Type Theory of Acyclic and Cyclic Algorithms without Chain Memory

Roussanka Loukanova

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

Women in EuroProofNet 2025
University of Strathclyde, Glasgow, Scotland
June 10, 2025
https://europroofnet.github.io/women-epn-2025/

Outline

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)
- In a series of papers, I extend $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ / L^{λ}_{r} by new computational facilities, see Loukanova [1, 2, 3, 4, 5, 6, 7, 8, 9]

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

- Gallin Types (1975)

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

- Abbreviations

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

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

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

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

$$\mathsf{Consts}_\sigma = K_\sigma = \{\mathsf{c}_0^\sigma, \mathsf{c}_1^\sigma, \dots\} \tag{2a}$$

$$\wedge, \vee, \to \ \in \mathsf{Consts}_{(\tau \to (\tau \to \tau))}, \ \tau \in \{\mathsf{t}, \widetilde{\mathsf{t}}\} \quad \text{(logical constants)} \tag{2b}$$

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

$$\mathsf{PureV}_\sigma = \{v_0^\sigma, v_1^\sigma, \dots\} \tag{2d}$$

$$\mathsf{RecV}_\sigma = \mathsf{MemoryV}_\sigma = \{p_0^\sigma, p_1^\sigma, \dots\} \tag{2e}$$

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

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

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

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

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

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

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

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

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

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

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

Here, I present a reduction rule that removes "chain" assignments:

- $q := p, \;\; p := A$
- $q := \lambda(\overrightarrow{y})(p(\overrightarrow{y})), \; p := A$ (modulo λ-abstraction)

Chain Rule

For any $A, A_i \in$ Terms, $p, q, p_j \in$ RecVars, $y_k \in$ PureVars, such that $A_i\{\, q :\equiv p \,\}$ is the replacement of all occurrences of q in A_i with p, for $i \in \{\, 0, 1, \dots, n \,\}$, $j \in \{\, 1, \dots, n \,\}$, $k \in \{\, 0, 1, \dots, m \,\}$ $(n, m \geq 0)$,

$$C \equiv_{\mathsf{c}} \big[A_0 \text{ where } \{\, q := \lambda(\overrightarrow{y})(p(\overrightarrow{y})),\; p := A, p_1 := A_1, \dots, p_n := A_n \,\}\big] \tag{4a}$$

(chain)

$$\Rightarrow_{\mathsf{ch}} \big[A_0\{\, q :\equiv p \,\} \text{ where } \{\, p := A, p_1 := A_1\{\, q :\equiv p \,\}, \dots, p_n := A_n\{\, q :\equiv p \,\} \,\}\big] \tag{4b}$$

Compositional SynSem Interface

- The syntactic components are rendered directly into canonical forms:

$$\text{the} \xrightarrow{\text{render}} d \text{ where } \{ d := \mathit{the} \} : ((\widetilde{\mathsf{e}} \to \widetilde{\mathsf{t}}) \to \widetilde{\mathsf{e}}) \tag{5a}$$

$$[\text{the cube}]_{\mathrm{NP}} \xrightarrow{\text{render}} T^0 \equiv i \text{ where } \{ i := d(c),\ d := \mathit{the}, \tag{5b}$$

$$\underbrace{c := \mathit{cube}}_{\text{specification of } c} \ \} \quad : \widetilde{\mathsf{e}} \tag{5c}$$

$$[\text{is large}]_{\mathrm{VP}} \xrightarrow{\text{render}} T_{\mathit{isLarge}} \equiv b \text{ where } \{ b := \mathit{isLarge} \} \ : (\widetilde{\mathsf{e}} \to \widetilde{\mathsf{t}}) \tag{5d}$$

- Composition of the sub-terms directly into canonical forms:

$$\{ [\text{The cube}]_{\mathrm{NP}}, [\text{is large}]_{\mathrm{VP}} \}_{\mathrm{S}} \xrightarrow{\text{render}} T^2 \equiv \mathsf{cf}(T_{\mathit{isLarge}}(T^0)) \tag{6a}$$

$T^1 \equiv T_{\mathit{isLarge}}(T^0) : \widetilde{\mathsf{t}}$ (state-dependent proposition)

$$\Rightarrow b(e) \text{ where } \{ e := i, i := d(c), d := \mathit{the},\ c := \mathit{cube}, \tag{6b}$$

$b := \mathit{isLarge} \} : \widetilde{\mathsf{t}}$ (without (chain) rule)

$$T^1 \Rightarrow_{\mathsf{ch}} b(i) \text{ where } \{ i := d(c), d := \mathit{the},\ c := \mathit{cube}, \quad \text{by (chain)}$$

$$b := \mathit{isLarge} \} \equiv \mathsf{cf}(T_{\mathit{isLarge}}(T^0)) \equiv T^2 : \widetilde{\mathsf{t}} \tag{6c}$$

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 Semantics with:
  - denotations
  - algorithms for computing denotations

Conclusion

- The algorithmic semantics of $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ L^{λ}_{r} is determined by the canonical forms $\mathsf{cf}(A)$:

$$\underbrace{\text{Syntax of } \mathrm{L}^{\lambda}_{\mathrm{ar}}\ (\mathrm{L}^{\lambda}_{r}) \Longrightarrow \text{Algorithms: } \mathsf{alg}(A) = \mathsf{alg}(\mathsf{cf}(A)) \Longrightarrow \text{Denotations } \mathsf{den}(A)}_{\text{Algorithmic Semantics of } \mathrm{L}^{\lambda}_{\mathrm{ar}}(\mathrm{L}^{\lambda}_{r})}$$

Looking Forward!
Thanks!

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