Modeling Context Information for Computational Semantics with the Language of Acyclic Recursion

Roussanka Loukanova

Abstract The paper explores the expressiveness of the L^{λ}_{ar} language for richly informative lexical semantics. It introduces an extended version of the language L^{λ}_{ar} to represent semantic information associated with lexemes that have denotations dependent on context information such as space and time locations, "speaker" or "listener" agents, and other potential context components. The technique is demonstrated by rendering referential lexical items such as demonstratives, deictic pronouns, and definite descriptions. It is applicable to any language with richly informative inflection structure.

1 Background and Recent Developments

In a sequence of papers, Moschovakis introduced a class of languages of recursion as a new approach to the mathematical notion of algorithm for applications to computational semantics of languages, in similar ways as to semantics of programing languages, e.g., see [7] for FLR, and [8] for L^{λ}_{ar}. Brief introductions to the syntax and semantics of the higher-order type theory of recursion is also presented in other papers, see [4] and [6]. For the purpose of self-containment, we give a short introduction to the language L^{λ}_{ar}, with details that are relevant to the specifics of this paper.

The technical apparatus of the paper is an inspiration of Situation Semantics utterance types, see [3], into a version of Moschovakis' acyclic recursion language. This consideration does not induce direct relations between versions of Situation Semantics and the language L^{λ}_{ar}, primarily because L^{λ}_{ar} has functional types for total functions, while Situation Semantics uses relational, situated types that are complex information constructs representing partial objects. We extend L^{λ}_{ar} to a version with enriched type system, by adding "uttered" NL expressions inside formally de-

Uppsala, Sweden

fined objects, called utterances, that represent important components of context information. The result is an extended version of L^{λ}_{ar}, with explicit representation of utterance information into formal terms.

2 Short Introduction to the Language of Acyclic Recursion

Types: The types of L^{λ}_{ar} is the smallest set *Types* defined recursively, by using a notational variant of Backus-Naur Form (BNF):

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

Intuitively, the types represent: e the entities of the domain; t is for truth values; s is for states; $(\tau_1 \to \tau_2)$ is for unary functions from objects of type τ_1 to objects of type τ_2. A possibility for more detailed treatment of states is introduced further in this paper, to represent more detailed context information such as time, space location, speaker, etc.

Syntax of L^{λ}_{ar}: The *vocabulary of* L^{λ}_{ar} consists of pairwise disjoint sets, for each type τ: (1) a finite set of constants of type τ, $K_\tau = \{\mathsf{c}_0, \mathsf{c}_1, \ldots, \mathsf{c}_{k_\tau}\}$; (2) a set of pure variables of type τ, $\mathit{PureVars}_\tau = \{\mathsf{v}_0, \mathsf{v}_1, \ldots\}$; and (3) a set of recursion variables, called also locations, of type τ, $\mathit{RecVars}_\tau = \{\mathsf{p}_0, \mathsf{p}_1, \ldots\}$. *Terms of* L^{λ}_{ar}: The recursive rules for the set of L^{λ}_{ar} terms can be expressed by using a notational variant of BNF, which is loaded with additional notations for assigning types. We write $X : \xi$ just in case when X is a term of type ξ (which is not per se part of the term), and X^{ξ} when the type ξ is associated with X (X^{ξ} corresponds to type checking for X). Alternative rules are separated by "|" and are followed by brief explanations:

$$\begin{array}{lll}
A :\equiv & \mathsf{c}^{\tau} : \tau & ;; A \text{ is a constant } c^{\tau} \text{ and of type } \tau; \quad \text{(const-term)} \\
\mid & x^{\tau} : \tau & ;; A \text{ is a variable } x^{\tau} \text{ and of type } \tau; \quad \text{(var-term)} \\
\mid & B^{(\sigma\to\tau)}(C^{\sigma}) : \tau & \text{(appl-term)} \\
\mid & \lambda(v^{\sigma})(B^{\tau}) : (\sigma \to \tau) & (\lambda\text{-term}) \\
\mid & A_0^{\sigma_0} \,\mathsf{where}\, \{p_1^{\sigma_1} := A_1^{\sigma_1}, \ldots, p_n^{\sigma_n} := A_n^{\sigma_n}\} : \sigma_0 & \text{(rec-term})
\end{array}$$

where: in (appl-term), A is a term of type τ, which is application of a term $B^{(\sigma\to\tau)}$ to a term C^{σ}; in (λ-term), A is a term of type $(\sigma \to \tau)$, which is λ-abstraction from a pure variable v^{σ} over a term B^{τ}; in (rec-term), A is a term (recursion term) of type σ_0, in case that, for each $0 \le i \le n$ ($n \ge 0$), $A_i^{\sigma_i}$, is a term of type σ_i, $\{p_i^{\sigma_i} : 1 \le i \le n\}$ are pairwise different locations of the corresponding types, and $\{p_1^{\sigma_1} := A_1^{\sigma_1}, \ldots, p_n^{\sigma_n} := A_n^{\sigma_n}\}$ is an acyclic system. And, where a set $\{p_1^{\sigma_1} := A_1^{\sigma_1}, \ldots, p_n^{\sigma_n} := A_n^{\sigma_n}\}$, for $n \ge 0$[1], is an *acyclic system of assignments* exactly when there is a ranking function $\mathit{rank} : \{p_1, \ldots, p_n\} \longrightarrow \mathbb{N}$ such that, for all $0 \le i, j \le n$, if p_j occurs free in A_i, then $\mathit{rank}(p_j) < \mathit{rank}(p_i)$.

[1] When $n = 0$, the system of assignments is empty, and $A :\equiv A_0^{\sigma_0} \,\mathsf{where}\, \{\} : \sigma_0 :\equiv A_0 : \sigma_0$.

Denotational Semantics of L_{ar}^{λ}: A frame of typed semantic objects $\mathbb{T}$ is a set of typed sets (or classes) $\mathbb{T} = \{\mathbb{T}_{\sigma} \mid \sigma \in Types\}$, where $\mathbb{T}_e \neq \varnothing$ are the entities, $\mathbb{T}_t = \{0, 1, er\}$ are *truth values*, $\mathbb{T}_s \neq \varnothing$ is a set of *states*. In this paper, we define the states as complex objects, with the purpose of modeling utterance information. Given the set G of all variable assignments, the *denotation function* den is:

(D1) $\mathsf{den}(x)(g) = g(x)$; $\mathsf{den}(c)(g) = \mathscr{I}(c)$;
(D2) $\mathsf{den}(A(B))(g) = \mathsf{den}(A)(g)(\mathsf{den}(B)(g))$;
(D3) $\mathsf{den}(\lambda x(B))(g) : \mathbb{T}_{\tau} \longrightarrow \mathbb{T}_{\sigma}$, where $x : \tau$ and $B : \sigma$, is the function defined as follows: for every $t \in \mathbb{T}_{\tau}$, $[\mathsf{den}(\lambda x(B))(g)](t) = \mathsf{den}(B)(g\{x := t\})$;
(D4) $\mathsf{den}(A_0 \text{ where } \{p_1 := A_1, \ldots, p_n := A_n\})(g) = \mathsf{den}(A_0)(g\{p_1 := \overline{p}_1, \ldots, p_n := \overline{p}_n\})$, where $\overline{p_i} = \mathsf{den}(A_i)(g\{p_{k_1} := \overline{p}_{k_1}, \ldots, p_{k_m} := \overline{p}_{k_m}\})$, with $p_{k_1}, \ldots, p_{k_m}$ all the recursion variables with ranks lower than $\mathsf{rank}(p_i)$, for $i \in \{1, \ldots, n\}$.

The Reduction Calculus of L_{ar}^{λ} provides a set of reduction rules (see, [8]) for simplifying the terms A into their canonical forms $\mathsf{cf}(A)$. The canonical form $\mathsf{cf}(A)$ represents the algorithm (when it exists) for calculating the denotation of A, in the simplest way: from the denotations of the basic facts that are computed incrementally and "stored" in locations. Intuitively, an acyclic system $\{p_1 := A_1, \ldots, p_n := A_n\}$ defines recursive "computations" of the values to be assigned to the locations $p_1, \ldots, p_n$, which end after a finite number of steps; $rank(p_j) < rank(p_i)$ means that the value A_i assigned to p_i, may depend on the values of the free p_j in A_i.

3 Richness of L_{ar}^{λ} Type-theoretic Models

Expressiveness of the Language of L_{ar}^{λ}: As proved in [8], L_{ar}^{λ} is a proper extension of Gallin's TY_2, and, thus, of Montague's IL. That L_{ar}^{λ} strictly extends Montague's IL, is demonstrated by terms that are in L_{ar}^{λ}, but not in IL. Typed λ-calculus, with recursion terms, offers its own object language for modeling ambiguity and other semantic underspecification in NL. In addition, there is more subtle representation of different co-reference relations, i.e., co-denotation relations, and antecedent–anaphora relations, see [4] and [6]. In this paper, we explore further the expressiveness of the language L_{ar}^{λ}, with respect to representing semantic information that is associated with language expressions and contributed by their lexical components. In many cases, the denotation $\mathsf{den}(\varphi)$ of an expression strictly depends on the context of use, incl. on utterance components, and the concept of speaker's reference that vary over utterances. For example, among many other utterance dependent language units are the proper names, deictic pronominal expressions, demonstratives, etc. In this section, we initiate exploration of the rich type-theoretic semantic structures of L_{ar}^{λ}, with respect to capturing context dependent semantic information that is presented in NL utterances and. More specifically, we introduce a possibility to use model-theoretic states for modeling utterances of NL expressions and described possible-world indexes.

"Direct" Denotational Semantics of NL Expressions: In this subsection, we introduce the idea of using frames of typed semantic objects $\mathbb{T}$ for "direct" denotational interpretations of NL lexical and phrasal expressions, with the purpose of incorporating the notion of utterance into type-theoretic states. We consider a possibility for the objects in the domain of the states $\mathbb{T}_s$, which is a little bit more elaborated than suggested on p.9-10, of [8], and which borrows some ideas of state informativeness from Situation Semantics (e.g., see [1], [2]). Here, we consider a possibility to define states that represent language utterances (i.e., instances of language use) and speakers' references, as the elements of a set $\mathbb{T}_{refState} \subseteq (\mathbb{T}_u \times \mathbb{T}_{spRef} \times \mathbb{T}_d)$, where the elements of $\mathbb{T}_u$ and $\mathbb{T}_{spRef}$ (and, possibly, $\mathbb{T}_d$) have components that are systematically defined by a formal system of syntax and semantics of some object natural language (for example English, Bulgarian, Russian, etc.). Semantic frames include (as language modeling) representatives of language expressions as components of utterances $\mathbb{T}_u$. For the purposes of this paper, we set: $\mathbb{T}_u \subseteq \mathbb{T}_{speaker} \times \mathbb{T}_{time} \times \mathbb{T}_{space} \times \mathbb{T}_{expr}$ and $\mathbb{T}_d \subseteq \mathbb{T}_{world} \times \mathbb{T}_{time} \times \mathbb{T}_{space}$, where $\mathbb{T}_{time} \subseteq \mathbb{T}_e$ is a set of time moments; $\mathbb{T}_{space} \subseteq \mathbb{T}_e$ is a set of space locations; $\mathbb{T}_{expr}$ is a set of well-formed language expressions, defined by a formal (computational) system of lexicon and grammar; $\mathbb{T}_{speaker} \subseteq \mathbb{T}_{human} \subseteq \mathbb{T}_e$ is a set of people making utterances in the object NL; $\mathbb{T}_{world} \neq \varnothing$ is a nonempty set of possible worlds, taken as primitive objects of the type s; $\mathbb{T}_{spRef}$ is a set of functions, called *speakers' references*: $\delta\colon \mathbb{T}_{speaker} \times \mathbb{T}_{time} \times \mathbb{T}_{space} \times \mathbb{T}_{expr} \longrightarrow \mathbb{T}$.

Reference States: We assume that the formal grammar of NL can be assotiated, in a rigouros way, with a maping Υ, from the syntactic categories of the grammar, into the types of L^{λ}_{ar} (for example, see [9]). The translating procedure Υ should be such that, for every well-formed NL expression φ, of a syntactic category $\mathsf{SynCat}(\varphi)$, the type $\Upsilon(\mathsf{SynCat}(\varphi))$ has to reflect the intuitions about the objects that are (typically) denoted by uttarences of φ. We delimit the set $\mathbb{T}_{spRef}$ as follows: For every $u \in \mathbb{T}_u$, $\delta \in \mathbb{T}_{spRef}$ and $d \in \mathbb{T}_d$, the tuple $\langle u, \delta, d\rangle \in \mathbb{T}_{refState}$ iff for some $a \in \mathbb{T}_{speaker}$, $t_u \in \mathbb{T}_{time}$, $l_u \in \mathbb{T}_{space}$, $\varphi \in \mathbb{T}_{expr}$, $w_d \in \mathbb{T}_{world}$, $t_d \in \mathbb{T}_{time}$, $l_d \in \mathbb{T}_{space}$,

$$u = \langle a, t_u, l_u, \varphi\rangle,\ d = \langle w_d, t_d, l_d\rangle, \text{ and} \tag{2a}$$

$$\text{there is a function } \mathsf{evalindex}\colon \mathbb{T}_u \longrightarrow \mathbb{T}_d \text{ for which} \tag{2b}$$

$$d = \langle w_d, t_d, l_d\rangle = \mathsf{evalindex}(a, t_u, l_u, \varphi)$$

$$\delta(a, t_u, l_u, \varphi) \in \mathbb{T}_{\Upsilon(\mathsf{SynCat}(\varphi))} \tag{2c}$$

$$\text{For every } rs \in \mathbb{T}_{spRef}, \tag{2d}$$

$$[\mathsf{den}(\varphi)(g)](rs) = \begin{cases} \delta(a, t_u, l_u, \varphi), & \text{if } u = \langle a, t_u, l_u, \varphi\rangle \text{ and } rs = \langle u, \delta, d\rangle, \\ er, & \text{otherwise.} \end{cases}$$

Note that the states are not primitive objects: they are complex, structured objects in the semantic frame, represented by tuples of objects including agents (called speakers), time and space locations, and "uttered" language expressions, φ. Such tuples encode essential, context dependent information. While "uttered" expressions φ participate in the semantic frame, and they may have syntactic structure according to

some grammar of the rendered language, the syntactical analyses of φ are not part of the semantic frame. In this paper, we do not make any assumptions for the rendered languages, neither for what kinds of computational syntax may be used. While we do not consider restrictions over what are the "uttered" expressions that may participate in the frame, except that they are potentially randed into L^{λ}_{ar}, primarily, we envisage lexical renderings.

We call the tuples $u = \langle a, t_u, l_u, \varphi \rangle \in \mathbb{T}_u$ *utterances* of φ, iff for some δ and d, $\langle u, \delta, d \rangle \in \mathbb{T}_{refState}$. We call the tuple $\langle u, \delta, d \rangle$ the *current state*, or the *reference state* for the utterance u iff $\langle u, \delta, d \rangle \in \mathbb{T}_{refState}$, i.e., iff (2a)-(2d) hold. In such a case, we also call the component d the *evaluation index* of the (current) state $\langle u, \delta, d \rangle$, which is determined by the following function for every $\langle a, t_u, l_u, \varphi \rangle$, $\mathsf{evalindex}(a, t_u, l_u, \varphi) = \langle w_d, t_d, l_d \rangle = d$. Each tuple $u = \langle a, t_u, l_u, \varphi \rangle$ represents an instance of usage of φ (which is an expression or lexical item, such as an word), by the speaker a, at the moment t_u and the location l_u. The values of the function $\delta \in \mathbb{T}_{spRef}$, $\delta(a, t_u, l_u, \varphi)$, will be called *a's references* in the utterance $u = \langle a, t_u, l_u, \varphi \rangle$. The clause in (2d) includes the possibility for errors, because the denotation of φ can be erroneous for some states *rs*, for example, where the utterance u component of *rs* is not of φ, i.e., $u = \langle a, t_u, l_u, \psi \rangle$ and $\varphi \not\equiv \psi$. The clauses (2a)-(2d) represents utterance dependent denotations, where the speaker refers to a state component d to evaluate the denotation of φ in d. In possible-world semantics, such as Montague's IL, the elements $d \in \mathbb{T}_d$ are called *indexes*. In Situation Semantics, the element $d \in \mathbb{T}_d$ corresponds to a *resource state* (i.e., more correctly, in Situation Semantics, d would be a *resource situation*) for evaluation of the expression φ. When the expression φ is a sentence, the index d, which is provision of the utterance, represents intuitions for a *described state*, because the tuple $\langle w_d, t_d, l_d \rangle$ represents a world w_d, and a space-time point $\langle t_d, l_d \rangle$, referred to by the speaker, as holding (satisfying) the propositional content denoted by φ. In possible-worlds semantics, e.g., as in Montague's IL or its variants, d represents the current, or the "actual" world, for evaluation of the denotation of an expression. Since, in this paper, we use the L^{λ}_{ar} calculus for computational semantics of NL, we use terminology, which (we believe) is in the spirit of L^{λ}_{ar}. The denotation $[\mathsf{den}(\varphi)(g)]$ is the Carnap intension of the proposition made by the speaker. Intuitively, the denotation $[\mathsf{den}(\varphi)(g)](rs) = \delta(a, t_u, l_u, \varphi)$ is the object to which the speaker's refers in the reference state *rs*. When φ is a sentence, this is the truth value, which the speaker associates with the statement made by the utterance u, for the reference index d. This is justified when the reference index d supports the factual content of the utterance of φ.

Deictic Use of Demonstratives in "Direct" Semantics: Demonstrative lexemes, e.g., "here" and "there", are strictly dependent on utterance components and speaker's references, for their strong deictic usage. Let $u \in \mathbb{T}_u$, $\delta \in \mathbb{T}_{spRef}$, and $d \in \mathbb{T}_d$, be such that for arbitrary, but fixed $a \in \mathbb{T}_{speaker}$, $t_u \in \mathbb{T}_{time}$, $l_u \in \mathbb{T}_{space}$, $\varphi \in \{\text{here}, \text{there}\} \subset \mathbb{T}_{expr}$, $w_d \in \mathbb{T}_{world}$, $t_d \in \mathbb{T}_{time}$, $l_d \in \mathbb{T}_{space}$, such that (2a)-(2d) hold, with $\tau \equiv e$, i.e., $\mathsf{den}(\varphi)(g)\colon rs \to e$. In particular, $u = \langle a, t_u, l_u, \varphi \rangle$, are:

$$[\mathsf{den}(\text{here})(g)](u, \delta, d) = \delta(a, t_u, l_u, \text{here}) = l_u = l_d \tag{3a}$$

$$[\mathsf{den}(\text{there})(g)](u, \delta, d) = \delta(a, t_u, l_u, \text{there}) = l_d \in \mathbb{T}_{space}, \text{ and } l_d \neq l_u \tag{3b}$$

Deictic Use of Pronouns in Direct Semantics: Assume that in an utterance u of a sentence like "She smiles", the pronoun "she" is used by the speaker to refer to a particular individual $c \in \mathbb{T}_e$, i.e., $\delta(a, t_u, l_u, \text{she}) = c$. Then

$$[\mathsf{den}(\text{she})(g)] = rs \mapsto \delta(a, t_u, l_u, \text{she}) = rs \mapsto c \tag{4a}$$

$$[\mathsf{den}(\text{she})(g)](u, \delta, d) = \delta(a, t_u, l_u, \text{she}) = c \tag{4b}$$

If $\langle u, \delta, d\rangle \in \mathbb{T}_{refState}$ is the *reference state*, the propositional denotation of the sentence "She smiles" is $smiles(rs \mapsto c) \in \mathbb{T}_t$. The denotation of this sentence in $\langle u, \delta, d\rangle$ is the truth value 1 just in case the individual c smiles in the evaluation index d; i.e., the proposition $smiles(rs \mapsto c) \in \mathbb{T}_t$ is true in d iff $smiles(rs \mapsto c)(u, \delta, d) = 1$. In neither of the above cases, the denotation carries the information that the individual c is female, i.e., the gender information that (depending on the specific language) is part of the semantics of the pronoun "she". The denotational value of the above sentence, with respect to the utterance, by itself, does not specify how it is obtained, i.e., "computed" from the denotations of the lexical components of the sentence. A canonical form, together with formalization of utterances, provides information for that. Pronouns like "she", "he", "it", etc., in their deictic use, resemble demonstratives like "here" and "there", for their strong denotational dependency on the speaker's reference δ and thus on the utterance u. However, the semantics of these pronouns differ from the semantics of the demonstrative "here", in an essential way: these pronouns can participate in anaphoric relations, which is part of their abstract linguistics meaning. This is not necessarily associated with the demonstrative "here", and is not possible for the pronoun "I". The denotation of "I", similarly to that of the demonstratives, is strictly dependent on its utterance, by denoting the speaker, but is independent on the speaker's reference.

4 "Indirect" Lexical Semantics of NL with L^{λ}_{ar}

Why "Indirect" Semantic Representation of NL? It is possible to present "direct" semantics of NL, by formulating direct correspondences, from formal syntax of well-formed NL constructs, to semantic structures, based on typed domain frames. Such attempts for "direct" semantics (exemplified above) would be less perspicacious. Indirect model-theoretic semantics of NL, by rendering it into type-theoretic language, is far more transparent and comes with many other superiorities. Type-theoretical languages are well-studied, have rich theoretical results and model theories. In addition, they offer possibilities for software developments and automatic processing. In this paper, we focus on rendering lexical items into L^{λ}_{ar}. Rendering NL expressions will be discussed only for expository purpose and is topic of other work. More specifically, we will use the language L^{λ}_{ar} to represent semantic information associated with some lexemes, the denotations of which depend on their utterances. Such information is "propagated" recursively into larger NL constructs, by grammar rules.

Rendering Utterance Information into Type-theoretic Terms: The specific denotations of various language units can be dependent on various components of specific utterances: however, the dependency on utterance components can be part of their abstract semantics, in a pattern-like way, as linguistic items out of any specific utterance. This is why, rendering NL items that depend on context of use, can be facilitated by representing utterance components as parts of type-theoretic terms, i.e., in this paper, L^{λ}_{ar}-terms. For this purpose, we take a version of L^{λ}_{ar} language with special constants. For every NL expression φ, let $\tau_{refState}(\varphi)$ be the type of the reference states in the domain $\mathbb{T}_{refState}$, essentially determined by utterances of φ. Note that $\tau_{refState}(\varphi)$ are not ground types. Let $\sigma_u(\varphi)$ be the type of utterances of φ, and let $\mathsf{Type}(\mathsf{render}(\varphi))$ be the type of the rendering of the expression φ. For the current expository purpose, we can take the type of the reference states to be the type of their characteristic function: $\tau_{refState}(\varphi) \equiv (\sigma_u(\varphi) \to ((\sigma_u(\varphi) \to \mathsf{Type}(\mathsf{render}(\varphi))) \to (s \to (\tilde{e} \to (\tilde{e} \to \tilde{t})))))$. We add constants to the language L^{λ}_{ar}, to represent context information: *speaker* for the utterance speaker, *uTime* for the utterance time, *uSpace* for the utterance space, *lWorld* for the local world of the index, *lTime* for the local time, and *lSpace* for the local space of the index, and *spRefs* for the speaker's reference function. The denotations of these constants give the respective components of the reference states, by being defined as follows. For every reference state $r \equiv \langle\langle a, t_u, l_u, \varphi\rangle, \delta, \langle w_d, t_d, l_d\rangle\rangle \in \mathbb{T}_{refState}$,

$$speaker\colon (\tau_{refState}(\varphi) \to e) \text{ and } [\mathsf{den}(speaker)](r) = a \tag{5a}$$
$$uTime\colon (\tau_{refState}(\varphi) \to e) \text{ and } [\mathsf{den}(uTime)](r) = t_u \tag{5b}$$
$$uSpace\colon (\tau_{refState}(\varphi) \to e) \text{ and } [\mathsf{den}(uSpace)](r) = l_u \tag{5c}$$
$$lWorld\colon (\tau_{refState}(\varphi) \to s) \text{ and } [\mathsf{den}(lWorld)](r) = w_d \tag{5d}$$
$$lTime\colon (\tau_{refState}(\varphi) \to e) \text{ and } [\mathsf{den}(lTime)](r) = t_d \tag{5e}$$
$$lSpace\colon (\tau_{refState}(\varphi) \to e) \text{ and } [\mathsf{den}(lSpace)](r) = l_d \tag{5f}$$

and, the constant *spRefs* for speaker's references is such that, for every expression φ, assumed as being a part of utterance types:

$$spRefs\colon (\varphi \to (\tau_{refState}(\varphi) \to \mathsf{Type}(\mathsf{render}(\varphi)))) \tag{6}$$

$$[\mathsf{den}(spRefs(\varphi))](r) = \begin{cases} \delta(a, t_u, l_u, \varphi), & \text{if } \langle a, t_u, l_u, \varphi\rangle \text{ is utterance component in } r \\ er, & \text{otherwise.} \end{cases} \tag{7}$$

5 Utterance Dependency in the Lexicon: Rendering Deictic Lexical Items

There are different kinds of lexical ambiguities, either by association to different part of speech lexical category, or due to "pure" semantic ambiguity of some lex-

emes. An overview of how lexical ambiguity propagates into phrasal ambiguity is presented in [5]. The semantic representation of many lexical items is tightly dependent on prominent context elements such as participating agents, speakers, listeners, space and time locations, some of which we consider in what follows. We demonstrate the technique by considering primarily deictic lexical items.

Deictic Use of Demonstratives in Indirect Semantics Now, demonstrative lexical items, for example such as "here" and "there", can be *rendered* into L^{λ}_{ar} as follows:

$$\begin{aligned} \text{here} \xrightarrow{\mathsf{render}} \mathit{the}(h)\,\mathsf{where}\,\{h &:= \lambda x(r(x) \wedge u(x) \wedge l(x)), \\ r &:= \lambda x(x(\text{here}) = \mathit{spRefs}(\text{here})), \\ u &:= \lambda x(x(\text{here}) = \mathit{uSpace}), \\ l &:= \lambda x(x(\text{here}) = \mathit{lSpace})\}, \end{aligned} \tag{8}$$

where $x\colon \varphi \to (\tau_{\mathit{refState}}(\varphi) \to e)$ is a pure variable.

Deictic Use of Pronouns in Indirect Semantics

$$\begin{aligned} \text{she} \xrightarrow{\mathsf{render}} \mathit{the}(h)\,\mathsf{where}\,\{h &:= \lambda x(r(x) \wedge i(x)), \\ r &:= \lambda x(x = \mathit{spRefs}(\text{she})), \\ i &:= \lambda x\, \mathit{female}(x)\} \end{aligned} \tag{9}$$

Anaphoric usages and underspecified ambiguities between deictic and anaphoric occurrences of pronouns in NL expressions is not a subject of this paper. Here is just a possible rendering:

$$\begin{aligned} \text{she} \xrightarrow{\mathsf{render}} \mathit{the}(h)\,\mathsf{where}\,\{h &:= \lambda x(r(x) \wedge i(x)), \\ i &:= \lambda x\, \mathit{female}(x)\} \end{aligned} \tag{10}$$

Note that, in (10), the location variable r is free, which gives the possibility for associating a specific reading (meaning and denotation) to the pronoun "she" at a later stage, when context prompts for a specific usage. I.e., an underspecified rendering of a pronominal, like in (10), can be bound, either by the speaker's references, or by an antecedent element in a larger expression. The rendering (10) of "she" carries the linguistic information, which is encoded explicitly by the lexical entry, about the gender of the individual denoted by it. Similar gender and number information that is encoded in some lexical items, and is proper semantic information, rather than merely syntactic agreement co-occurrence flag, needs to be included in the rendering type-theoretic terms, as part of their "logic form". Next we show how semantic contributions by inflections can be included in the L^{λ}_{ar} terms that serve as semantic representatives of inflected forms. Exhaustive treatment of inflection rules, such as verbal tenses and aspects, is not a subject of the paper.

Definite Determiner in English Note that the reference states (via utterance expressions φ) provide "resource" states for evaluation of the denotations:

$$\text{the} \xrightarrow{\text{render}} \mathit{the}: \tag{11}$$
$$(\varphi \to (((\tau_{\mathit{refState}}(\varphi) \to e) \to (\tau_{\mathit{refState}}(\varphi) \to t)) \to (\tau_{\mathit{refState}}(\text{the } \varphi) \to e)))$$

Definite NPs in English The speaker's references reflect the referential use of definite descriptions. This is achieved by a constraint over the denotation value via the assignment (12b), which requires the value $\mathit{spRefs}(\text{the homework})$ for the denoted object (via grammar rules for NPs):

$$\text{the homework} \xrightarrow{\text{render}} \mathit{the}(H)\,\text{where}\,\{H := \lambda x(r(x) \wedge h(x)), \tag{12a}$$
$$r := \lambda x(x = \mathit{spRefs}(\text{the homework})), \tag{12b}$$
$$h := \mathit{homework}\} \tag{12c}$$
$$: (\tau_{\mathit{refState}}(\text{the homework}) \to e) \tag{12d}$$

Attributive use of definite NPs is represented by terms like (12a)-(12c), without the assignment (12b).

Definite NPs in Bulgarian

$$\text{домашното} \xrightarrow{\text{render}} \mathit{the}(H)\,\text{where}\,\{H := \lambda x(r(x) \wedge h(x)), \tag{13a}$$
$$r := \lambda x(x = \mathit{spRefs}(\text{домашното})), \tag{13b}$$
$$h := \mathit{homework}\} \tag{13c}$$
$$: (\tau_{\mathit{refState}}(\text{домашното}) \to e) \tag{13d}$$

Indefinite NPs in Bulgarian The semantic distinction between the part of speech of common nouns and indefinite NPs, in Bulgarian, is represented by the types of the rendering terms. For example:

$$\text{домашно} \xrightarrow{\text{render}} \mathit{homework} \qquad \text{common noun} \tag{14a}$$
$$: ((\tau_{\mathit{refState}}(\text{домашно}) \to e) \to (\tau_{\mathit{refState}}(\text{домашно}) \to t))$$
$$\text{домашно} \xrightarrow{\text{render}} \mathit{homework} \qquad \text{existential quantifier} \tag{14b}$$
$$: (\tilde{e} \to \tilde{t}) \to (\tau_{\mathit{refState}}(\text{домашно}) \to t)$$

6 Conclusion

Rendering NL lexicon and grammatical constructs into terms of a type-theoretical language, for the purpose of semantic "representations", has many other positive arguments, as demonstrated firstly by the original works of [9]. The continuing theoretical and computational advances of type theories bring more arguments, especially when targeting computational grammar that covers lexicon with morphological structure that propagates into the phrasal structure of sentences, and complex syntax-semantics inter-relations in languages. The language of acyclic recursion

L^{λ}_{ar} is one of the latest examples for such advances. L^{λ}_{ar} is an elegant mathematical system, which is also richly expressive, not simply by being a proper extension of a higher-order language such as Montague's IL. Language ambiguities and underspecified semantic representation are under major attention in computational semantics and, generally, in NLP, not only because complete resolution of ambiguities present NLP with computational intractability, but is also unnecessary in the absence of relevant context dependant information. The language and theory of L^{λ}_{ar} offers not only a novel algorithmic treatment of Frege's distinction between sense and denotation, it also has its own object language and calculi for representing semantic underspecification of NL expressions. Additionally, our major focus has been the exploration of the expressiveness of L^{λ}_{ar} for more adequate representation of semantic information that is carried by lexical entities, such as deictic language units and lexical inflection, which is typically interpreted depending on information provided by specific context. Such language phenomena, with Slavonic languages just as examples, are in need of computational theory that faithfully captures semantic information and incorporation into computational syntax-semantics interface. Morphology is a rich topic, which requires extensive exploration by development of computational syntax-semantics interface, and the research in this paper is in such directions.

References

1. Jon Barwise and John Perry. *Situations and Attitudes*. Cambridge, MA:MIT press, 1983.
2. Keith Devlin. Situation theory and situation semantics. In Dov Gabbay and John Woods, editors, *Handbook of the History of Logic*, volume Volume 7, pages 601–664. Elsevier, 2008.
3. Roussanka Loukanova. Quantification and intensionality in situation semantics. In Alexander Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing*, volume 2276 of *Lecture Notes in Computer Science*, pages 32–45. Springer Berlin / Heidelberg, 2002.
4. Roussanka Loukanova. Typed lambda language of acyclic recursion and scope underspecification. In Reinhard Muskens, editor, *Workshop on New Directions in Type-theoretic Grammars*, ESSLLI 2007, pages 73–89, Dublin, Ireland, 2007.
5. Roussanka Loukanova. Computational syntax-semantics interface. In Gemma Bel-Enguix and Maria Dolores Jiménez-López, editors, *Language as a Complex System: Interdisciplinary Challenges*, pages 111–150. Cambridge Scholars Publishing, 2010.
6. Roussanka Loukanova. Reference, co-reference and antecedent-anaphora in the type theory of acyclic recursion. In Gemma Bel-Enguix and M. Dolores Jiménez-López, editors, *Bio-Inspired Models for Natural and Formal Languages*, pages 81–102. Cambridge Scholars Publishing, 2011.
7. Yiannis N. Moschovakis. Sense and denotation as algorithm and value. In J. Oikkonen and J. Vaananen, editors, *Lecture Notes in Logic*, number 2 in Lecture Notes in Logic, pages 210–249. Springer, 1994.
8. Yiannis N. Moschovakis. A logical calculus of meaning and synonymy. *Linguistics and Philosophy*, 29:27–89, 2006.
9. Richmond H. Thomason, editor. *Formal Philosophy: Selected Papers of Richard Montague*, ed. Richmond Thomason. Yale University Press, New Haven, Connecticut, 1974. Edited, with an introduction, by Richmond H. Thomason.