Type-Theory of Acyclic Algorithms for Semantics of Natural Languages

Roussanka Loukanova

Department of Mathematics, Stockholm University, Sweden
Institute of Mathematics and Informatics,Bulgarian Academy of Sciences, Bulgaria

56th Linguistics Colloquium
https://sites.google.com/view/lingcoll

Zoom

This presentation is about applications of:

- Type-Theory of Acyclic Algorithms, $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ as a new approach to:
  - the math notion of algorithm
  - Computational Semantics (CompSem)

Placement of $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ in a class of type theories

$$\text{Montague IL} \subsetneq \text{Gallin TY}_2 \subsetneq \text{Moschovakis } \mathrm{L}^{\lambda}_{\mathrm{ar}} \subsetneq \text{Moschovakis } \mathrm{L}^{\lambda}_{\mathrm{r}} \quad (1)$$

$$\equiv \text{Situation Theory} \quad (2)$$

- Montague IL for PTQ (1970-73)
- Gallin TY_2 (1975)
- $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ Moschovakis [3] (2006), Loukanova (since 2007)

  $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ is a higher-order type theory with acyclic recursion terms:
  - the (algorithmic) meanings of the meaningful $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ terms are acyclic algorithms that close-off
- $\mathrm{L}^{\lambda}_{\mathrm{r}}$ is a higher-order type theory with full recursion

- The HL expressions are rendered into some $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ terms representing their semantics
- By reduction calculi of $\mathrm{L}^{\lambda}_{\mathrm{ar}}$, the terms are reduced to canonical terms
- The canonical terms determine algorithms for computing the denotational interpretations
- $\therefore$ it's more efficient to render HL directly into canonical terms

$$\text{John likes Mary's father.} \tag{3a}$$

$$\xrightarrow{\text{render}} \Big[\mathit{like}\big(\mathit{father\_of}\,(\mathit{mary})\big)\Big](\mathit{john}) \;:\; \widetilde{\mathrm{t}} \tag{3b}$$

$$\Rightarrow \;\ldots (\text{by applying rules of the Reduction Calculus of } L^{\lambda}_{ar}) \tag{3c}$$

$$\Rightarrow_{\mathsf{cf}} \mathit{like}(f)(j) \text{ where } \{\, j := \mathit{john},\, m := \mathit{mary},\; f := \mathit{father\_of}\,(m)\} \tag{3d}$$

$$(\textit{a recursion term} \text{ for iterative computations}) \tag{3e}$$

$$\not\approx L(f)(j) \text{ where } \{\, L := \mathit{like},\; j := \mathit{john},\, m := \mathit{mary},\; f := \mathit{father\_of}\,(m)\} \tag{3f}$$

- β-reduction is not valid (in full) for algorithmic equivalence
- β-reduction and the other established results of classic λ-calculi are valid denotationally
    - denotational β-reduction is valid and essential in deciding algorithmic equivalence between terms

$$\text{Mary likes herself.} \tag{4a}$$

$$\xrightarrow{\text{render}} \lambda(x)\big(\mathit{like}(x)(x)\big)(\mathit{mary}) \tag{4b}$$

$$\Rightarrow_{\text{cf}} \lambda(x)\big(\mathit{like}(x)(x)\big)(m) \text{ where } \{m := \mathit{mary}\} \tag{4c}$$

Alternative renderings: (4c) and (5b) are algorithmically equivalent.

$$\text{Mary likes herself.} \tag{5a}$$

$$\xrightarrow{\text{render}}_{\text{cf}} \mathit{like}(m)(m) \text{ where } \{m := \mathit{mary}\} \qquad \text{(canonically)} \tag{5b}$$

$$\approx \lambda(x)\big(\mathit{like}(x)(x)\big)(m) \text{ where } \{m := \mathit{mary}\} \tag{5c}$$

Algorithmically, β reduction is not valid:

$$\lambda(x)\big(\mathit{like}(x)(x)\big)(\mathit{mary}) \tag{6a}$$

$$\xrightarrow[\beta]{\text{assume}} \mathit{like}(\mathit{mary})(\mathit{mary}) \tag{6b}$$

$$\Rightarrow_{\text{cf}} \mathit{like}(m_1)(m_2) \text{ where } \{m_1 := \mathit{mary},\ m_2 := \mathit{mary}\} \tag{6c}$$

Therefore:

$$\lambda(x)\big(\mathit{like}(x)(x)\big)(\mathit{mary}) \not\approx \mathit{like}(\mathit{mary})(\mathit{mary}) \tag{7a}$$

This work is about development and applications of:

- Type-Theory of Acyclic Algorithms, L_{ar}^{λ}
  as a new approach to the math notion of algorithm and
  Computational Semantics (CompSem)
  - a new operator for (acyclic) recursion, Moschovakis [3] (2006)
    - Iterative computations:
    by saving the algorithmic steps and values in memory locations (e.g., for use and reuse)
  - Free memory locations represent semantic underspecification:
    can be specified by instantiation in recursion scope, Loukanova [1]
  - a new operator for restrictions, Loukanova [2], for:
    - restricted computations, and restricted semantic objects
    - restricted memory locations, as generalised, restricted parameters
- Applications
  - Computational Semantics (CompSem)
    of formal (FL) and natural languages (NL)
  - Syntax-Semantics (CompSynSem) of human language (NL / HL)
  - Machine Translation via SynSem Interfaces (prospects)
  - etc.

- Generalised Computational Grammar:
  via faithfull representations of CompSynSem interfaces in HL
  - Hierarchical lexicon with morphological structure and lexical rules
  - Lexicon that propagates into the phrasal structure of sentences
  - Syntax of HL expressions (phrasal and grammatical dependences)
  - Syntax-semantics inter-relations in lexicon and phrases
  - Abstract and Parametric across syn categories and semantic types
    vs.
    Specific Instantiations of lexemes and phrases,
    in and across languages
- A Big Picture — simplified and approximated, but realistic:

Algorithmic CompSynSem of Human Language

$$\underbrace{\text{HL Syn} \iff \mathrm{L}^{\lambda}_{\mathrm{ar}} \,/\, \mathrm{L}^{\lambda}_{\mathrm{r}} \,/ SitT \iff \underbrace{\text{Algorithmic Semantics}}_{\text{Canonical Computations}} \implies \text{Denotations}}_{\text{Algorithmic CompSynSem Interfaces}}$$

(I've done quite a lot of it, but still a lot to do!)

- Algorithmic CompSynSem Syntax-Semantics Interfaces in HL (English, etc.)

Translations via Algorithmic Syntax-Semantics Interfaces
Human Languages and $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ / SithT

$$\text{Lexicon of } \mathsf{NL}_0 \iff \text{Syn of } \mathsf{NL}_0 \xleftrightarrow[\text{render}^{-1}]{\text{render}} \mathrm{L}^{\lambda}_{\mathrm{ar}} \text{ /SitT Canonical Terms}$$

$\uparrow$

possible

modifications

of the terms

$\downarrow$

$$\{\text{Lexicon of } \mathsf{NL}_i \iff \text{Syn of } \mathsf{NL}_i \xleftrightarrow[\text{render}^{-1}]{\text{render}} \mathrm{L}^{\lambda}_{\mathrm{ar}} \text{ /SitT Canonical Terms}$$

$$\mid 1 \leq i \leq n\}$$

- Development of L^{λ}_{ar}, L^{λ}_{r}
- Dependent-Type Theory of Situated Information and Algorithms:
- Extending the coverage of computational semantics

- More to come

- With gratitudes to the team of
  56th LINGUISTICS COLLOQUIUM
- Special thanks to
  Coordinator Reinhard Rapp

THANK YOU!

Some References I

Roussanka Loukanova.
Relationships between Specified and Underspecified Quantification by the Theory of Acyclic Recursion.
ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal, 5(4):19–42, 2016.
URL: https://doi.org/10.14201/ADCAIJ2016541942.

Roussanka Loukanova.
Type-Theory of Parametric Algorithms with Restricted Computations.
In Yucheng et al., editor, *Distributed Computing and Artificial Intelligence, 17th International Conference*, pages 321–331, Cham, 07 August 2020 2021. Springer International Publishing.
URL: https://doi.org/10.1007/978-3-030-53036-5_35.

Yiannis N. Moschovakis.
A Logical Calculus of Meaning and Synonymy.
Linguistics and Philosophy, 29(1):27–89, 2006.
URL: https://doi.org/10.1007/s10988-005-6920-7.