Proceedings of the Federated Conference on Computer Science and Information Systems pp. 57–66
DOI: 10.15439/2017F465
ISSN 2300-5963 ACSIS, Vol. 11

Binding Operators in Type-Theory of Algorithms for Algorithmic Binding of Functional Neuro-Receptors

Roussanka Loukanova
Stockholm University
Email: rloukanova@gmail.com

***Abstract*—This paper is on a new approach to mathematics of the notion of algorithm. We extend the higher-order, type-theory of acyclic recursion, i.e., of typed, state-dependent algorithms, which was originally introduced by Moschovakis in [1]. We introduce the concept of recursive λ-binding of argument slots across a sequence of mutually recursive assignments. The primary applications of the extended theory are to computational semantics of formal and natural languages, and to computational neuroscience. We investigate some properties of algorithmic equivalence of functions and relations that bind argument slots of other functions and relations across the recursion operator acting via mutually recursive assignments.**

I. Introduction

THE IDEAS of the new approach to the mathematical notion of algoritm, by a theory of formal languages of functional recursion, were introduced by Moschovakis [2]. The initial steps for extending the approach in [2] to a typed theory $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ of acyclic algorithms, were introduced by Moschovakis in [1]. The theory $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ and its formal language, also denoted by $\mathrm{L}^{\lambda}_{\mathrm{ar}}$, use terms formed under an acyclicity condition restricting the theory to acyclic algorithms that always terminate their calculations after finite number of steps. In addition, $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ uses *currying* coding of functions and relations that have multiple arguments, via sequences of unary functions and corresponding terms denoting them. The idea of such coding was initially given by Gottlob Frege. Later, Shönfinkel re-introduced it by mathematical precision. Then, Curry [3] developed the coding into a fully formalised technique, nowadays popularly named as currying. The type theory $\mathrm{L}^{\lambda}_{\mathrm{r}}$ of algorithms with full recursion, i.e., of algorithms that are not necessarily acyclic, is under development along with $\mathrm{L}^{\lambda}_{\mathrm{ar}}$.

The type theory $\mathrm{L}^{\lambda}_{\mathrm{r}}$, including its sub-theory $\mathrm{L}^{\lambda}_{\mathrm{ar}}$, extends Gallin λ-calculus and its logic TY_2 (see Gallin [4]), in various aspects. Similarly to traditional λ-calculi, $\mathrm{L}^{\lambda}_{\mathrm{r}}$ and $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ employ function application and λ-abstraction for construction of complex terms that denote composite functions with components that can involve other functions. E.g., if f is a constant denoting a unary function, then $\lambda(x)f(x^3)$ is a term denoting another unary function. The theories $\mathrm{L}^{\lambda}_{\mathrm{r}}$ and $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ extend traditional λ-calculi, by adding a specialised recursion operator designated by the constant where. E.g., the formal terms (1b) and (1c) are constructed by using the constant where. The $\mathrm{L}^{\lambda}_{\mathrm{r}}$ terms (1a)–(1c) denote the same function. The term (1c) represent the algorithm for computing the denotation of these terms stepwise. At first, the function that is the denotation of the term $\lambda(x)(x^3)$ is computed, e.g., as a table of argument values and corresponding function values, and saved in the memory slot p. After that, the denotation of $\lambda(x)[f(p(x))]$ is computed by using the data saved in the memory slot p.

$$\lambda(x)f(x^3) \tag{1a}$$

$$\lambda(x)[f(p) \text{ where } \{\, p := x^3 \,\}] \tag{1b}$$

$$\lambda(x)[f(p(x))] \text{ where } \{\, p := \lambda(x)(x^3) \,\} \tag{1c}$$

In this way, $\mathrm{L}^{\lambda}_{\mathrm{r}}$ and $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ extend the expressive power of λ-calculus. Actually, $\mathrm{L}^{\lambda}_{\mathrm{r}}$ is a mathematical theory of the notion of algorithm, which is equivalent to modelling the notion of algorithm, e.g., by Turing machines. The sub-theory $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ models acyclic algorithms, i.e., computations that always end after a finite number of steps. Importantly, this is achieved by the recursion operator where, at the object level of $\mathrm{L}^{\lambda}_{\mathrm{r}}$ for modelling algorithmic computations. The formal theories $\mathrm{L}^{\lambda}_{\mathrm{r}}$ and $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ have reduction calculi, in various versions. By using the standard reduction calculus of $\mathrm{L}^{\lambda}_{\mathrm{r}}$ and $\mathrm{L}^{\lambda}_{\mathrm{ar}}$, the term (1a) can be reduced to (1c) (and even to a more basic term).

The type theory $\mathrm{L}^{\lambda}_{\mathrm{r}}$ represents crucial semantic distinctions in formal and natural languages. We have demonstrated that $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ has major applications to computational semantics and computational syntax-semantics interfaces of human language. The work in this paper is on development of the mathematics of the notion of algorithms by targeting broad applications to Artificial Intelligence and robotics. In Section II, we give an overview of related work on type-theory of situated algorithms and situated information. Primary applications of $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ have been achieved for computational semantics and computational syntax-semantics interfaces of human language. Development of computational syntax-semantics interfaces, by using $\mathrm{L}^{\lambda}_{\mathrm{ar}}$, offers significant steps forward to computational representation of context-dependency and ambiguities in human language. In particular, recursion terms with free recursion variables, i.e., memory variables, represent parametric information and parametric algorithms.

This paper is on theoretical development of $\mathrm{L}^{\lambda}_{\mathrm{r}}$ and $\mathrm{L}^{\lambda}_{\mathrm{ar}}$. Section IV presents the syntax of an extended version $\mathrm{L}^{\lambda}_{\mathrm{raa}}$ of $\mathrm{L}^{\lambda}_{\mathrm{ar}}$, which has terms with components for restrictions. In the major Section V, we focus on some properties of generalised binding operators in the type theory of acyclic recursion

$\mathrm{L}^{\lambda}_{\mathrm{ar}}$. We point out that the results presented in Section V, about the binding operators, hold for the language $\mathrm{L}^{\lambda}_{\mathrm{raa}}$, too. We target applications to computational neuroscience for modelling computational power of neural networks, e.g., as described in Section VI.

II. Related work

By providing the technical notion of binding accros recursive assignments, this paper is directly related to and extends the work in Loukanova [5]. For some more explanations, see the beginning of Section IV.

Terms with restrictions, as in Section V, were originally introduced for the first time in Loukanova [6]. That work is on the formalisation of major notions of algorithmic granularity and algorithmic underspecification defined inherently, at the object level of the languages of the typed theory of recursion $\mathrm{L}^{\lambda}_{\mathrm{r}}$ and $\mathrm{L}^{\lambda}_{\mathrm{ar}}$. Closely related to the work here, the paper [6] introduces two kinds of constraints on possible specifications of underspecified recursion variables by: (1) general acyclicity constraints, and (2) constraints that arise from specific applications. The theory of acyclic recursion is employed to represent semantic ambiguities in human language, which can not be resolved when only partial knowledge is available, even in specific contexts, with specific speakers and their references. The work in [6] takes the direction of formalisation of the notion of algorithmic underspecification carrying constraints, and fine-granularity specifications via syntax-semantics interfaces. For more details on representation of underspecification in semantics of human language, by using the type theory of acyclic recursion $\mathrm{L}^{\lambda}_{\mathrm{ar}}$, see Loukanova [7], [8], [9], [5].

The idea of generalised, restricted parameters were originally, for the first time, introduced by Barwise and Perry [10]. An early, more precise mathematical introduction of restricted parameters was given by Loukanova and Cooper [11], and then by Loukanova [12], [13], [14], [15]. Restricted parameters, as semantic objects, in relational semantic domains of mathematical structures, were presented more officially, i.e., mathematically, in Loukanova [16]. The first introduction of formal language of restricted parameters is given by Loukanova [17], which introduces a higher-order, type-theoretical formal language of information content that is partial, parametric, underspecified, dependent on situations, and recursive. The formal system is extended by Loukanova [18]. While the formal syntax of that language is relational and semantically designates relational semantic structures, it is the first, original formalisation of the semantic concept of generalised, restricted parameters and parametric networks. The terms of that formal language represent situation-theoretic objects. The language has specialised terms for constrained computations by mutual recursion. It introduces terms representing nets of parameters that are simultaneously constrained to satisfy restrictions. The restricted terms presented here in Section V are close in their formal structure to corresponding terms in the formal languages in [17], [18]. In this paper, we limit the formal language and theory to functional structures of typed functions, via Curry coding, see Curry [3].

III. Overview of the Type-Theory of Acyclic Recursion

Here we give a brief overview of L^{λ}_{ar} to facilitate the exposition in the rest of the paper. For details, see Moschovakis [1]. and Loukanova [5], [19].

A. Syntax of L^{λ}_{ar}

a) The set $\mathsf{Types}_{\mathrm{L}^{\lambda}_{\mathrm{ar}}}$ *of* L^{λ}_{ar}*:* is the smallest set defined recursively by the following rules in Backus-Naur form (BNF):

$$\tau :\equiv e \mid t \mid s \mid (\tau_1 \to \tau_2) \tag{2}$$

The type e is for primitive objects that are entities of the semantic domains, as well as for the terms of $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ denoting such entities. The type s is for states consisting of context information, e.g., possible worlds (situations), time and space locations, speakers, listeners; t is the type of the truth values. The type $(\tau_1 \to \tau_2)$ is for functions from objects of type τ_1 to objects of type τ_2. The type (3) is for functions on n-arguments of corresponding types $\tau_1, \ldots, \tau_n$ that take values of type σ, by currying coding.

$$(\tau_1 \to \cdots \to (\tau_n \to \sigma)) \quad \sigma, \tau_i \in \mathsf{Types},\ n \geq 0 \tag{3}$$

The formal language $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ has typed vocabulary. For each type $\tau \in \mathsf{Types}$:

Constants K**:** denumerable set of typed constants

$$K_\tau = \{\mathsf{c}_0{}^\tau, \ldots, \mathsf{c}^\tau_k, \ldots\} \tag{4a}$$

$$K = \bigcup\nolimits_\tau K_\tau \tag{4b}$$

Pure variables PV: denumerable set of typed pure variables

$$\mathsf{PV}_\tau = \{\mathsf{v}_0, \mathsf{v}_1, \ldots\} \tag{5a}$$

$$\mathsf{PV} = \bigcup\nolimits_\tau \mathsf{PV}_\tau \tag{5b}$$

Recursion (memory) variables RV: denumerable set of typed recursion (memory) variables

$$\mathsf{RV}_\tau = \{\mathsf{r}_0, \mathsf{r}_1, \ldots\} \tag{6a}$$

$$\mathsf{RV} = \bigcup\nolimits_\tau \mathsf{RV}_\tau \tag{6b}$$

Variables:

$$\mathsf{Vars}_\tau = \mathsf{PV}_\tau \cup \mathsf{RV}_\tau \tag{7a}$$

$$\mathsf{Vars} = \mathsf{PV} \cup \mathsf{RV} \tag{7b}$$

In addition to the terms the typical λ-calculi, the language L^{λ}_{ar} has new ones formed by using the facility of the recursion, i.e., memory, variables and a new operator for term construction, which we call *recursion operator*, designated by the operator constant where, in infix notation.

The recursive rules for generating the set of L^{λ}_{ar}-terms are given in (8a)–(8e), by using the extended, typed Backus-Naur (TBNF) form, with the assumed types given as superscripts. We also use the typical notation for type assignments: $A : \tau$, to express that A is a term of type τ.

Definition 1. *The set* $\mathsf{Terms}_{\mathrm{L}^{\lambda}_{\mathrm{ar}}}$ *of the terms of* $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ *consists of the expressions generated by the following rules, in Typed Backus-Naur Form (TBNF):*

$$A :\equiv \mathsf{c}^{\tau} : \tau \tag{8a}$$
$$\mid x^{\tau} : \tau \tag{8b}$$
$$\mid B^{(\sigma\to\tau)}(C^{\sigma}) : \tau \tag{8c}$$
$$\mid \lambda(v^{\sigma})(B^{\tau}) : (\sigma \to \tau) \tag{8d}$$
$$\mid A_0^{\sigma} \text{ where } \{\, p_1^{\sigma_1} := A_1^{\sigma_1}, \ldots, p_n^{\sigma_n} := A_n^{\sigma_n} \,\} : \sigma \tag{8e}$$

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

Acyclicity Constraint (AC): *the sequence of assignments* $\{p_1 := A_1, \ldots, p_n := A_n\}$ *is* acyclic *iff there is a function* $\mathsf{rank} : \{p_1, \ldots, p_n\} \longrightarrow \mathbb{N}$ *such that, for all* $p_i, p_j \in \{p_1, \ldots, p_n\}$,

$$\text{if } p_j \text{ occurs freely in } A_i \text{ then } \mathsf{rank}(p_j) < \mathsf{rank}(p_i) \tag{9}$$

Types_{τ} is the set of the terms of type τ, For each $\tau \in$ TYPE.

We call the terms of the form (10) *recursion terms*, or alternatively where-*terms*:

$$[A_0 \text{ where } \{p_1 := A_1, \ldots, p_n := A_n\}] \tag{10}$$

We say that a term A is *explicit* if the constant where does not occur in it.

Notation 1. *We shall use the abbreviation* (11a) *for stated-dependent types sigma, and* (11b) *for state-dependent truth values:*

$$\widetilde{\sigma} \equiv \mathsf{s} \to \sigma \tag{11a}$$
$$\widetilde{\mathsf{t}} \equiv \mathsf{s} \to \mathsf{t} \tag{11b}$$

We may use the following abbreviations and similar variants:

Notation 2.

$$\overrightarrow{p} := \overrightarrow{A} \equiv\ p_1 := A_1,\ \ldots,\ p_n := A_n \quad (n \geq 0) \tag{12a}$$

Notation 3.

$$H(\overrightarrow{x}) \equiv H(x_1)\ldots(x_n) \tag{13}$$
$$\lambda(\overrightarrow{v_j}) \equiv \lambda(v_{j,1}, \ldots, v_{j,l_j}) \equiv \lambda(v_{j,1})\ldots\lambda(v_{j,l_j}) \tag{14}$$

We use the typical notation $\mathbb{N}$ of the set of the natural numbers.

Definition 2 (Immediate terms)**.** *The set of the* immediate terms*, which we denote by* ImT*, is defined as follows:*

Definition 3 (Immediate Terms)**.** *The set* ImT *of immediate terms is defined as follows:*

$$\mathsf{ImT}^{\tau} :\equiv X \mid \tag{15a}$$
$$Y(v_1)\ldots(v_m) \tag{15b}$$
$$\mathsf{ImT}^{(\sigma_1\to\ldots(\sigma_n\to\tau))} :\equiv\ \lambda(u_1)\ldots\lambda(u_n)Y(v_1)\ldots(v_m) \tag{15c}$$

where $n \geq 0$, $m \geq 0$*;* $u_i \in \mathsf{PV}_{\sigma_i}$*, for* $i = 1, \ldots, n$*;* $v_j \in \mathsf{PV}_{\tau_j}$*, for* $j = 1, \ldots, m$*;* $X \in \mathsf{PV}_{\tau}$*,* $Y \in \mathsf{RV}_{(\tau_1\to\ldots\to(\tau_m\to\tau))}$.

Definition 4 (Proper terms)**.** *A term* A *is* proper *if it is not immediate, e.g, the set* PrT *of the proper terms of* $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ *consists of all terms that are not in* ImT*:*

$$\mathsf{PrT} = (\mathsf{Terms} - \mathsf{ImT}) \tag{16}$$

B. Reduction Calculus

a) Reduction Rules:

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

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

Compositionality:

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

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

If $A_i \Rightarrow B_i$, for $i = 0, \ldots, n$, then

$$A_0 \text{ where } \{\, p_1 := A_1, \ldots, p_n := A_n \,\} \Rightarrow B_0 \text{ where } \{\, p_1 := B_1, \ldots, p_n := B_n \,\} \tag{c-r}$$

Head rule:

$$\big(A_0 \text{ where } \{\, \overrightarrow{p} := \overrightarrow{A} \,\}\big) \text{ where } \{\, \overrightarrow{q} := \overrightarrow{B} \,\} \Rightarrow A_0 \text{ where } \{\, \overrightarrow{p} := \overrightarrow{A},\ \overrightarrow{q} := \overrightarrow{B} \,\} \tag{h}$$

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

Bekič-Scott rule:

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

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

Recursion-application rule:

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

given that no p_i occurs freely 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 recursion variable

λ-rule:

$$\lambda(u)(A_0 \text{ where } \{\, p_1 := A_1, \ldots, p_n := A_n \,\}) \Rightarrow \lambda(u)A_0' \text{ where } \{\, p_1' := \lambda(u)A_1', \ldots, p_n' := \lambda(u)A_n' \,\} \tag{λ}$$

where for all $i = 1, \ldots, n$, p_i' is a fresh recursion variable 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.:

$$A_i' \equiv A_i\{\, p_1 :\equiv p_1'(u), \ldots, p_n :\equiv p_n'(u) \,\} \quad \text{for all } i \in \{\, 1, \ldots, n \,\} \tag{20}$$

Definition 5 (Reduction Relation)**.** *The* reduction relation *in* $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ *is the smallest relation, denoted by* $\Rightarrow$*, between terms that is closed under the reduction rules.*

Definition 6 (Term Irreducibility)**.** *We say that a term* $A \in \mathsf{Terms}$ *is* irreducible *if and only if*

$$\textit{for all } B \in \mathsf{Terms}\textit{, if } A \Rightarrow B\textit{, then } A \equiv_c B \tag{21}$$

Here we shall present some of the major results that are essential for algorithmic semantics and which have direct relevance to this paper.

Theorem 1 (Canonical Form Theorem: existence and uniqueness of the canonical forms)**.** *(See Moschovakis [1], § 3.1.) 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.*

Theorem 2 (Referential Synonymy Theorem)**.** *(For the original theorem, see Moschovakis [1]) Two terms* A *and* B *are algorithmically synonymous, i.e., algorithmically equivalent,* $A \approx B$*, if and only if there are explicit, irreducible terms of corresponding types:* $A_0 : \sigma_0, \ldots, A_n : \sigma_n$*, and* $B_0 : \sigma_0, \ldots, B_n : \sigma_n$ $(n \geq 0)$*, such that:*

$$A^{\sigma_0} \Rightarrow_{\mathsf{cf}} A_0^{\sigma_0} \text{ where } \{\, p_1 := A_1^{\sigma_1}, \ldots, p_n := A_n^{\sigma_n} \,\} \tag{22a}$$
$$B^{\sigma_0} \Rightarrow_{\mathsf{cf}} B_0^{\sigma_0} \text{ where } \{\, p_1 := B_1^{\sigma_1}, \ldots, p_n := B_n^{\sigma_n} \,\} \tag{22b}$$

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

$$\mathsf{den}(A_i)(g) = \mathsf{den}(B_i)(g), \quad \textit{for all } g \in G \tag{23}$$

Thus, A and B are algorithmically synonymous, $A \approx B$, if and only if

1) either A and B are proper terms that have the same denotations computed by the same algorithm
2) or A and B are immediate and have the same denotations

When $A \approx B$, we also say that A and B are referentially synonymous, in case we refer to the algorithms they designate.

Theorem 3 (Compositionality Theorem for algorithmic synonymy)**.** *For all* $A \in \mathsf{Terms}_\sigma$*,* $B, C \in \mathsf{Terms}_\tau$*,* $x \in \mathsf{PV}_\tau$*, such that the substitutions* $A\{\, x :\equiv B \,\}$*, and* $A\{\, x :\equiv C \,\}$ *are free, i.e., do not cause variable collisions:*

$$B \approx C \quad \Longrightarrow \quad A\{\, x :\equiv B \,\} \approx A\{\, x :\equiv C \,\} \tag{24}$$

Proof. See Moschovakis [1], § 3.22. □

Corollary 1. *For all explicit, irreducible terms* $A : \sigma$ *and* $B : \sigma$*,*

$$A \approx B \quad \textit{iff} \quad \mathsf{den}(A)(g) = \mathsf{den}(B)(g), \ \textit{for all } g \in G \tag{25}$$

$$\frac{A \approx B}{A \approx B} \tag{27}$$

$$A \approx A \qquad \frac{B \approx A}{A \approx B} \qquad \frac{A \approx B \qquad B \approx C}{A \approx C} \tag{28}$$

$$\frac{A_1 \approx B_1 \qquad A_2 \approx B_2}{A_1(A_2) \approx B_1(B_2)} \qquad \frac{A \approx B}{\lambda(u)A \approx \lambda(u)B} \tag{29}$$

$$\frac{A_0 \approx B_0 \qquad A_1 \approx B_1 \qquad \ldots \qquad A_n \approx B_n}{A_0 \text{ where } \{\, \overrightarrow{p} := \overrightarrow{A} \,\} \approx B_0 \text{ where } \{\, \overrightarrow{q} := \overrightarrow{B} \,\}} \tag{30}$$

$$\frac{\models C = D}{C \approx D}\ (*) \qquad \frac{}{\big(\lambda(u)C\big)(v) \approx C\{u :\equiv v\}}\ (C \text{ e.i.}) \tag{31}$$

where:
"e.i." abbreviates "explicit, irreducible";
(*): C, D are both e.i. terms;
$\models C = D \iff$ for all $g \in G, \mathsf{den}(C)(g) = \mathsf{den}(B)(g)$.
$u, v \in \mathsf{PV}$ and the substitution $C\{u :\equiv v\}$ is free.

TABLE I
THE CALCULUS OF ALGORITHMIC SYNONYMY

Corollary 2. *For every explicit, irreducible term* $A : (\sigma \to \tau)$ *and* $x \in \mathsf{PV}_\sigma$*,* $x : \sigma$*, such that* x *does not occur in* A*:*

$$\lambda(x)\,(A(x)) \approx A \tag{26}$$

IV. Sequential Binders

Loukanova [5] renders sentences of human language, which contain several quantifiers with multiple scope interpretations, into underspecified $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ terms. These terms contain quantifier expressions Q_i, e.g., for $i = 1, 2, 3$, that can have multiple scope distributions over a joint core relation h, depending on context. The common characteristics of such terms is that regardless of the specified scope distribution of the quantifier subterms Q_i, each Q_i binds a fixed argument slot of h, i.e., i-th argument of h.

Recursion terms in canonical forms provide a very sophisticated and elegant representation of scope distributions. They display the common factors across multiple scope distributions corresponding to a given sentence A with several quantifiers. By factoring out the differences, the canonical forms of the $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ terms representing different scopes give a common *underspecified* term that represents the set of all scope distributions for A. Such a term has free recursion variables that can be instantiated to specific scope distributions. The technique is based on formal linking of each of the quantifiers Q_i with the corresponding i-th argument slot of h that it binds, across λ-abstractions, recursion assignments, and reduction steps.

The details of the formalisation of linking the quantifiers to the respective argument slots that they bind across recursive assignments are left open In [5].

The rest of this paper elaborates the formalisation of bindig concepts for a broad class of terms that bind argument slots. The class of these terms include terms denoting quantifiers and other binding relations and functions.

For sake of rigour and clarity, in Theorem 4, we provide detailed assumptions, the formal types (33a)–(33h), and extra

subterms in (32a)-(32e) and (34b)–(34i). These details are important for the proof. They can be ignored for understanding the essence of the theorem. Similarly, we provide such details in other theorems and results presented in this paper.

Theorem 4 (Reduction of Strong Binders 4)**.** *Let T_{m+1} be the term (32a)-(32e):*

$$T_{m+1} \equiv Q_{i_m}\Big[\lambda\, x_{i_m} Q_{i_{(m-1)}}\big[\lambda\, x_{i_{(m-1)}} Q_{i_{(m-2)}}[\tag{32a}$$

$$\lambda\, x_{i_{(m-2)}} Q_{i_{(m-3)}}[\lambda\, x_{i_{(m-3)}} Q_{i_{(m-4)}}[\tag{32b}$$

$$\dots$$

$$\lambda\, x_{i_{(j+1)}} Q_{i_j}[\lambda\, x_{i_j} Q_{i_{(j-1)}}[\lambda\, x_{i_{(j-1)}} Q_{i_{(j-2)}}[\tag{32c}$$

$$\dots$$

$$\lambda\, x_{i_3} Q_{i_2}[\tag{32d}$$

$$\lambda\, x_{i_2} Q_{i_1}[\lambda\, x_{i_1} H(x_1)\dots(x_n)]]]]]]]]\Big] \tag{32e}$$

where we assume that:

- $m, n \in \mathbb{N}$, $m, n \geq 1$
- $x_1, \dots, x_n \in \mathsf{PV}$ *are pure variables of types* σ_i, *i.e.,* $(x_i : \sigma_i)$, *for* $\sigma_i \in \mathsf{Types}$, $i = 1, \dots, n$
- $x_{i_1}, \dots, x_{i_m} \in \mathsf{PV}$ *are pure variables of types* σ_{i_j}, *i.e.,* $(x_{i_j} : \sigma_{i_j})$, *for* $\sigma_{i_j} \in \mathsf{Types}$, $i_j \in \mathbb{N}$, $j = 1, \dots, m$
- Q_{i_1}, …, Q_{i_m}, H *are terms of the corresponding types in (33a)–(33h):*

$$H : (\sigma_1 \to \dots \to (\sigma_{n-1} \to (\sigma_n \to \sigma))) \tag{33a}$$

$$Q_{i_1} : ((\sigma_{i_1} \to \sigma) \to \tau_{i_1}) \tag{33b}$$

$$Q_{i_2} : ((\sigma_{i_2} \to \tau_{i_1}) \to \tau_{i_2}) \tag{33c}$$

$$\dots$$

$$Q_{i_{j-1}} : ((\sigma_{i_{j-1}} \to \tau_{i_{j-2}}) \to \tau_{i_{j-1}}) \tag{33d}$$

$$Q_{i_j} : ((\sigma_{i_j} \to \tau_{i_{j-1}}) \to \tau_{i_j}) \tag{33e}$$

$$Q_{i_{j+1}} : ((\sigma_{i_{j+1}} \to \tau_{i_j}) \to \tau_{i_{j+1}}) \tag{33f}$$

$$\dots$$

$$Q_{i_{m-1}} : ((\sigma_{i_{m-1}} \to \tau_{i_{m-2}}) \to \tau_{i_{m-1}}) \tag{33g}$$

$$Q_{i_m} : ((\sigma_{i_m} \to \tau_{i_{m-1}}) \to \tau_{i_m}) \tag{33h}$$

(The types are such that T_{m+1} in (32a)-(32e) is a well-formed term.)

In addition, assume the following:

(1) H is a proper, i.e., not immediate, term

(2) $m \leq n$

(3) $x_1, \dots, x_n \in \mathsf{PV}$ are pairwise different, pure variables

(4) $x_{i_1}, \dots, x_{i_m} \in \mathsf{PV}$ are pairwise different, pure variables

(5) $\{x_{i_1}, \dots, x_{i_m}\} \subseteq \{x_1, \dots, x_n\}$, i.e., $\{i_1, \dots, i_m\} \subseteq \{1, \dots, n\}$

Then, the term T_{m+1} in (32a)-(32e) can be reduced to the term R_{m+1} in (34b)–(34i):

$$T_{m+1} \Rightarrow R_{m+1} \equiv \tag{34a}$$

$$Q_{i_m}[R^0_{i_m}] \text{ where } \{ \tag{34b}$$

$$R^0_{i_m} := \lambda(x_{i_m}) Q_{i_{(m-1)}}[R^1_{i_{(m-1)}}(x_{i_m})], \tag{34c}$$

$$R^1_{i_{(m-1)}} := \lambda(x_{i_m})\lambda(x_{i_{(m-1)}}) Q_{i_{(m-2)}}[R^2_{i_{(m-2)}}(x_{i_m})(x_{i_{(m-1)}})], \tag{34d}$$

$$R^2_{i_{(m-2)}} := \lambda(x_{i_m})\lambda(x_{i_{(m-1)}})\lambda(x_{i_{(m-2)}}) Q_{i_{(m-3)}}[R^3_{i_{(m-3)}}(x_{i_m})(x_{i_{(m-1)}})(x_{i_{(m-2)}})], \tag{34e}$$

$$\dots$$

$$R^{m-(j+1)}_{i_{(j+1)}} := \lambda(x_{i_m})\dots\lambda(x_{i_{(j+1)}}) Q_{i_j}[R^{m-j}_{i_j}(x_{i_m})\dots(x_{i_{(j+1)}})], \tag{34f}$$

$$R^{m-j}_{i_j} := \lambda(x_{i_m})\dots\lambda(x_{i_{(j+1)}})\lambda(x_{i_j}) Q_{i_{(j-1)}}[R^{m-(j-1)}_{i_{(j-1)}}(x_{i_m})\dots(x_{i_j})], \tag{34g}$$

$$\mid \textit{for } j = m, \dots, 2,$$

$$\dots$$

$$R^{(m-2)}_{i_2} := \lambda(x_{i_m})\dots\lambda(x_{i_2}) Q_{i_1}[R^{(m-1)}_{i_1}(x_{i_m})\dots(x_{i_2})], \tag{34h}$$

$$R^{(m-1)}_{i_1} := \lambda(x_{i_m})\dots\lambda(x_{i_2})\lambda(x_{i_1}) H(\overrightarrow{x}) \} \tag{34i}$$

for some fresh recursion variables $R^l_k \in \mathsf{RV}$ of the types (35a)–(35f):

$$R^{(m-1)}_{i_1} : (\sigma_{i_m} \to \dots \to (\sigma_{i_2} \to (\sigma_{i_1} \to \sigma))\dots) \tag{35a}$$

$$R^{(m-2)}_{i_2} : (\sigma_{i_m} \to \dots \to (\sigma_{i_2} \to \tau_{i_1})\dots) \tag{35b}$$

$$\dots$$

$$R^{m-j}_{i_j} : (\sigma_{i_m} \to \dots \to (\sigma_{i_{j+1}} \to (\sigma_{i_j} \to \tau_{i_{j-1}}))\dots) \tag{35c}$$

$$R^{m-(j+1)}_{i_{(j+1)}} : (\sigma_{i_m} \to \dots \to (\sigma_{i_{j+1}} \to \tau_{i_j})\dots) \tag{35d}$$

$$\dots$$

$$R^1_{i_{(m-1)}} : (\sigma_{i_m} \to (\sigma_{i_{m-1}} \to \tau_{i_{m-2}})) \tag{35e}$$

$$R^0_{i_m} : (\sigma_{i_m} \to \tau_{i_{m-1}}) \tag{35f}$$

Proof. The proof is by induction on the number of the terms $Q_{i_1}, \dots, Q_{i_m}$. It uses the reduction rules of $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ (and $\mathrm{L}^{\lambda}_{\mathrm{r}}$) and verifies the types.

The superscripts of the variables $R^{m-j}_{i_j}$ are counters of the number of the applications of the λ-rule (λ). For sake of space, we do not include the proof here. □

Note 1. *The types of the terms $Q_{i_1}, \dots, Q_{i_m}$, H do not need to be such that $Q_{i_1}, \dots, Q_{i_m}$ can denote quantifiers over arguments of a range denoted by H, which are in the focus of the work in Loukanova [5]. In this paper, we investigate the broader class of terms $Q_{i_1}, \dots, Q_{i_m}$, H, such that $Q_{i_1}, \dots, Q_{i_m}$ can bind argument slots of the term H.*

Note 2. *The requirements (4)–(5) in Theorem 4 guarantee that there is binding of existing argument slots of H. I.e., the bindings in the term T_{m+1} in (32a)-(32e) are strong, not vacuous. Therefore, the chained bindings by R_{m+1} in (34b)–(34i) are strong too. The term H may still denote a function that does not depend essentially on some of its arguments, including such that are bound by some Q_{i_j}. The requirements (4)–(5) in Theorem 4 can be removed in a general term T_{m+1} of the same form (32a)-(32e), while the reduction to the term R_{m+1} in (34b)–(34i) holds.*

Theorem 5. *The term R_{m+1} in (34b)–(34i) is algorithmically synonymous (equivalent) with the term R'_{m+1} in (36b)–(36i).*

$$R_{m+1} \approx R'_{m+1} \equiv \tag{36a}$$
$$Q_{i_m}[\lambda(x_{i_m})R^0_{i_m}(x_{i_m})] \text{ where } \{ \tag{36b}$$
$$R^0_{i_m} := \lambda(x_{i_m})Q_{i_{(m-1)}}[\lambda(x_{i_{(m-1)}})R^1_{i_{(m-1)}}(x_{i_m})(x_{i_{(m-1)}})], \tag{36c}$$
$$R^1_{i_{(m-1)}} := \lambda(x_{i_m})\lambda(x_{i_{(m-1)}})Q_{i_{(m-2)}}[\lambda(x_{i_{(m-2)}})R^2_{i_{(m-2)}}(x_{i_m})(x_{i_{(m-1)}})(x_{i_{(m-2)}})], \tag{36d}$$
$$R^2_{i_{(m-2)}} := \lambda(x_{i_m})\lambda(x_{i_{(m-1)}})\lambda(x_{i_{(m-2)}})Q_{i_{(m-3)}}[\lambda(x_{i_{(m-3)}})R^3_{i_{(m-3)}}(x_{i_m})(x_{i_{(m-1)}})(x_{i_{(m-2)}})(x_{i_{(m-3)}})], \tag{36e}$$
$$\dots$$
$$R^{m-(j+1)}_{i_{(j+1)}} := \lambda(x_{i_m})\dots\lambda(x_{i_{(j+1)}})Q_{i_j}[\lambda(x_{i_j})R^{m-j}_{i_j}(x_{i_m})\dots(x_{i_{(j+1)}})(x_{i_j})], \tag{36f}$$
$$R^{m-j}_{i_j} := \lambda(x_{i_m})\dots\lambda(x_{i_{(j+1)}})\lambda(x_{i_j})Q_{i_{(j-1)}}[\lambda(x_{i_{(j-1)}})R^{m-(j-1)}_{i_{(j-1)}}(x_{i_m})\dots (x_{i_j})(x_{i_{(j-1)}})], \tag{36g}$$
$$\mid \textit{for } j = m, \dots, 2,$$
$$\dots$$
$$R^{(m-2)}_{i_2} := \lambda(x_{i_m})\dots\lambda(x_{i_2})Q_{i_1}[\lambda(x_{i_1}) R^{(m-1)}_{i_1}(x_{i_m})\dots(x_{i_2})(x_{i_1})], \tag{36h}$$
$$R^{(m-1)}_{i_1} := \lambda(x_{i_m})\dots\lambda(x_{i_1})H(\overrightarrow{x})\ \} \tag{36i}$$

Proof. For every $i_j \in \{ i_1, \dots, i_m \}$, from (35c), we have that $R^{m-j}_{i_j} \in \mathsf{RV}$:

$$R^{m-j}_{i_j}(x_{i_m})\dots(x_{i_{(j+1)}}) : (\sigma_{i_j} \to \tau_{i_{j-1}}) \tag{37a}$$
$$\therefore\ R^{m-j}_{i_j}(x_{i_m})\dots(x_{i_{(j+1)}})(x_{i_j}) : \tau_{i_{j-1}} \tag{37b}$$
$$\therefore \lambda(x_{i_j})R^{m-j}_{i_j}(x_{i_m})\dots(x_{i_{(j+1)}})(x_{i_j}) : (\sigma_{i_j} \to \tau_{i_{j-1}}) \tag{37c}$$

Since $R_{i_j} \in \mathsf{RV}$, the terms $R^{m-j}_{i_j}(x_{i_m})\dots(x_{i_{(j+1)}})(x_{i_j})$ and $\lambda(x_{i_j})R^{m-j}_{i_j}(x_{i_m})\dots(x_{i_{(j+1)}})(x_{i_j})$ are immediate, and thus explicite, irreducible. Furthermore, for all $g \in G$:

$$\mathsf{den}(R^{m-j}_{i_j}(x_{i_m})\dots(x_{i_{(j+1)}}))(g) = \mathsf{den}(\lambda(x_{i_j})R^{m-j}_{i_j}(x_{i_m})\dots(x_{i_{(j+1)}})(x_{i_j}))(g) \tag{38}$$

By Corollary 2, it follows that:

$$R^{m-j}_{i_j}(x_{i_m})\dots(x_{i_{(j+1)}}) \approx \lambda(x_{i_j})R^{m-j}_{i_j}(x_{i_m})\dots(x_{i_{(j+1)}})(x_{i_j}) \quad \text{for every } i_j \in \{ i_1, \dots, i_m \} \tag{39}$$

From (39), by using the rules for algorithmic synonymy in Table I, it follows that

$$R_{m+1} \approx R'_{m+1} \tag{40}$$

Thus, the terms R_{m+1} and R'_{m+1} are algorithmically equivalent. □

Definition 7 (Recursive Distance)**.** *Let T be the a term of the form:*

$$T \equiv A_0 \text{ where } \{ p_n := A_n, \dots, \tag{41a}$$
$$p_{i+1} := A_{i+1},\ p_i := A_i, \dots, \tag{41b}$$
$$p_1 := A_1, \ \} \tag{41c}$$

The recursive distance $\mathsf{Rdist}(p_n, H, A_1) = \mathsf{Rdist}(p_n, H, p_1) = \mathsf{Rdist}(A_n, H, A_1) = \mathsf{Rdist}(A_n, H, p_1)$ *of A_n (or its p_n), from a subterm H of a term A_1 (or its recursion memory p_1), in a recursion term T of the form (41a)–(41c) (modulo congruence with respect to the order of the assignments), is defined by induction:*

$$\mathsf{Rdist}(p_i, H, A_i) = \mathsf{Rdist}(A_i, H, A_i) = \mathsf{Rdist}(p_i, H, p_i) = 0, \quad \textit{if } H \textit{ occurs in } A_i \tag{42a}$$

$$\mathsf{Rdist}(p_{i+1}, H, A_1) = \mathsf{Rdist}(A_{i+1}, H, A_1) = \mathsf{Rdist}(p_{i+1}, H, p_1) = \mathsf{Rdist}(A_{i+1}, H, p_1) = \min\{ \mathsf{Rdist}(p_i, H, p_1) \mid \ p_i \textit{ occurs in } A_{i+1} \} + 1, \tag{42b}$$

Note: $\mathsf{Rdist}(p_n, H, A_1)$ is a partial function.

Theorem 6 (Binding Across Recursion 6)**.** *Let R_{m+1} be a term of the form (43b)–(43h). as in Theorem 5.*

$$R_{m+1} \equiv \tag{43a}$$
$$Q_{i_m}[\lambda(x_{i_m})R^0_{i_m}(x_{i_m})] \text{ where } \{ \tag{43b}$$
$$R^0_{i_m} := \lambda(x_{i_m})Q_{i_{(m-1)}}[\lambda(x_{i_{(m-1)}})R^1_{i_{(m-1)}}(x_{i_m})(x_{i_{(m-1)}})], \tag{43c}$$
$$R^1_{i_{(m-1)}} := \lambda(x_{i_m})\lambda(x_{i_{(m-1)}})Q_{i_{(m-2)}}[\lambda(x_{i_{(m-2)}})R^2_{i_{(m-2)}}(x_{i_m})(x_{i_{(m-1)}})(x_{i_{(m-2)}})], \tag{43d}$$
$$\dots$$
$$R^{m-(j+1)}_{i_{(j+1)}} := \lambda(x_{i_m})\dots\lambda(x_{i_{(j+1)}})Q_{i_j}[\lambda(x_{i_j})R^{m-j}_{i_j}(x_{i_m})\dots(x_{i_{(j+1)}})(x_{i_j})], \tag{43e}$$
$$R^{m-j}_{i_j} := \lambda(x_{i_m})\dots\lambda(x_{i_{(j+1)}})\lambda(x_{i_j})Q_{i_{(j-1)}}[\lambda(x_{i_{(j-1)}})R^{m-(j-1)}_{i_{(j-1)}}(x_{i_m})\dots (x_{i_j})(x_{i_{(j-1)}})], \tag{43f}$$
$$\mid \textit{for } j = m, \dots, 2,$$
$$\dots$$
$$R^{(m-2)}_{i_2} := \lambda(x_{i_m})\dots\lambda(x_{i_2})Q_{i_1}[\lambda(x_{i_1}) R^{(m-1)}_{i_1}(x_{i_m})\dots(x_{i_2})(x_{i_1})], \tag{43g}$$
$$R^{(m-1)}_{i_1} := \lambda(x_{i_m})\dots\lambda(x_{i_1})H(x_1)\dots(x_n), \tag{43h}$$
$$\overrightarrow{p} := \overrightarrow{A}\ \} \tag{43i}$$

Then, for every $i_j \in \{ i_1, \ldots, i_m \}$, (1) the term Q_{i_j} binds the i_j-th argument slot of H, and (2) $R_{i_j}^{m-j}$ is λ-abstraction $\lambda(x_{i_m}) \ldots \lambda(x_{i_{(j+1)}})\lambda(x_{i_j})$ over the i_m-th, $\ldots$, i_j-th argument slots of H in this specific order.

Proof. The proof is by induction on $j = 1, \ldots, m$, i.e., on the recursive distance of $R_{i_j}^{m-j}$ from H in $R_{i_1}^{m-1}$. □

Theorem 7 (Binding Across Recursion 7)**.** *Let R_{m+1} be a term of the form (44b)–(44h), as in Theorem 4:*

$$R_{m+1} \equiv \tag{44a}$$
$$Q_{i_m}[R_{i_m}^0] \text{ where } \{ \tag{44b}$$
$$R_{i_m}^0 := \lambda(x_{i_m}) Q_{i_{(m-1)}}[R_{i_{(m-1)}}^1(x_{i_m})], \tag{44c}$$
$$R_{i_{(m-1)}}^1 := \lambda(x_{i_m})\lambda(x_{i_{(m-1)}}) Q_{i_{(m-2)}}[R_{i_{(m-2)}}^2(x_{i_m})(x_{i_{(m-1)}})], \tag{44d}$$
$$\ldots$$
$$R_{i_{(j+1)}}^{m-(j+1)} := \lambda(x_{i_m}) \ldots \lambda(x_{i_{(j+1)}}) Q_{i_j}[R_{i_j}^{m-j}(x_{i_m}) \ldots (x_{i_{(j+1)}})], \tag{44e}$$
$$R_{i_j}^{m-j} := \lambda(x_{i_m}) \ldots \lambda(x_{i_{(j+1)}})\lambda(x_{i_j}) Q_{i_{(j-1)}}[R_{i_{(j-1)}}^{m-(j-1)}(x_{i_m}) \ldots (x_{i_j})], \tag{44f}$$
$$\mid \textit{for } j = m, \ldots, 2,$$
$$\ldots$$
$$R_{i_2}^{(m-2)} := \lambda(x_{i_m}) \ldots \lambda(x_{i_2}) Q_{i_1}[R_{i_1}^{(m-1)}(x_{i_m}) \ldots (x_{i_2})], \tag{44g}$$
$$R_{i_1}^{(m-1)} := \lambda(x_{i_m}) \ldots \lambda(x_{i_2})\lambda(x_{i_1}) H(\overrightarrow{x}) \tag{44h}$$
$$\overrightarrow{p} := \overrightarrow{A} \} \tag{44i}$$

Then, for every $i_j \in \{ i_1, \ldots, i_m \}$, (1) Q_{i_j} binds the i_j-th argument slot of H, and (2) $R_{i_j}^{m-j}$ is λ-abstraction $\lambda(x_{i_m}) \ldots \lambda(x_{i_{(j+1)}})\lambda(x_{i_j})$ over the i_m-th, $\ldots$, i_j-th argument slots of H in this specific order.

Proof. The proof is by induction on $j = 1, \ldots, m$, i.e., on the recursive distance of $R_{i_j}^{m-j}$ from H in $R_{i_1}^{m-1}$. □

Now, we show that Theorem 4 holds by weakening the requirement (3).

Theorem 8 (Reduction of Strong Binders 8)**.** *Let T_{m+1} be the term (45a)-(45e):*

$$T_{m+1} \equiv Q_{i_m}\Big[\lambda\, x_{i_m} Q_{i_{(m-1)}}\big[\lambda\, x_{i_{(m-1)}} Q_{i_{(m-2)}}[\tag{45a}$$
$$\lambda\, x_{i_{(m-2)}} Q_{i_{(m-3)}}[\lambda\, x_{i_{(m-3)}} Q_{i_{(m-4)}}[\tag{45b}$$
$$\ldots$$
$$\lambda\, x_{i_{(j+1)}} Q_{i_j}[\lambda\, x_{i_j} Q_{i_{(j-1)}}[\lambda\, x_{i_{(j-1)}} Q_{i_{(j-2)}}[\tag{45c}$$
$$\ldots$$
$$\lambda\, x_{i_3} Q_{i_2}[\tag{45d}$$
$$\lambda\, x_{i_2} Q_{i_1}[\lambda\, x_{i_1} H(v_1) \ldots (v_k)]]]]]]]]\Big] \tag{45e}$$

where we assume that:

- $m, n \in \mathbb{N}$, $m, n \geq 1$
- $x_{i_1}, \ldots, x_{i_m} \in \mathsf{PV}$ *are pure variables of types* $(x_{i_j} : \sigma_{i_j})$, *for* $\sigma_{i_j} \in \mathsf{Types}$, $i_j \in \mathbb{N}$, $j = 1, \ldots, m$
- Q_{i_1}, $\ldots$, Q_{i_m}, H *are terms of the corresponding types in (33a)–(33h):*

In addition, assume the following:

(1) H is a proper, i.e., not immediate, term
(2) $m \leq n$
(3) $v_1, \ldots, v_k \in \mathsf{PV}$ are pure variables, not necessarily pairwise different.
(4) $x_{i_1}, \ldots, x_{i_m} \in \mathsf{PV}$ are pairwise different, pure variables
(5) $\{ x_{i_1}, \ldots, x_{i_m} \} \subseteq \{ v_1, \ldots, v_k \}$, i.e., $\{ i_1, \ldots, i_m \} \subseteq \{ 1, \ldots, n \}$

Then, the term T_{m+1} in (45a)-(45e) can be reduced to the term R_{m+1} in (46a)–(46i)

$$T_{m+1} \Rightarrow R_{m+1} \equiv \tag{46a}$$
$$Q_{i_m}[R_{i_m}^0] \text{ where } \{ \tag{46b}$$
$$R_{i_m}^0 := \lambda(x_{i_m}) Q_{i_{(m-1)}}[R_{i_{(m-1)}}^1(x_{i_m})], \tag{46c}$$
$$R_{i_{(m-1)}}^1 := \lambda(x_{i_m})\lambda(x_{i_{(m-1)}}) Q_{i_{(m-2)}}[R_{i_{(m-2)}}^2(x_{i_m})(x_{i_{(m-1)}})], \tag{46d}$$
$$R_{i_{(m-2)}}^2 := \lambda(x_{i_m})\lambda(x_{i_{(m-1)}})\lambda(x_{i_{(m-2)}}) Q_{i_{(m-3)}}[R_{i_{(m-3)}}^3(x_{i_m})(x_{i_{(m-1)}})(x_{i_{(m-2)}})], \tag{46e}$$
$$\ldots$$
$$R_{i_{(j+1)}}^{m-(j+1)} := \lambda(x_{i_m}) \ldots \lambda(x_{i_{(j+1)}}) Q_{i_j}[R_{i_j}^{m-j}(x_{i_m}) \ldots (x_{i_{(j+1)}})], \tag{46f}$$
$$R_{i_j}^{m-j} := \lambda(x_{i_m}) \ldots \lambda(x_{i_{(j+1)}})\lambda(x_{i_j}) Q_{i_{(j-1)}}[R_{i_{(j-1)}}^{m-(j-1)}(x_{i_m}) \ldots (x_{i_j})], \tag{46g}$$
$$\mid \textit{for } j = m, \ldots, 1,$$
$$\ldots$$
$$R_{i_2}^{(m-2)} := \lambda(x_{i_m}) \ldots \lambda(x_{i_2}) Q_{i_1}[R_{i_1}^{(m-1)}(x_{i_m}) \ldots (x_{i_2})], \tag{46h}$$
$$R_{i_1}^{(m-1)} := \lambda(x_{i_m}) \ldots \lambda(x_{i_2})\lambda(x_{i_1}) H(\overrightarrow{v}) \} \tag{46i}$$

for some fresh recursion variables $R_k^l \in \mathsf{RV}$ of the types (35a)–(35f).

Proof. The proof is similar to that of Theorem 4. □

Similarly, Theorem 5 holds by weakening the requirement (3), but we do not present it here for sake of space.

Theorem 9 (Reduction of Strong Binders 9)**.** *Let T_{m+1} be the term* [1] *(47a)-(47e):*

$$T_{m+1} \equiv Q_{i_m}\Big[\lambda\, x_{i_m} Q_{i_{(m-1)}}\big[. \lambda\, x_{i_{(m-1)}} Q_{i_{(m-2)}}[\tag{47a}$$

[1] The difference from Theorem 8 is that now H is immediate, i.e.., not proper

$$\lambda\, x_{i_{(m-2)}} Q_{i_{(m-3)}} [\lambda\, x_{i_{(m-3)}} Q_{i_{(m-4)}} [\tag{47b}$$

...

$$\lambda\, x_{i_{(j+1)}} Q_{i_j} [\lambda\, x_{i_j} Q_{i_{(j-1)}} [\lambda\, x_{i_{(j-1)}} Q_{i_{(j-2)}} [\tag{47c}$$

...

$$\lambda\, x_{i_3} Q_{i_2} [\tag{47d}$$

$$\lambda\, x_{i_2} Q_{i_1} [\lambda\, x_{i_1} H(v_1) \ldots (v_k)]]]]]]]]] \Big] \tag{47e}$$

where we assume that:

- $m, n \in \mathbb{N},\ m, n \geq 1$
- $x_{i_1}, \ldots, x_{i_m} \in \mathit{PV}$ *are pure variables of types* $(x_{i_j} : \sigma_{i_j})$, *for* $\sigma_{i_j} \in \mathsf{Types},\ i_j \in \mathbb{N},\ j = 1, \ldots, m$
- $Q_{i_1}, \ldots, Q_{i_m}, H$ *are terms of the corresponding types in* (33a)–(33h)*:*

In addition, assume the following:

(1) H *is an immediate, term*
(2) $m \leq n$
(3) $v_1, \ldots, v_k \in \mathit{PV}$ *are pure variables, not necessarily pairwise different.*
(4) $x_{i_1}, \ldots, x_{i_m} \in \mathit{PV}$ *are pairwise different, pure variables*
(5) $\{ x_{i_1}, \ldots, x_{i_m} \} \subseteq \{ v_1, \ldots, v_k \}$, *i.e.,* $\{ i_1, \ldots, i_m \} \subseteq \{ 1, \ldots, n \}$

Then, the term T_{m+1} *in* (47a)-(47e) *can be reduced to the term* R_{m+1} *in* (48a)–(48h)*:*

$$T_{m+1} \Rightarrow R_{m+1} \equiv \tag{48a}$$

$$Q_{i_m}[R^0_{i_m}] \text{ where } \{ \tag{48b}$$

$$R^0_{i_m} := \lambda(x_{i_m}) Q_{i_{(m-1)}} [R^1_{i_{(m-1)}}(x_{i_m})], \tag{48c}$$

$$R^1_{i_{(m-1)}} := \lambda(x_{i_m}) \lambda(x_{i_{(m-1)}}) Q_{i_{(m-2)}} [R^2_{i_{(m-2)}}(x_{i_m})(x_{i_{(m-1)}})], \tag{48d}$$

$$R^2_{i_{(m-2)}} := \lambda(x_{i_m}) \lambda(x_{i_{(m-1)}}) \lambda(x_{i_{(m-2)}}) Q_{i_{(m-3)}} [R^3_{i_{(m-3)}}(x_{i_m})(x_{i_{(m-1)}})(x_{i_{(m-2)}})], \tag{48e}$$

...

$$R^{m-(j+1)}_{i_{(j+1)}} := \lambda(x_{i_m}) \ldots \lambda(x_{i_{(j+1)}}) Q_{i_j} [R^{m-j}_{i_j}(x_{i_m}) \ldots (x_{i_{(j+1)}})], \tag{48f}$$

$$R^{m-j}_{i_j} := \lambda(x_{i_m}) \ldots \lambda(x_{i_{(j+1)}}) \lambda(x_{i_j}) Q_{i_{(j-1)}} [R^{m-(j-1)}_{i_{(j-1)}}(x_{i_m}) \ldots (x_{i_j})], \tag{48g}$$

$$\mid \textit{for } j = m, \ldots, 1,$$

...

$$R^{(m-2)}_{i_2} := \lambda(x_{i_m}) \ldots \lambda(x_{i_2}) Q_{i_1} [\lambda(x_{i_1}) H(v_1) \ldots (v_k)] \tag{48h}$$

$$\} \tag{48i}$$

for some fresh recursion variables $R^l_k \in \mathit{RV}$ *of the types* (35a)–(35f)*.*

Proof. The proof is similar to that of Theorem 8. □

A result as in Theorem 5 corresponding to Theorem 9 can be obtained for any immediate term H, but we do not present it here for sake of space. Theorem 4, 8, 9 still hold after removing the requirements (4)–(5), by allowing vacuous binding.

V. Algorithmic Computations with Restrictions

In this section, we give a very brief introduction of an extension of the formal language $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ to a formal language $\mathrm{L}^{\lambda}_{\mathrm{raa}}$ of Type Theory of Restricted Acyclic Algorithms (TTofRAA). If the acyclicity constraint AC (1) is dropped out, $\mathrm{L}^{\lambda}_{\mathrm{r}}$ is extended to a language $\mathrm{L}^{\lambda}_{\mathrm{ra}}$ of typed, restricted algorithms with full recursion. The language and theory $\mathrm{L}^{\lambda}_{\mathrm{ra}}$ is a mathematical model of the notion of algorithm, while $\mathrm{L}^{\lambda}_{\mathrm{raa}}$ models the notion of acyclic algorithm. In this paper, we focus on giving the syntax of $\mathrm{L}^{\lambda}_{\mathrm{raa}}$, with some intuitive explanations, for an introduction to potential applications of the results in the previous section to model neural connections as result of binding functionality of receptors.

The denotational and algorithmic semantics of $\mathrm{L}^{\lambda}_{\mathrm{raa}}$ ($\mathrm{L}^{\lambda}_{\mathrm{ra}}$), the reduction calculus, and various theoretical results are subject of other, forthcoming work. The language $\mathrm{L}^{\lambda}_{\mathrm{raa}}$ ($\mathrm{L}^{\lambda}_{\mathrm{ra}}$) is the result of extending $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ by adding a constant such that and corresponding terms having a component for restrictions, i.e., constraints. Such terms with restrictions designate a restrictor over computations as an operator. The restrictor operator, in combination with the recursion operator designated by where-terms, model algorithmic systems that are equipped with memory cells. Results of computations are storred in the memory cells. Memory cells are typed, i.e., they can hold only data of the respective type. In addition, some of the computations and the memory cells are restricted by propositional conditions. The propositional restrictions can represent properties of objects that have to be fulfilled. Our purpose is to use $\mathrm{L}^{\lambda}_{\mathrm{raa}}$ and $\mathrm{L}^{\lambda}_{\mathrm{ra}}$ for modelling functionality of biological entities that have valences or receptors for binding across sequences of structural entities with binding connections, such as neurons and neural networks.

The formal language $\mathrm{L}^{\lambda}_{\mathrm{raa}}$ has the same set of types as $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ defined in (2), i.e., $\mathsf{Types}_{\mathrm{L}^{\lambda}_{\mathrm{raa}}} = \mathsf{Types}_{\mathrm{L}^{\lambda}_{\mathrm{ar}}}$. The vocabulary of $\mathrm{L}^{\lambda}_{\mathrm{raa}}$ is the same as that of $\mathrm{L}^{\lambda}_{\mathrm{ar}}$, i.e., $\mathrm{L}^{\lambda}_{\mathrm{raa}}$ has typed constants and two kinds of typed variables: *pure variables*, for λ-abstraction operator; and, *memory (recursion) variables*, for storing information. The set $\mathsf{Terms}_{\mathrm{L}^{\lambda}_{\mathrm{raa}}}$ of the terms of $\mathrm{L}^{\lambda}_{\mathrm{raa}}$ are defined by adding one more structural rule for the operator constant such that to (8a)–(8e).

Definition 8. *The set* $\mathsf{Terms}_{\mathrm{L}^{\lambda}_{\mathrm{raa}}}$ *of the terms of* $\mathrm{L}^{\lambda}_{\mathrm{raa}}$ *consists of the expressions generated by the following rules in TBNF:*

$$A :\equiv \mathsf{c}^{\tau} : \tau \tag{49a}$$

$$\mid x^{\tau} : \tau \tag{49b}$$

$$\mid B^{(\sigma \to \tau)}(C^{\sigma}) : \tau \tag{49c}$$

$$\mid \lambda(v^{\sigma})\,(B^{\tau}) : (\sigma \to \tau) \tag{49d}$$

$$\mid \big(A^{\sigma_0}_0 \text{ where } \{ p^{\sigma_1}_1 := A^{\sigma_1}_1, \ldots, p^{\sigma_n}_n := A^{\sigma_n}_n \}\big) : \sigma_0 \tag{49e}$$

$$\mid \big(\mathsf{A}^{\sigma_0}_0 \text{ such that } \{ \mathsf{C}^{\tau_1}_1, \ldots, \mathsf{C}^{\tau_m}_m \}\big) : \sigma_0 \tag{49f}$$

given that: c is a constant; x is a variable of ether kind; $A_1 : \sigma_1, \ldots, A_n : \sigma_n$, $\mathsf{C}_1 : \tau_1, \ldots, \mathsf{C}_m : \tau_m \in \mathsf{Terms}_{\mathrm{L}^{\lambda}_{\mathrm{raa}}}$; $p_i : \sigma_i$, $i = 1, \ldots, n$ $(n \geq 0)$ in (49e) are pairwise different recursion variables of respective types, such that the sequence of assignments $\{ p_1^{\sigma_1} := A_1^{\sigma_1}, \ldots, p_n^{\sigma_n} := A_n^{\sigma_n} \}$ satisfies the Acyclicity Constraint (AC); and each τ_i is either the type t of truth values, or the type $\widetilde{\mathsf{t}}$ of state dependent truth values (see (11b)).

By extending the reduction calculus of $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ with a rule for reducing terms with restrictions of the form (49f), the results in Section III and Section IV hold for the extended language $\mathrm{L}^{\lambda}_{\mathrm{raa}}$ of restricted algorithms. These properties are not in the subject of this paper because they require extensive mathematical work. They will be in a forthcoming paper devoted to them.

The rules (49a)–(49f) applied recursively provide very expressive formal terms that represent algorithmic computations that end after finite number of computational steps, because of the Acyclicity Constraint (AC).

In particular, combination of the rules (49e) and (49f), gives terms of recursion and restrictors of the forms (50a)–(50b) and (51a)–(51b):

$$\big(\big[A_0^{\sigma_0} \ \mathsf{where} \ \{ p_1^{\sigma_1} := A_1^{\sigma_1}, \ldots, p_n^{\sigma_n} := A_n^{\sigma_n} \} \big] \tag{50a}$$

$$\mathsf{such\ that} \ \{ \mathsf{C}_1^{\tau_1}, \ldots, \mathsf{C}_m^{\tau_m} \} \big) \tag{50b}$$

$$\big(\big[A_0^{\sigma_0} \ \mathsf{such\ that} \ \{ \mathsf{C}_1^{\tau_1}, \ldots, \mathsf{C}_m^{\tau_m} \} \big] \tag{51a}$$

$$\mathsf{where} \ \{ p_1^{\sigma_1} := A_1^{\sigma_1}, \ldots, p_n^{\sigma_n} := A_n^{\sigma_n} \} \big) \tag{51b}$$

VI. Modelling Algorithmic Neural Networks

A. Procedural and Declerative Neural Networks

We present the use of the recursion terms with constraints for modelling neural networks.

Neural systems (in peripheral and central nervous systems) of living organisms, even as simple as Drosophila melanogaster, have innate faculty of both procedural and declarative memory, see, e.g., Kandel et al. [20] and Squire and Kandel [21].

a) Neural Networks of Procedural Memory: Here, we propose to model procedural memory by terms having recursive assignments of the form (49e), while employing the entire range of term forms (49a)–(49f).

The systems of assignments in terms of the form (49e) represent mutually recursive computations of the denotations of the terms $A_i^{\sigma_i}$, which are saved in the corresponding memory cells p_i. Procedural memory is modelled via the assignments $p_i^{\sigma_i} := A_i^{\sigma_i}$. The system of mutually recursive assignments (52b) models the following fundamental phenomena of functional, procedural neural networks:

1) The collection (52a) is a recursively linked network of memory cells $p_i : \sigma_i$ of corresponding types
2) The system (52b) has algorithmic, i.e., procedural, nature of a network of memory cells p_i:
   Under completion of the computation of the data $A_i^{\sigma_i}$, i.e., of the denotation of $A_i^{\sigma_i}$, it is saved in the designated memory cell, i.e., in the neuron p_i.
3) The rank function, in according to the Acyclicity Constraint (AC), by (9), guarantees that the network (52b) has memorised the corresponding data pieces, after completing the algorithmic computations

$$p_1 : \sigma_1, \ldots, p_n : \sigma_n \tag{52a}$$

$$\{ p_1^{\sigma_1} := A_1^{\sigma_1}, \ldots, p_n^{\sigma_n} := A_n^{\sigma_n} \} \tag{52b}$$

b) Declarative Information: Declarative information is modelled by terms of the form $A : \tau$, where τ is either the type t of truth values, or the type $\widetilde{\mathsf{t}}$ of state dependent truth values (see (11b)).

c) Neural Networks of Declarative Memory: Here, we model declarative memory by the specialised networks, or subnetworks, of systems of assignments of the form (53a):

$$t_1^{\sigma_1} := P_1^{\sigma_1}, \ldots, t_k^{\sigma_n} := P_k^{\sigma_n} \tag{53a}$$

$$\text{for } P_i : \tau_i, \text{ where} \tag{53b}$$

$$\tau_i \text{ is either the type } \mathsf{t} \text{ of truth values, or} \tag{53c}$$

$$\text{the type } \widetilde{\mathsf{t}} \text{ of state dependent truth values} \tag{53d}$$

Declarative memory is innately integrated into networks of procedural memory. That is, neural networks of declarative memory (53a) are typically integrated as recursive subsystems of more general procedural assignments (52b):

$$\{ t_1 : \tau_1, \ldots, t_n : \tau_k \} \subseteq \{ p_1 : \sigma_1, \ldots, p_n : \sigma_n \} \tag{54}$$

B. Algorithmic Binding of Functional Neuro-Receptors

A term T_{m+1} of the form (55a)–(55f) represents a neural network. The head term (55a)–(55e) is a neural sub-network of sequentially bound neural cells (55c), which are sequentially linked by binding functionality. Each subterm $\lambda x_{i_j} Q_{i_{(j-1)}}$ models a neural cell. Its neural body $Q_{i_{j-1}} : ((\sigma_{i_{j-1}} \to \tau_{i_{j-2}}) \to \tau_{i_{j-1}})$ has a receptor represented by its argument slot of the corresponding type $(\sigma_{i_{j-1}} \to \tau_{i_{j-2}})$.

The λ-abstraction λx_{i_j} in $\lambda x_{i_j} Q_{i_{(j-1)}}$ represents the axon of the neural cell $\lambda x_{i_j} Q_{i_{(j-1)}}$. Similarly, the λ-abstraction $\lambda x_{i_{(j-1)}}$, in $\lambda x_{i_{(j-1)}} Q_{i_{(j-2)}}$, represents the axon of $\lambda x_{i_{(j-1)}} Q_{i_{(j-2)}}$. In the subsequently bound (linked) neural cells, represented by the subterm of the form (55c), $Q_{i_{(j-1)}}$ binds the axon $x_{i_{(j-1)}}$ of $\lambda x_{i_{(j-1)}} Q_{i_{(j-2)}}$, for each $j = 3, \ldots, (m-3)$.

$$T_{m+1} \equiv Q_{i_m} \Big[\lambda x_{i_m} Q_{i_{(m-1)}} \big[\lambda x_{i_{(m-1)}} Q_{i_{(m-2)}} [\tag{55a}$$

$$\lambda x_{i_{(m-2)}} Q_{i_{(m-3)}} [\lambda x_{i_{(m-3)}} Q_{i_{(m-4)}} [\tag{55b}$$

$$\ldots$$

$$\lambda x_{i_{(j+1)}} Q_{i_j} [\lambda x_{i_j} Q_{i_{(j-1)}} [\lambda x_{i_{(j-1)}} Q_{i_{(j-2)}} [\tag{55c}$$

$$\ldots$$

$$\lambda x_{i_3} Q_{i_2} [\tag{55d}$$

$$\lambda x_{i_2} Q_{i_1} [\lambda x_{i_1} H(x_1) \ldots (x_n)]]]]]]]]] \Big] \tag{55e}$$

$$\mathsf{where} \ \{ \overrightarrow{p} := \overrightarrow{A} \} \tag{55f}$$

The term T_{m+1} of the form (55a)–(55f) represents a neural network in its 'encapsulated' form, where the algorithmic steps of binding axons by the corresponding receptors are 'hidden' below encapsulating membranes.

In the canonical form $\mathsf{cf}(T_{m+1})$, the head term of T_{m+1} representing the head neural sub-network, i.e., (55a)–(55e), is reduced to a subterm R_{m+1} that have the structural form of R_{m+1} in (34b)–(34i). The term R_{m+1} represents the innate, inner algorithmic structure of the same neural sub-network of T_{m+1}, inside its encapsulating membrane. On the other hand, the neural network R_{m+1} is algorithmically synonymous (equivalent) with the term R'_{m+1} in (36b)–(36i), while they are structurally different.

VII. Forthcoming and Future Work

The recursion assignments in Section IV include λ-terms binding argument slots of the "innermost" subterm, sequentially by recursion within the scope of the recursion operator where. We have started the exposition by reducing the term (32a)-(32e). That resulted the specific variables for the λ-abstracts. However, these terms are congruent to terms by renaming variables bound by the λ-operator. There are more interesting results related to linking of the bindings related to these λ-terms and renaming variables abstracted away with λ-operator. Such properties of the binding operators Q_{i_j} introduced in this paper are in our forthcoming work.

Other forthcoming work is to relate the results in this paper with the reduction calculus in Loukanova [19] and rendering expressions of natural language, e.g., similar to the underspecified quantification presented in Loukanova [5], as well as with other extensions of $\mathrm{L}^{\lambda}_{\mathrm{ar}}$.

Questions whether the approach presented in Ślęzak et al. [22] is comparable with the type theory of Moschovakis algorithms extended in this paper, and if yes, how, remains open work. Studying the shared ideas and differences in these approaches may provide mutual enrichments and developments.

References

[1] Y. N. Moschovakis, "A logical calculus of meaning and synonymy," *Linguistics and Philosophy*, vol. 29, no. 1, pp. 27–89, Feb 2006. [Online]. Available: http://dx.doi.org/10.1007/s10988-005-6920-7

[2] ——, "Sense and denotation as algorithm and value," in *Lecture Notes in Logic*, ser. Lecture Notes in Logic, J. Oikkonen and J. Vaananen, Eds. Springer, 1994, no. 2, pp. 210–249.

[3] H. B. Curry and R. Feys, *Combinatory logic*. Amsterdam: North-Holland Publishing Company, 1958, vol. 1.

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

[5] R. Loukanova, "Relationships between Specified and Underspecified Quantification by the Theory of Acyclic Recursion," *ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal*, vol. 5, no. 4, pp. 19–42, 2016. [Online]. Available: http://dx.doi.org/10.14201/ADCAIJ201654

[6] ——, "Algorithmic Granularity with Constraints," in *Brain and Health Informatics*, ser. Lecture Notes in Computer Science, K. Imamura, S. Usui, T. Shirao, T. Kasamatsu, L. Schwabe, and N. Zhong, Eds. Springer International Publishing, 2013, vol. 8211, pp. 399–408. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-02753-1_40

[7] ——, "Specification of Underspecified Quantifiers via Question-Answering by the Theory of Acyclic Recursion," in *Flexible Query Answering Systems 2015*, ser. Advances in Intelligent Systems and Computing, T. Andreasen, H. Christiansen, J. Kacprzyk, H. Larsen, G. Pasi, O. Pivert, G. D. Tré, M. A. Vila, A. Yazici, and S. Zadrożny, Eds. Springer International Publishing, 2016, vol. 400, pp. 57–69. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-26154-6_5

[8] ——, "Acyclic Recursion with Polymorphic Types and Underspecification," in *Proceedings of the 8th International Conference on Agents and Artificial Intelligence*, J. van den Herik and J. Filipe, Eds., vol. 2. SCITEPRESS — Science and Technology Publications, Lda., 2016, pp. 392–399. [Online]. Available: http://dx.doi.org/10.5220/0005749003920399

[9] ——, "Underspecified Quantification by the Theory of Acyclic Recursion," in *Trends in Practical Applications of Scalable Multi-Agent Systems, the PAAMS Collection*, F. de la Prieta, J. M. Escalona, R. Corchuelo, P. Mathieu, Z. Vale, T. A. Campbell, S. Rossi, E. Adam, D. M. Jiménez-López, M. E. Navarro, and N. M. Moreno, Eds. Cham: Springer International Publishing, 2016, pp. 237–249. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-40159-1_20

[10] J. Barwise and J. Perry, *Situations and Attitudes*. Cambridge, MA:MIT press, 1983, republished as [23].

[11] R. Loukanova and R. Cooper, "Some Situation Theoretical Notions," in *Annuaire de l'Universite de Sofia "St. Kliment Ohridski". Faculte de mathematiques et informatique.* "St. Kliment Ohridski" University Press, 1993, vol. tome 87(1993). livre 1 – mathematiques, pp. 297–306.

[12] R. Loukanova, "Situation semantical analysis of natural language," Ph.D. dissertation, Faculty of Mechanics and Mathematics, Moscow State University (MGU), Moscow, 1991, (in Russian).

[13] ——, "Russellian and Strawsonian Definite Descriptions in Situation Semantics," in *Computational Linguistics and Intelligent Text Processing*, ser. Lecture Notes in Computer Science, A. Gelbukh, Ed. Springer Berlin / Heidelberg, 2001, vol. 2004, pp. 69–79. [Online]. Available: http://dx.doi.org/10.1007/3-540-44686-9_6

[14] ——, "Generalized Quantification in Situation Semantics," in *Computational Linguistics and Intelligent Text Processing*, ser. Lecture Notes in Computer Science, A. Gelbukh, Ed. Springer Berlin / Heidelberg, 2002, vol. 2276, pp. 46–57. [Online]. Available: http://dx.doi.org/10.1007/3-540-45715-1_4

[15] ——, "Quantification and Intensionality in Situation Semantics," in *Computational Linguistics and Intelligent Text Processing*, ser. Lecture Notes in Computer Science, A. Gelbukh, Ed. Springer Berlin / Heidelberg, 2002, vol. 2276, pp. 32–45. [Online]. Available: http://dx.doi.org/10.1007/3-540-45715-1_3

[16] ——, "Situation Theory, Situated Information, and Situated Agents," in *Transactions on Computational Collective Intelligence XVII*, ser. Lecture Notes in Computer Science, N. T. Nguyen, R. Kowalczyk, A. Fred, and F. Joaquim, Eds. Springer Berlin Heidelberg, 2014, vol. 8790, pp. 145–170. [Online]. Available: http://dx.doi.org/10.1007/978-3-662-44994-3_8

[17] ——, "A Formalization of Generalized Parameters in Situated Information," in *Proceedings of the 8th International Conference on Agents and Artificial Intelligence*, J. van den Herik and J. Filipe, Eds., vol. 1. SCITEPRESS — Science and Technology Publications, Lda., 2016, pp. 343–353. [Online]. Available: http://dx.doi.org/10.5220/0005850303430353

[18] ——, "Typed theory of situated information and its application to syntax-semantics of human language," in *Partiality and Underspecification in Information, Languages, and Knowledge*, H. Christiansen, M. D. Jiménez-López, R. Loukanova, and L. S. Moss, Eds. Cambridge Scholars Publishing, 2017, pp. 151–188.

[19] ——, "γ-Reduction in Type Theory of Acyclic Recursion," 2017, (to appear).

[20] E. Kandel, T. Jessell, S. Siegelbaum, J. Schwartz, and A. J. Hudspeth, Eds., *Principles of neural science*. McGraw-Hill, Health Professions Division, 2000. [Online]. Available: http://www.principlesofneuralscience.com

[21] L. Squire and E. Kandel, *Memory: From Mind to Molecules*. Roberts & Co., 2009.

[22] D. Ślęzak, A. Janusz, W. Świeboda, H. S. Nguyen, J. G. Bazan, and A. Skowron, "Semantic analytics of PubMed content," in *Information Quality in e-Health*. Springer, 2011, pp. 63–74.

[23] J. Barwise and J. Perry, *Situations and Attitudes*, ser. The Hume Series. Stanford, California: CSLI Publications, 1999.