], the general quantification patterns in terms like (5g)–(5k) and (7e)–(7a) can be reduced to the term Q, (9a)–(9f). In a brief summary, the term Q has congruent forms with respect to renaming the pure variables in the λ-abstracts, as well as the recursion variables bound by the constant where. However, maintaining the indexes provides visualization of linking the quantifier bindings.

$$Q \equiv R_n \ \mathsf{where}\ \{ \tag{9a}$$

$$R^{(n-1)}_{\pi(n)} := \lambda(x_{\pi(1)}) \ldots \lambda(x_{\pi(n)}) h(x_1) \ldots (x_n) \tag{9b}$$

$$R^{(j-1)}_{\pi(j)} := \lambda(x_{\pi(1)}) \ldots \lambda(x_{\pi(j)}) Q_{\pi(j+1)}[\tag{9c}$$

$$\lambda(x_{\pi(j+1)}) R^{j}_{\pi(j+1)}(x_{\pi(1)}) \ldots (x_{\pi(j)})(x_{\pi(j+1)})] \tag{9d}$$

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

$$R_{n+1} := Q_{\pi(1)}[\lambda(x_{\pi(1)}) R_{\pi(1)}(x_{\pi(1)})], \tag{9e}$$

$$Q_i := q_i(d_i) \qquad (\text{for } i = 1, \ldots, n)\} \tag{9f}$$

While the term Q in (9a)–(9f) is underspecified with respect to the free recursion variables $h, q_i, d_i \in \mathsf{FreeV}(Q)$, the order of the relations Q_i, for $i = 1, \ldots, n$, is specified by any given, specific permutation π. 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 $\mathrm{L}^{\lambda}_{\mathrm{ar}}$.

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 properties and the relations, but also on lexical classifications of human languages.

4 Underspecified Scope Distribution

The expression (9a)–(9f) implicitly carries a pattern for underspecified $\mathrm{L}^{\lambda}_{\mathrm{ar}}$-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 3.1 to illustrate the technique. Note that we use extended terms with additional sub-expressions (10e) that add constraints over free recursion variables, as introduced in [10]. 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 (10e) 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 $\pi : \{1, \ldots, n\} \rightarrow \{1, \ldots, n\}$. Loukanova [10] uses another kind of constraints, and the relation between them and the constraints in (10e) 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.

$$U \equiv R_4 \textsf{ where } \{l_1 := Q_1(R_1),\ l_2 := Q_2(R_2),\ l_3 := Q_3(R_3), \tag{10a}$$
$$Q_1 := q_1(d_1),\ Q_2 := q_2(d_2),\ Q_3 := q_3(d_3), \tag{10b}$$
$$q_1 := some,\ q_2 := two,\ q_3 := every, \tag{10c}$$
$$d_1 := student,\ d_2 := paper,\ d_3 := professor,\ h := give\} \tag{10d}$$
$$\textsf{s.t. } \{\ Q_i\ \lambda\text{-binds the } i\text{-th argument of } h \text{ via } R_i, \tag{10e}$$
$$R_4 \text{ is assigned to a closed subterm with fully scope specified } Q_i \text{ (for } i = 1, \ldots, 3),\ \} \tag{10f}$$

Now, from the underspecified (10a)–(10f), we derive one of the possible closed L_{ar}^{λ}-terms, (11a)– (11j), 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 (10e)–(10f). 2. The λ-abstracts are nested within the where-scopes, accordingly, by the dependencies.

$$U_{321} \equiv R_4 \textsf{ where } \{ \tag{11a}$$
$$R_4 := l_3,\ l_3 := Q_3(R_3),\ Q_3 := q_3(d_3), \tag{11b}$$
$$q_3 := every,\ d_3 := professor, \tag{11c}$$
$$R_3 := \lambda(x_3)\Big[_3\, l_2 \textsf{ where } \{_3\, l_2 := Q_2(R_2),\ Q_2 := q_2(d_2), \tag{11d}$$
$$q_2 := two,\ d_2 := paper, \tag{11e}$$
$$R_2 := \lambda(x_2)\big[_2\, l_1 \textsf{ where } \{_2\, l_1 := Q_1(R_1), \tag{11f}$$
$$Q_1 := q_1(d_1), \tag{11g}$$
$$q_1 := some, d_1 := student, \tag{11h}$$
$$R_1 := \lambda(x_1) h(x_1)(x_2)(x_3), \tag{11i}$$
$$h := give\,\}_2\big]_2\,\}_3\Big]_3\,\}\,\} \tag{11j}$$

By using reductions including the important (λ) and (γ) rules, similarly to the ones in Section 3.1, we reduce the term U_{321} in (11a)–(11j), to the γ-canonical form in (12a)–(12g). Note that these reductions use more applications of the (γ) rule, due to the additional assignments in the scope of the λ-abstractions, which are subject to the (λ) rule.

$$\textsf{cf}_{\gamma}(U_{321}) \equiv R_4 \textsf{ where } \{ \tag{12a}$$
$$R_4 := l_3,\ l_3 := Q_3(R_3),\ Q_3 := q_3(d_3), \tag{12b}$$
$$R_3 := \lambda(x_3) l_2^1(x_3),\ l_2^1 := \lambda(x_3) Q_2(R_2^1(x_3)),\ Q_2 := q_2(d_2), \tag{12c}$$
$$R_2^1 := \lambda(x_3)\,\lambda(x_2) l_1^2(x_3)(x_2),\ l_1^2 := \lambda(x_3)\,\lambda(x_2) Q_1(R_1^2(x_3)(x_2)), \tag{12d}$$
$$Q_1 := q_1(d_1),\ R_1^2 := \lambda(x_3)\,\lambda(x_2)\,\lambda(x_1) h(x_1)(x_2)(x_3), \tag{12e}$$
$$q_3 := every,\ d_3 := professor,\ q_2 := two,\ d_2 := paper, \tag{12f}$$
$$q_1 := some,\ d_1 := student,\ h := give\} \tag{12g}$$

Each pair of the first two assignments in (12b), (12c), and (12d) can be merged. Formally, this merging is via extending the reduction calculi of $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ by adding suitable reduction rules, which is not in the subject of this paper. The result is the term S_{321}, (13a)–(13g), that is not algorithmically (step-by-step) equivalent to the terms U_{321}, (11a)–(11j), and $\mathsf{cf}_{\gamma}(U_{321})$, (12a)–(12g), 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}, (13a)–(13g), is more simple, by avoiding the unnecessary computations denoted by the merged assignments. Otherwise, U_{321}, (11a)–(11j), preserves all other computational steps, represented by the assignments.

$$\begin{aligned}
S_{321} \equiv R_4 \ \mathsf{where}\ \{ && \text{(13a)}\\
& R_4 := Q_3(R_3),\ Q_3 := q_3(d_3), && \text{(13b)}\\
& R_3 := \lambda(x_3)Q_2(R^1_2(x_3)),\ Q_2 := q_2(d_2), && \text{(13c)}\\
& R^1_2 := \lambda(x_3)\,\lambda(x_2)Q_1(R^2_1(x_3)(x_2)),\ Q_1 := q_1(d_1), && \text{(13d)}\\
& R^2_1 := \lambda(x_3)\,\lambda(x_2)\,\lambda(x_1)h(x_1)(x_2)(x_3), && \text{(13e)}\\
& q_3 := \mathit{every},\ d_3 := \mathit{professor},\ q_2 := \mathit{two},\ d_2 := \mathit{paper}, && \text{(13f)}\\
& q_1 := \mathit{some},\ d_1 := \mathit{student},\ h := \mathit{give}\,\} && \text{(13g)}
\end{aligned}$$

5 Conclusions and Future Work

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, (9a)–(9f), 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 $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ 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, (4b)–(4e), and T_2, (6b)–(7e). 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 (10a)–(10f). The $\mathrm{L}^{\lambda}_{\mathrm{ar}}$-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 (11a)–(11j) by instantiating the binding scope of the quantifiers. Furthermore, the agent can derive, from U_{321}, the more simple term S_{321}, (13a)–(13g).

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 3, 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 [7]. 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 $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ at its object level. In addition, it is applicable to more general relations.

The technique of Minimal Recursion Semantics (MRS), see Copestake et al. [2], 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., see [1] and [3]. 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 [8, 9] introduces a technique for syntax-semantics interface in computational grammar, which uses $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ 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 by Loukanova and Jiménez-López [11] 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 (10e)–(10f) for linking the quantifiers to the respective argument slots they bind.

References

1. Bender, E.: Deep Linguistic Processing with HPSG (DELPH-IN). http://moin.delph-in.net
2. Copestake, A., Flickinger, D., Pollard, C., Sag, I.: Minimal recursion semantics: an introduction. Research on Language and Computation **3**, 281–332 (2005)
3. Csli linguistic grammars online (lingo) lab at stanford university. http://lingo.stanford.edu
4. Gallin, D.: Intensional and Higher-Order Modal Logic. North-Holland (1975)
5. Hobbs, J.R., Shieber, S.M.: An algorithm for generating quantifier scopings. Computational Linguistics **13**(1–2), 47–63 (1987)
6. Loukanova, R.: γ-Reduction in Type Theory of Acyclic Recursion (to appear)
7. Loukanova, R.: Generalized quantification in situation semantics. In: Gelbukh, A. (ed.) Computational Linguistics and Intelligent Text Processing. Lecture Notes in Computer Science, vol. 2276, pp. 46–57. Springer, Heidelberg (2002)
8. Loukanova, R.: Syntax-semantics interface for lexical inflection with the language of acyclic recursion. In: Bel-Enguix, G., Dahl, V., Jiménez-López, M.D. (eds.) Biology, Computation and Linguistics – New Interdisciplinary Paradigms, Frontiers in Artificial Intelligence and Applications, vol. 228, pp. 215–236. IOS Press, Amsterdam, Berlin, Tokyo, Washington, DC (2011)

9. Loukanova, R.: Semantic information with type theory of acyclic recursion. In: Huang, R., Ghorbani, A.A., Pasi, G., Yamaguchi, T., Yen, N.Y., Jin, B. (eds.) Proceedings of the 8th International Conference on Active Media Technology, AMT 2012, Macau, China, December 4-7, 2012. Lecture Notes in Computer Science, vol. 7669, pp. 387–398. Springer (2012)
10. Loukanova, R.: Algorithmic granularity with constraints. In: Imamura, K., Usui, S., Shirao, T., Kasamatsu, T., Schwabe, L., Zhong, N. (eds.) Brain and Health Informatics. Lecture Notes in Computer Science, vol. 8211, pp. 399–408. Springer International Publishing (2013)
11. Loukanova, R., Jiménez-López, M.D.: 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., Julián, V. (eds.) Highlights on Practical Applications of Agents and Multi-Agent Systems. Advances in Intelligent and Soft Computing, vol. 156, pp. 53–60. Springer, Heidelberg (2012)
12. Moschovakis, Y.N.: The formal language of recursion. The Journal of Symbolic Logic **54**(04), 1216–1252 (1989)
13. Moschovakis, Y.N.: The logic of functional recursion. In: Logic and Scientific Methods, pp. 179–207. Kluwer Academic Publishers. Springer (1997)
14. Moschovakis, Y.N.: A logical calculus of meaning and synonymy. Linguistics and Philosophy **29**, 27–89 (2006)
15. Thomason, R.H. (ed.): Formal Philosophy: Selected Papers of Richard Montague. Yale University Press, New Haven (1974)