Parametric Information via Type Theory of Acyclic Algorithms

Roussanka Loukanova

Institute of Mathematics and Informatics (IMI)
Bulgarian Academy of Sciences (BAS), Bulgaria

WG3 + MCLP Session
https://europroofnet.github.io/wg3-Sept2025/
Institut Pascal, 17–19 Sep 2025
Part of Final EuroProofNet Symposium

Outline

Type-Theory of Acyclic / Full Algorithms: $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ / L^{λ}_{r}, Moschovakis [10]

Algorithmic CompSynSem of Natural Language (NL) via $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ / L^{λ}_{r}

$$\underbrace{\text{NL Syn} \underbrace{\Longleftrightarrow}_{\text{render}} \mathrm{L}^{\lambda}_{\mathrm{ar}} / \mathrm{L}^{\lambda}_{r} \Longleftrightarrow \underbrace{\text{Algorithms for Comp}}_{\text{Algorithmic Semantics}} \Longrightarrow \underbrace{\text{Denotations}}_{\text{Denotational Semantics}}}_{\text{Algorithmic CompSynSem Interfaces}}$$

- Denotational Semantics of $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ / L^{λ}_{r}: by induction on terms
- Reduction Calculus $A \Rightarrow B$ of $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ / L^{λ}_{r}: by (10+) reduction rules
- The reduction calculus of $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ / L^{λ}_{r} is effective
  Theorem: For every $A \in$ Terms, there is unique, up to congruence, canonical form cf(A), such that:

$$A \Rightarrow_{\mathsf{cf}} \mathsf{cf}(A)$$

- Algorithmic Semantics of $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ / L^{λ}_{r}
  For every algorithmically meaningful $A \in$ Terms:
    - cf(A) determines the algorithm alg(A) for computing den(A)
- I've been extending $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ / L^{λ}_{r} Loukanova [1, 2, 3, 4, 5, 6, 7, 8, 9]

Syntax of Type Theory of Algorithms (TTA): Types, Vocabulary

- Gallin Types (1975)

$$\tau ::= \mathsf{e} \mid \mathsf{t} \mid \mathsf{s} \mid (\tau \to \tau) \qquad \text{(Types)}$$

- Abbreviations

$$\widetilde{\sigma} \equiv (\mathsf{s} \to \sigma), \text{ for state-dependent objects of type } \widetilde{\sigma} \qquad \text{(1a)}$$

$$\widetilde{\mathsf{e}} \equiv (\mathsf{s} \to \mathsf{e}), \text{ for state-dependent entities} \qquad \text{(1b)}$$

$$\widetilde{\mathsf{t}} \equiv (\mathsf{s} \to \mathsf{t}), \text{ for state-dependent truth vals: propositions} \qquad \text{(1c)}$$

- Typed Vocabulary, for all $\sigma \in$ Types

$$\mathsf{Consts}_\sigma = K_\sigma = \{\mathsf{c}^\sigma_0, \mathsf{c}^\sigma_1, \dots\} \qquad \text{(2a)}$$

$$\land, \lor, \to \ \in \mathsf{Consts}_{(\tau \to (\tau \to \tau))}, \ \tau \in \{\mathsf{t}, \widetilde{\mathsf{t}}\} \quad \text{(logical constants)} \qquad \text{(2b)}$$

$$\neg \in \mathsf{Consts}_{(\tau \to \tau)}, \ \tau \in \{\mathsf{t}, \widetilde{\mathsf{t}}\} \quad \text{(logical constant for negation)} \qquad \text{(2c)}$$

$$\mathsf{PureV}_\sigma = \{v^\sigma_0, v^\sigma_1, \dots\} \qquad \text{(2d)}$$

$$\mathsf{RecV}_\sigma = \mathsf{MemoryV}_\sigma = \{p^\sigma_0, p^\sigma_1, \dots\} \qquad \text{(2e)}$$

$$\mathsf{PureV}_\sigma \cap \mathsf{RecV}_\sigma = \varnothing, \qquad \mathsf{Vars}_\sigma = \mathsf{PureV}_\sigma \cup \mathsf{RecV}_\sigma \qquad \text{(2f)}$$

Definition (Terms of TTA: L^{λ}_{ar} acyclic recursion / L^{λ}_{r} full recursion)

$$A :\equiv \mathsf{c}^{\sigma} : \sigma \mid x^{\sigma} : \sigma \mid \mathsf{B}^{(\rho\to\sigma)}(\mathsf{C}^{\rho}) : \sigma \mid \lambda(v^{\rho})\,(\mathsf{B}^{\sigma}) : (\rho \to \sigma) \tag{3a}$$

$$\mid \mathsf{A}_0^{\sigma_0} \text{ where } \{\, p_1^{\sigma_1} := \mathsf{A}_1^{\sigma_1}, \ldots, \ldots, p_n^{\sigma_n} := \mathsf{A}_n^{\sigma_n} \,\} : \sigma_0 \quad \text{(recursion term)} \tag{3b}$$

$$\mid \wedge\,(A_2^{\tau})(A_1^{\tau}) : \tau \mid \vee\,(A_2^{\tau})(A_1^{\tau}) : \tau \mid \to (A_2^{\tau})(A_1^{\tau}) : \tau \tag{3c}$$

$$\mid \neg(B^{\tau}) : \tau \tag{3d}$$

$$\mid \forall(v^{\sigma})(B^{\tau}) : \tau \mid \exists(v^{\sigma})(B^{\tau}) : \tau \quad \text{(pure quantifiers)} \tag{3e}$$

$$\mid \mathsf{A}_0^{\sigma_0} \text{ such that } \{\, \mathsf{C}_1^{\tau_1}, \ldots, \mathsf{C}_m^{\tau_m} \,\} : \sigma_0' \quad \text{(restrictor terms)} \tag{3f}$$

$$\mid \mathsf{ToScope}(B^{\widetilde{\sigma}}) : (\mathsf{s} \to \widetilde{\sigma}) \quad \text{(unspecified scope)} \tag{3g}$$

$$\mid \mathcal{C}(B^{\widetilde{\sigma}}(s)) : \widetilde{\sigma} \quad \text{(closed scope)} \tag{3h}$$

- $\mathsf{c}^{\sigma} \in \mathsf{Consts}_{\sigma}$, $x^{\sigma} \in \mathsf{PureV}_{\sigma} \cup \mathsf{RecV}_{\sigma}$, $v^{\sigma} \in \mathsf{PureV}_{\sigma}$
- $\mathsf{B}, \mathsf{C} \in \mathsf{Terms}$, $p_i^{\sigma_i} \in \mathsf{RecV}_{\sigma_i}$, $A_i^{\sigma_i} \in \mathsf{Terms}_{\sigma_i}$, $\mathsf{C}_j^{\tau_j} \in \mathsf{Terms}_{\tau_j}$
- $\tau, \tau_j \in \{\mathsf{t}, \widetilde{\mathsf{t}}\}$, $\widetilde{\mathsf{t}} \equiv (\mathsf{s} \to \mathsf{t})$ (type of propositions)
  $\mathsf{ToScope} : (\widetilde{\sigma} \to (\mathsf{s} \to \widetilde{\sigma}))$, $\mathcal{C} : (\sigma \to \widetilde{\sigma})$, $s : \mathsf{RecV}_{\mathsf{s}}$ (state), $\sigma \equiv \mathsf{t}$

VP Ellipsis

Example

John loves [his wife]$_{\mathrm{NP}}$, and Peter does too. (4a)

John likes himself, and Peter does too. (4b)

(4a) By limiting (4a) to readings where John loves his own wife

- strict reading of (4a): Peter loves the same person, i.e., [his]$_j$ wife = [John's]$_j$ wife
- sloppy reading of (4a): Peter$_p$ loves [his (own)]$_p$ wife

(4b) reflexive pronouns have restricted denotations

- strict reading of (4b): Peter likes John
- sloppy reading of (4b): Peter likes himself

Rendering Ordinary English: Underspecification of VP Ellipsis in L^{λ}_{ar}

- $h_1, h_2 \in$ RecV, $h_1, h_2 \in$ FreeV(A_0)
  free recursion variables, i.e., memory slots, in (6a)–(6e):

$$\text{John loves his wife, and Peter does too.} \xrightarrow{\text{render}} A_0 \tag{5}$$

$$\begin{aligned}
A_0 \equiv [p_1 \land p_2] \text{ where } \{p_1 &:= L(h_1)(j), && \text{(6a)}\\
L &:= \lambda(x)\lambda(y)\big[love(w(x))(y)\big], && \text{(6b)}\\
p_2 &:= L(h_2)(p), && \text{(6c)}\\
w &:= wife, && \text{(6d)}\\
j &:= john,\ p := peter\} && \text{(6e)}
\end{aligned}$$

- By $h_1, h_2 \in$ FreeV(A), A, (6a)–(6e), represents parametric semantic information corresponding to the underspecified, abstract linguistic meaning of the sentence (5)
- Context may provide specification values of h_1, h_2

1. $h_1 := j,\ h_2 := h_1$
   - A term representing a strict reading:
     - John loves his own wife, and Peter loves the same person
     - $h_2 := h_1$ respects the reference of "his" and the anaphoric "too"
     - "too" has "loves [his]$_j$ wife" as its antecedent, where the name "John" does not occur
2. $h_2 := h_1$, and h_1 is a free recursion variable
   - a term for another strict reading, where
     - John loves the individual that is denoted by the free recursion variable h_1, via the variable valuation of h_1 (the speaker's references)
     - Peter loves the same individual, denoted by h_2, not by direct denotation, but by picking it from the value of h_1.
3. $h_1 := j,\ h_2 := p$
   - a sloppy reading, where each of the men, John and Peter, loves his own wife

Algorithmic Specifications vs Underspecification of VP Ellipsis in L^{λ}_{ar}

- $p_1, p_2, L, r, j, p \in$ RecV, instantiated
- $h_1, h_2 \in$ RecV, $h_1, h_2 \in$ FreeV(A_1)

$$\text{John loves his wife, and Peter does too.} \xrightarrow{\text{render}} A_1 \tag{7}$$

$$A_1 \equiv [p_1 \wedge p_2] \text{ where } \{p_1 := L(h_1)(j), \tag{8a}$$

$$L := \lambda(x)\lambda(y)\big[r(w(x))(y)\big], \tag{8b}$$

$$p_2 := L(h_2)(p), \tag{8c}$$

$$r := love,\ w := wife, \tag{8d}$$

$$j := john,\ p := peter\} \tag{8e}$$

- (8a)–(8c) is a parametric algorithm that can be instantiated by a class of specific properties and objects, of respective types
- By $h_1, h_2 \in$ FreeV(A), A, (8a)–(8e), represents semantically underspecified, parametric information of the sentence (7)
- Context may provide specification values of h_1, h_2

Opt1: Math Text: Algorithmic Specifications vs Underspecification of VP Ellipsis

$$[\ [\text{The number } j]_{\text{NP}}\ [\text{is less than its successor}]_{\text{VP}}\ ]_{\text{S}},\ \text{and } [p \text{ is too}]_{\text{S}} \xrightarrow{\text{render}} A \tag{9}$$

$$A \equiv [p_1 \wedge p_2] \text{ where } \{p_1 := L(h_1)(j), \tag{10a}$$

$$L := \lambda(x)\lambda(y)\big[r(w(x))(y)\big], \tag{10b}$$

$$p_2 := L(h_2)(p), \tag{10c}$$

$$t := the(c),\ c := number, \tag{10d}$$

$$r := LessThan,\ w := successor, \tag{10e}$$

$$j := \lambda(s)t(s_1),\ p := \lambda(s)t(s_2)\} \tag{10f}$$

- (10a)–(10c) is a parametric algorithm that can be instantiated by a class of specific properties and objects, of respective types
- By $h_1, h_2 \in \text{FreeV}(A)$, A, (10a)–(10e), represents semantically underspecified, parametric information of the sentence (9)
- Math discourse may provide specification values of h_1, h_2

Opt2: Math Text: Algorithmic Specifications vs Underspecification of VP Ellipsis

- $p_1, p_2, L, r, j, p \in \mathsf{RecV}$, instantiated
- $\mathit{the} \in \mathsf{Consts}_{((\widetilde{\mathsf{e}} \to \widetilde{\mathsf{t}}) \to \widetilde{\mathsf{e}})}$
- $< \ \in \mathsf{Consts}_{(\widetilde{\mathsf{e}} \to (\widetilde{\mathsf{e}} \to \widetilde{\mathsf{t}}))}$, $S, \mathit{number} \in \mathsf{Consts}_{(\widetilde{\mathsf{e}} \to \widetilde{\mathsf{t}})}$
- $h_1, h_2 \in \mathsf{RecV}$, $h_1, h_2 \in \mathsf{FreeV}(A)$

$$[\ [\text{The number } j]_{\mathrm{NP}}\ [\text{is less than its successor}]_{\mathrm{VP}}\ ]_{\mathrm{S}}, \text{ and } [p \text{ is too}]_{\mathrm{S}} \xrightarrow{\text{render}} A \tag{11}$$

$$A \equiv [p_1 \land p_2] \text{ where } \{p_1 := L(h_1)(j), \tag{12a}$$
$$L := \lambda(x)\lambda(y)\big[r(w(x))(y)\big], \tag{12b}$$
$$p_2 := L(h_2)(p), \tag{12c}$$
$$t := \mathit{the}(c),\ c := \mathit{number}, \tag{12d}$$
$$r := \ <,\ w := S, \tag{12e}$$
$$j := \lambda(s)t(s_1),\ p := \lambda(s)t(s_2)\} \tag{12f}$$

Motivation & Otlook for Type Theory L^{λ}_{ar} / L^{λ}_{r} / DTTSI

- Parametric Algorithmic Patterns for efficient semantic representations, ambiguities, and underspecifications
- Parameters can be instantiated depending on: context, specific areas of applications, etc.
- Translations between:
  - natural language of mathematics and
  - formal languages of proof and verification systems
- L^{λ}_{ar} / L^{λ}_{r} into Dependent-Type Theory of Situated Info (DTTSI)
- L^{λ}_{ar} / L^{λ}_{r}/ DTTSI provide Computational SynSem with:
  - denotations
  - algorithms for computing denotations

Outlook: Computational Theory and Applications to Proof and Verification Systems

- Computational Grammar that faithfully represents syntax-semantics interfaces of and between
  - natural and formal languages, including:
  - programming and specification languages
  - formal languages of proof and verification systems
- The Big Picture: realistic, by having developed significant reduction calculs of L^{λ}_{ar} and L^{λ}_{r}, which can cover major syntactic structures of natural and formal languages

$$\underbrace{\text{NL / Formal Syn} \iff L^{\lambda}_{ar}/L^{\lambda}_{r}/SitT \iff \text{Canonical Forms} \Longrightarrow \text{Denotations}}_{\text{Algorithmic SynSem}}$$

Looking Forward Ahead!

Some References I

Loukanova, R.: Acyclic Recursion with Polymorphic Types and Underspecification.
In: J. van den Herik, J. Filipe (eds.) Proceedings of the 8th International Conference on Agents and Artificial Intelligence, vol. 2, pp. 392–399. SciTePress — Science and Technology Publications, Lda. (2016).
URL https://doi.org/10.5220/0005749003920399

Loukanova, R.: 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

Some References II

- Loukanova, R.: Gamma-Reduction in Type Theory of Acyclic Recursion.
  Fundamenta Informaticae **170**(4), 367–411 (2019).
  URL https://doi.org/10.3233/FI-2019-1867

- Loukanova, R.: Gamma-Star Canonical Forms in the Type-Theory of Acyclic Algorithms.
  In: J. van den Herik, A.P. Rocha (eds.) Agents and Artificial Intelligence. ICAART 2018, *Lecture Notes in Computer Science, book series LNAI*, vol. 11352, pp. 383–407. Springer International Publishing, Cham (2019).
  URL https://doi.org/10.1007/978-3-030-05453-3_18

Some References III

- Loukanova, R.: Type-Theory of Acyclic Algorithms for Models of Consecutive Binding of Functional Neuro-Receptors. In: A. Grabowski, R. Loukanova, C. Schwarzweller (eds.) AI Aspects in Reasoning, Languages, and Computation, vol. 889, pp. 1–48. Springer International Publishing, Cham (2020). URL https://doi.org/10.1007/978-3-030-41425-2_1

- Loukanova, R.: Eta-Reduction in Type-Theory of Acyclic Recursion. ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal **12**(1), 1–22, e29199 (2023). URL https://doi.org/10.14201/adcaij.29199

Some References IV

Loukanova, R.: Logic Operators and Quantifiers in Type-Theory of Algorithms.
In: D. Bekki, K. Mineshima, E. McCready (eds.) Logic and Engineering of Natural Language Semantics. LENLS 2022, *Lecture Notes in Computer Science (LNCS)*, vol. 14213, pp. 173–198. Springer Nature Switzerland, Cham (2023).
URL https://doi.org/10.1007/978-3-031-43977-3_11

Loukanova, R.: Restricted Computations and Parameters in Type-Theory of Acyclic Recursion.
ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal **12**(1), 1–40 (2023).
URL https://doi.org/10.14201/adcaij.29081

Some References V

- Loukanova, R.: Semantics of Propositional Attitudes in Type-Theory of Algorithms.
  In: D. Bekki, K. Mineshima, E. McCready (eds.) Logic and Engineering of Natural Language Semantics. LENLS 2023, *Lecture Notes in Computer Science (LNCS)*, vol. 14569, pp. 260–284. Springer Nature Switzerland AG, Cham (2024).
  URL https://doi.org/10.1007/978-3-031-60878-0_15

- Moschovakis, Y.N.: 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