Pure versus Generalised Quantifiers

Roussanka Loukanova

Institute of Mathematics and Informatics (IMI)
Bulgarian Academy of Sciences (BAS), Sofia, Bulgaria
https://www.math.bas.bg/logic/loukanovarp/

International Conference on
Mathematical and Computational Linguistics for Proofs, MCLP
https://europroofnet.github.io/MCLP/

Institut Pascal, Orsay, France, 15–18 Sep 2025

Final EuroProofNet Symposium

Outline

1. Moschovakis (1989) [16] Formal Language of full recursion, untyped
2. Moschovakis (2006) [17], via examples of Natural Language (NL):
   Type-Theory of Acyclic / Full Recursion $\mathrm{L}_{\mathrm{ar}}^{\lambda}$ / L_{r}^{λ}
   Formal Syntax of $\mathrm{L}_{\mathrm{ar}}^{\lambda}$ + Reduction Calculus of $\mathrm{L}_{\mathrm{ar}}^{\lambda}$
3. Open: Algorithmic Dependent-Type Theory of Situated Information (DTTSitInfo): situated data including context assessments:

   - Loukanova (1989–1991) introduced
     math of recursively defined
     Type Theory of Situated Info: applied to Natural Language
     Sitation Semantics

Algorithmic CompSynSem of $\mathrm{L}_{\mathrm{ar}}^{\lambda}$ / L_{r}^{λ}

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

Development of Type-Theory of (Acyclic) Algorithms, L_r^{λ} / L_{ar}^{λ} / DTTSitInfo

Placement of L_{ar}^{λ} in a class of type theories

$$\text{Montague IL} \subsetneq \text{Gallin TY}_2 \subsetneq \text{Moschovakis } L_{ar}^{\lambda} \subsetneq \text{Moschovakis } L_r^{\lambda} \tag{1}$$

$$\stackrel{?}{\equiv} \text{DTTSitInfo} \tag{2}$$

- Type-Theory of (Acyclic) Algorithms, L_r^{λ} (L_{ar}^{λ}), provides:
  - a math notion of algorithm
  - Algorithmic Semantics of formal and natural languages
- L_{ar}^{λ} / L_r^{λ} is type theory of algorithms with acyclic / full recursion:
  - Introduced by Moschovakis [17] (2006) by examples from NL
  - Math development by motivations from NL, Loukanova [8, 9] (2019)
- In the works presented here, I extend L_{ar}^{λ} / L_r^{λ} by incorporating:
  - operators and corresponding reduction rules, by:
  - Preserving acyclicity of L_{ar}^{λ}

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

$$\widetilde{\sigma} \equiv (\mathsf{s} \to \sigma), \text{ for state-dependent objects of type } \widetilde{\sigma} \qquad (3a)$$

$$\widetilde{\mathsf{e}} \equiv (\mathsf{s} \to \mathsf{e}), \text{ for state-dependent entities} \qquad (3b)$$

$$\widetilde{\mathsf{t}} \equiv (\mathsf{s} \to \mathsf{t}), \text{ for state-dependent truth vals: propositions} \qquad (3c)$$

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

$$\mathsf{Consts}_{\sigma} = K_{\sigma} = \{\mathsf{c}_0^{\sigma}, \mathsf{c}_1^{\sigma}, \dots\} \qquad (4a)$$

$$\wedge, \vee, \to \ \in \mathsf{Consts}_{(\tau\to(\tau\to\tau))}, \ \tau \in \{\mathsf{t}, \widetilde{\mathsf{t}}\} \quad \text{(logical constants)} \qquad (4b)$$

$$\neg \in \mathsf{Consts}_{(\tau\to\tau)}, \ \tau \in \{\mathsf{t}, \widetilde{\mathsf{t}}\} \quad \text{(logical constant for negation)} \qquad (4c)$$

$$\mathsf{PureV}_{\sigma} = \{v_0^{\sigma}, v_1^{\sigma}, \dots\} \qquad (4d)$$

$$\mathsf{RecV}_{\sigma} = \mathsf{MemoryV}_{\sigma} = \{p_0^{\sigma}, p_1^{\sigma}, \dots\} \qquad (4e)$$

$$\mathsf{PureV}_{\sigma} \cap \mathsf{RecV}_{\sigma} = \varnothing, \qquad \mathsf{Vars}_{\sigma} = \mathsf{PureV}_{\sigma} \cup \mathsf{RecV}_{\sigma} \qquad (4f)$$

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

$$\mathsf{A} :\equiv \mathsf{c}^{\sigma} : \sigma \mid x^{\sigma} : \sigma \mid \mathsf{B}^{(\rho\to\sigma)}(\mathsf{C}^{\rho}) : \sigma \mid \lambda(v^{\rho})\,(\mathsf{B}^{\sigma}) : (\rho \to \sigma) \tag{5a}$$

$$\mid \mathsf{A}_0^{\sigma_0} \text{ where } \{\, p_1^{\sigma_1} := \mathsf{A}_1^{\sigma_1}, \ldots, \ldots, p_n^{\sigma_n} := \mathsf{A}_n^{\sigma_n} \,\} : \sigma_0 \quad \text{(recursion term)} \tag{5b}$$

$$\mid \wedge\,(A_2^{\tau})(A_1^{\tau}) : \tau \mid \vee\,(A_2^{\tau})(A_1^{\tau}) : \tau \mid \to (A_2^{\tau})(A_1^{\tau}) : \tau \tag{5c}$$

$$\mid \neg(B^{\tau}) : \tau \tag{5d}$$

$$\mid \forall(v^{\sigma})(B^{\tau}) : \tau \mid \exists(v^{\sigma})(B^{\tau}) : \tau \quad \text{(pure quantifiers)} \tag{5e}$$

$$\mid \mathsf{A}_0^{\sigma_0} \text{ such that } \{\, \mathsf{C}_1^{\tau_1}, \ldots, \mathsf{C}_m^{\tau_m} \,\} : \sigma_0' \quad \text{(restrictor terms)} \tag{5f}$$

$$\mid \mathsf{ToScope}(B^{\widetilde{\sigma}}) : (\mathsf{s} \to \widetilde{\sigma}) \quad \text{(unspecified scope)} \tag{5g}$$

$$\mid \mathcal{C}(B^{\widetilde{\sigma}}(s)) : \widetilde{\sigma} \quad \text{(closed scope)} \tag{5h}$$

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

Type Theory of Algorithms (TTA): L^{λ}_{ar} acyclic recursion: Terms + AC

$$A \equiv \mathsf{A}_0^{\sigma_0} \text{ where } \underbrace{\{ p_1^{\sigma_1} := \mathsf{A}_1^{\sigma_1}, \ldots, \ldots, p_n^{\sigma_n} := \mathsf{A}_n^{\sigma_n} \}}_{\text{acyclic system of assignments}} : \sigma_0 \tag{6}$$

- Acyclicity Constraint (AC), for L^{λ}_{ar}:

$$\{ p_1^{\sigma_1} := A_1^{\sigma_1}, \ldots, p_n^{\sigma_n} := A_n^{\sigma_n} \} \quad (n \geq 0) \tag{7a}$$

is acyclic system of assignments iff there exists a function

$$\mathsf{rank}\colon \{p_1, \ldots, p_n\} \to \mathbb{N} \tag{7b}$$

such that:

$$\text{if } p_j \text{ occurs freely in } A_i, \text{ then } \mathsf{rank}(p_i) > \mathsf{rank}(p_j) \tag{7c}$$

1. TTA L^{λ}_{ar} of acyclic recursion:
   Def. 1, (5b) + AC
2. TTA L^{λ}_{r} of full recursion: without acyclicity AC
   Def. 1, (5b) - AC

Algorithmic CompSynSem of $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ / 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
- 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)
- 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:

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

- In a series of papers, I extend $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ / L^{λ}_{r} by algorithmically computational facilities, see Loukanova [5, 6, 8, 9, 10, 11, 12, 13, 14]

Types of Restrictor Terms

In the restrictor term (5f) / (8),

$$A_0^{\sigma_0} \text{ such that } \{ C_1^{\tau_1}, \ldots, C_n^{\tau_n} \} : \sigma_0' \tag{8}$$

for each $i = 1, \ldots, n$:

- $\tau_i \equiv \mathsf{t}$ (state independent truth values), or
- $\tau_i \equiv \widetilde{\mathsf{t}} \equiv (\mathsf{s} \to \mathsf{t})$ (state dependent truth values)

$$\sigma_0' \equiv \begin{cases} \sigma_0, & \text{if } \tau_i \equiv \mathsf{t}, \text{ for all } i \in \{1, \ldots, n\} \quad \text{(9a)} \\ \sigma_0 \equiv (\mathsf{s} \to \sigma), & \text{if } \tau_i \equiv \widetilde{\mathsf{t}}, \text{ for some } i \in \{1, \ldots, n\}, \text{ and} \quad \text{(9b)} \\ & \quad \text{for some } \sigma \in \text{Types}, \ \sigma_0 \equiv (\mathsf{s} \to \sigma) \\ \widetilde{\sigma_0} \equiv (\mathsf{s} \to \sigma_0), & \text{if } \tau_i \equiv \widetilde{\mathsf{t}}, \text{ for some } i \in \{1, \ldots, n\}, \text{ and} \quad \text{(9c)} \\ & \quad \text{there is no } \sigma, \text{ s.th. } \sigma_0 \equiv (\mathsf{s} \to \sigma) \end{cases}$$

Denotational Semantics of $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ / $\mathrm{L}^{\lambda}_{rar}$

A standard semantic structure is a tuple $\mathfrak{A}(\mathsf{Consts}) = \langle\, \mathbb{T}, \mathcal{I}\,\rangle$ that satisfies the following conditions:

- $\mathbb{T} = \{\mathbb{T}_\sigma \mid \sigma \in \mathsf{Types}\}$ is a frame of typed objects
  $\{\,0, 1, er\,\} \subseteq \mathbb{T}_{\mathsf{t}} \subseteq \mathbb{T}_{\mathsf{e}}$ $\;(er_{\mathsf{t}} \equiv er_{\mathsf{e}} \equiv er \equiv error)$
  $\mathbb{T}_{\mathsf{s}} \neq \varnothing$ (the domain of *states*)
  $\mathbb{T}_{(\tau_1 \to \tau_2)} = (\mathbb{T}_{\tau_1} \to \mathbb{T}_{\tau_2}) = \{\, f \mid f\colon \mathbb{T}_{\tau_1} \to \mathbb{T}_{\tau_2}\,\}$ (standard str.)
  $er_\sigma \in \mathbb{T}_\sigma$, for every $\sigma \in \mathsf{Types}$ (designated typed errors)
- $\mathcal{I}\colon$ Consts $\longrightarrow \cup\,\mathbb{T}$ is a typed *interpretation function*:
  $\mathcal{I}(\mathsf{c}) \in \mathbb{T}_\sigma$, for every $\mathsf{c} \in \mathsf{Consts}_\sigma$
- $\mathfrak{A}$ is associated with the set of the typed variable valuations G:

$$G = \{g \mid g\colon\ \mathsf{PureV} \cup \mathsf{RecV} \longrightarrow \bigcup \mathbb{T} \quad \text{and, for every } X \in \mathsf{Vars}_\sigma,\ \ g(X) \in \mathbb{T}_\sigma\} \tag{10}$$

The Denotation Function of $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ / $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ (to be continued)

- Let's assume a given semantic structure $\mathfrak{A}$, and write den $\equiv$ den$^{\mathfrak{A}}$
- There is a unique function, called the *denotation function*:
  $\mathsf{den}^{\mathfrak{A}} \colon \mathsf{Terms} \longrightarrow \{\, f \mid f \colon G \longrightarrow \cup\, \mathbb{T}\,\}$
  defined by recursion on the structure of the terms

(D1)
1. $\mathsf{den}(X)(g) = g(x)$, for every $X \in \mathsf{Vars}$
2. $\mathsf{den}(\mathsf{c})(g) = \mathcal{I}(\mathsf{c})$, for every $\mathsf{c} \in \mathsf{Consts}$

(D2) $\mathsf{den}(A(B))(g) = \mathsf{den}(A)(g)(\mathsf{den}(B)(g))$

(D3) $\mathsf{den}(\lambda x(B))(g)\big(a\big) = \mathsf{den}(B)(g\{x := a\})$, for every $a \in \mathbb{T}_{\tau}$

The Denotation of the Recursion Terms (continuation) (to be continued)

(D4) $\text{den}(A_0 \text{ where } \{p_1 := A_1, \dots, p_n := A_n\})(g) =$
$\text{den}(A_0)(g\{p_1 := \overline{p}_1, \dots, p_n := \overline{p}_n\})$

where $\overline{p}_i \in \mathbb{T}_{\tau_i}$ are defined by recursion on $\text{rank}(p_i)$:

$$\overline{p_i} = \text{den}(A_i)(g\{p_{k_1} := \overline{p}_{k_1}, \dots, p_{k_m} := \overline{p}_{k_m}\})$$

given that $p_{k_1}, \dots, p_{k_m}$ are all of the recursion variables $p_j \in \{p_1, \dots, p_n\}$, s.t. $\text{rank}(p_j) < \text{rank}(p_i)$.

Intuitively:

- $\text{den}(A_1)(g), \dots, \text{den}(A_n)(g)$ are computed recursively, by $\text{rank}(p_i)$, and stored in p_i, $1 \leq i \leq n$
- the denotation $\text{den}(A_0)(g)$ may depend on the values stored in $p_1, \dots, p_n$

(D5) (for the constants of the logic operators) ...

The Denotation of the Logic-Quantifiers Terms (continuation) (to be continued)

(D6b) Simplified version, without considering the erroneous cases of er

The denotation of the state-dependent, pure existential quantifier, for $\tau = \widetilde{\mathsf{t}}$, $\mathsf{den}^{\mathfrak{A}}\big(\exists(v^{\sigma})(B^{\tau})\big)(g) \colon \mathbb{T}_{\mathsf{s}} \to \mathbb{T}_{\mathsf{t}}$ is such that:

$$\text{for every state } s \in \mathbb{T}_{\mathsf{s}}: \tag{11a}$$

$$\Big[\mathsf{den}^{\mathfrak{A}}\big(\exists(v^{\sigma})(B^{\tau})\big)(g)\Big](s) = 1 \ \text{(true in } s\text{)} \tag{11b}$$

$$\begin{aligned}&\text{iff there is } a \in \mathbb{T}_{\sigma}, \text{ in the semantic domain } \mathbb{T}_{\sigma}, \text{ such that:}\\ &\qquad \Big[\mathsf{den}^{\mathfrak{A}}\big(B^{\tau}\big)(g\{\, v := a \,\})\Big](s) = 1\end{aligned} \tag{11c}$$

The Denotation Function for the Restrictor Terms (continuation) (to be continued)

(D7) For every $g \in G$, and every state $s \in \mathbb{T}_{\mathsf{s}}$:

Case 1: for all $i \in \{1, \dots, n\}$, $C_i \in \mathsf{Terms}_{\mathsf{t}}$ (independent on states)

For every $g \in G$:

$$\mathsf{den}\left(A_0^{\sigma_0} \text{ s.t. } \{\, \overrightarrow{C} \,\}\right)(g) = \begin{cases} \mathsf{den}(A_0)(g), & \text{if, for all } i \in \{1, \dots, n\}, \\ & \quad \mathsf{den}(C_i)(g) = 1 \\ er_{\sigma_0} & \text{if, for some } i \in \{1, \dots, n\}, \\ & \quad \mathsf{den}(C_i)(g) = 0 \text{ or} \\ & \quad \mathsf{den}(C_i)(g) = er \end{cases} \tag{12}$$

Case 2: for some $i \in \{1, \ldots, n\}$, $C_i : \tilde{\mathsf{t}}$

$$\text{den}\left(A_0^{\sigma_0} \text{ s.t. } \{\overrightarrow{C}\}\right)(g)(s) \tag{13}$$

$$= \begin{cases} \text{den}(A_0)(g)(s), & \text{if } \text{den}(C_i)(g) = 1, \text{ for all } i \text{ s.th. } C_i : \mathsf{t}, \text{ and} \\ & \quad \text{den}(C_i)(g)(s) = 1, \text{ for all } i \text{ s.th. } C_i : \tilde{\mathsf{t}}, \text{ and} \\ & \quad \sigma_0 \equiv (\mathsf{s} \to \sigma) \\[2ex] \text{den}(A_0)(g), & \text{if } \text{den}(C_i)(g) = 1, \text{ for all } i \text{ s.th. } C_i : \mathsf{t}, \text{ and} \\ & \quad \text{den}(C_i)(g)(s) = 1, \text{ for all } i \text{ s.th. } C_i : \tilde{\mathsf{t}}, \text{ and} \\ & \quad \sigma_0 \not\equiv (\mathsf{s} \to \sigma), \quad \text{for all } \sigma \in \text{Types} \\[2ex] er_{\sigma_0'}, & \text{otherwise} \end{cases}$$

- $A \in$ Terms is explicit iff the operator where does not occur in A
- $A \in$ Terms is a *λ-calculus term* iff it is explicit and no recursion variable occurs in it

Definition (Immediate and Proper Terms)

- The set ImT of immediate terms is defined by recursion (14)

$$T :\equiv V \mid p(v_1)\dots(v_m) \mid \lambda(u_1)\dots\lambda(u_n)p(v_1)\dots(v_m) \tag{14}$$

for $V \in$ Vars, $p \in$ RecV, $u_i, v_j, \in$ PureV,
$i = 1, \dots, n,\ \ j = 1, \dots, m\ (m, n \geq 0)$

- Every $A \in$ Terms that is not immediate is proper

$$\mathsf{PrT} = (\mathsf{Terms} - \mathsf{ImT}) \tag{15}$$

Immediate terms do not carry algorithmic sense:
den$(p(v_1)\dots(v_m))$ is by variable valuation, in memory $p \in$ RecV.

Definition (Congruence Relation, informally)

The congruence relation is the smallest equivalence relation (i.e., reflexive, symmetric, transitive) between L^{λ}_{ar}-terms, $A \equiv_{c} B$, that is closed under:

1. operators of term-formation:
   - application
   - λ-abstraction
   - logic operators
   - pure, logic quantifiers
   - (acyclic) recursor
   - restrictor
2. renaming bound variables (pure and recursion), without causing variable collisions
3. re-ordering of the assignments within the acyclic sequences of assignments in the recursion terms
4. re-ordering of the restrictions in the restrictor terms

[Congruence] If $A \equiv_c B$, then $A \Rightarrow B$ (cong)

[Transitivity] If $A \Rightarrow B$ and $B \Rightarrow C$, then $A \Rightarrow C$ (trans)

[Compositionality]

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

- If $A \Rightarrow B$, and $\xi \in \{ \lambda, \exists, \forall \}$, then $\xi(u)(A) \Rightarrow \xi(u)(B)$ (lq-comp)

- If $A_i \Rightarrow B_i$ $(i = 0, \ldots, n)$, then

$$\begin{aligned} & A_0 \text{ where } \{ p_1 := A_1, \ldots, p_n := A_n \} \\ \Rightarrow\ & B_0 \text{ where } \{ p_1 := B_1, \ldots, p_n := B_n \} \end{aligned} \quad \text{(wh-comp)}$$

- If $A_0 \Rightarrow B_0$ and $C_i \Rightarrow R_i$ $(i = 0, \ldots, n)$, then

$$\begin{aligned} & A_0 \text{ such that } \{ C_1, \ldots, C_n \} \\ \Rightarrow\ & B_0 \text{ such that } \{ R_1, \ldots, R_n \} \end{aligned} \quad \text{(st-comp)}$$

Reduction Rules (to be continued)

[Head Rule] Given that $p_i \neq q_j$ and no p_i occurs freely in any B_j,

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

[Bekič-Scott Rule] Given that $p_i \neq q_j$ and no q_i occurs freely in any A_j

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

[Recursion-Application Rule] Given that no p_i occurs freely in B,

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

Reduction Rules (to be continued)

[Application Rule] Given that $B \in$ PrT is a proper term, and p is fresh, $p \in \big[\,\text{RecV} - \big(\,\text{FV}\,\big(A(B)\big) \cup \text{BV}\,\big(A(B)\big)\big)\big]$,

$$A(B) \;\Rightarrow\; \big[A(p) \text{ where } \{\, p := B \,\}\big] \tag{ap}$$

[λ and Quantifiers rules] Let $\xi \in \{\,\lambda, \exists, \forall\,\}$.
Given fresh $p_i' \in \big[\,\text{RecV} - \big(\,\text{FV}(A) \cup \text{BV}(A)\big)\big]$, $i = 1, \dots, n$, for $A \equiv A_0$ where $\{\, p_1 := A_1, \dots, p_n := A_n \,\}$ and replacements A_i' in (19):

$$A_i' \equiv \Big[A_i\{\, p_1 :\equiv p_1'(u), \dots, p_n :\equiv p_n'(u) \,\}\Big] \tag{19}$$

$$\begin{gathered} \xi(u)\,\Big(A_0 \text{ where } \{\, p_1 := A_1, \dots, p_n := A_n \,\}\Big) \\ \Rightarrow \xi(u)\, A_0' \text{ where } \{\, p_1' := \lambda(u)\, A_1', \dots, p_n' := \lambda(u)\, A_n' \,\} \end{gathered} \tag{ξ}$$

- each $R_i^{\tau_i} \in$ Terms in $\overrightarrow{R}$ is immediate and (has a type of truth value) $\tau_i \in \{ \mathsf{t}, \widetilde{\mathsf{t}} \}$
- each $C_j^{\tau_j} \in$ Terms is proper and has a type τ_j of truth value $\tau_j \in \{ \mathsf{t}, \widetilde{\mathsf{t}} \}$ $(j = 1, \ldots, m,\ m \geq 0)$
- $a_0, c_j \in$ RecV $(j = 1, \ldots, m)$ fresh

(st1) Rule A_0 is an immediate term, $m \geq 1$

$$
\begin{aligned}
&(A_0 \text{ such that } \{ C_1, \ldots, C_m, \overrightarrow{R} \}) \\
\Rightarrow &(A_0 \text{ such that } \{ c_1, \ldots, c_m, \overrightarrow{R} \}) \\
&\qquad \text{where } \{ c_1 := C_1,\ \ldots, c_m := C_m \}
\end{aligned}
\tag{st1}
$$

(st2) Rule A_0 is a proper term

$$
\begin{aligned}
&(A_0 \text{ such that } \{ C_1, \ldots, C_m, \overrightarrow{R} \}) \\
\Rightarrow &(a_0 \text{ such that } \{ c_1, \ldots, c_m, \overrightarrow{R} \}) \\
&\qquad \text{where } \{ a_0 := A_0, \\
&\qquad\qquad c_1 := C_1,\ \ldots, c_m := C_m \}
\end{aligned}
\tag{st2}
$$

Definition ($\gamma*$-condition)

A term $A \in$ Terms satisfies the γ^*-condition for an assignment $p := \lambda(\overrightarrow{u}^{\overrightarrow{\sigma}})\lambda(v^{\sigma})P^{\tau} : (\overrightarrow{\sigma} \to (\sigma \to \tau))$, with respect to $\lambda(v^{\sigma})$, iff A is of the form: (22a)–(22c):

$$A \equiv A_0 \text{ where } \{\, \overrightarrow{a} := \overrightarrow{A}, \tag{22a}$$

$$p := \lambda(\overrightarrow{u})\lambda(v)P, \tag{22b}$$

$$\overrightarrow{b} := \overrightarrow{B} \,\} \tag{22c}$$

such that the following holds:

1. $v \notin \text{FreeVars}(P)$
2. All occurrences of p in A_0, $\overrightarrow{A}$, and $\overrightarrow{B}$ are occurrences:
   - in $p(\overrightarrow{u})(v)$,
   - which are in the scope of $\lambda(v)$
     modulo renaming the variables $\overrightarrow{u}, v$

(γ^*)-rule

$$A \equiv A_0 \text{ where } \{ \overrightarrow{a} := \overrightarrow{A}, \tag{23a}$$

$$p := \lambda(\overrightarrow{u})\lambda(v)P, \tag{23b}$$

$$\overrightarrow{b} := \overrightarrow{B} \} \tag{23c}$$

$$\Rightarrow_{(\gamma^*)} A'_0 \text{ where } \{ \overrightarrow{a} := \overrightarrow{A}', \tag{23d}$$

$$p' := \lambda(\overrightarrow{u})P, \tag{23e}$$

$$\overrightarrow{b} := \overrightarrow{B'} \} \tag{23f}$$

given that:

- $A \in$ Terms satisfies the γ^*-condition (in Definition 4) for $p := \lambda(\overrightarrow{u})\lambda(v)P : (\overrightarrow{\sigma} \to (\sigma \to \tau))$, with respect to $\lambda(v)$
- $p' \in \mathsf{RecV}_{(\overrightarrow{\sigma} \to \tau)}$ is a fresh recursion variable
- $\overrightarrow{X'} \equiv \overrightarrow{X}\{p(\overrightarrow{u})(v) :\equiv p'(\overrightarrow{u})\}$ is the result of the replacements

  $X_i\{p(\overrightarrow{u})(v) :\equiv p'(\overrightarrow{u})\}$,
  i.e., replacing all occurrences of $p(\overrightarrow{u})(v)$ by $p'(\overrightarrow{u})$, in all corresponding parts $X_i \equiv A_i$, $X_i \equiv B_i$, in (23a)–(23f), modulo renaming the variables $\overrightarrow{u}, v$

The optional (chain) rule removes steps of repeated savings via chain-like term assignments. No need of logging the info in q:

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

Chain Rule

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

$$C \equiv_{\mathsf{c}} \big[A_0 \text{ where } \{\, q := \lambda(\overrightarrow{y})(p(\overrightarrow{y})),\ p := A, p_1 := A_1, \ldots, p_n := A_n \,\}\big] \tag{24a}$$

(chain)

$$\Rightarrow_{\mathsf{ch}} D \equiv_{\mathsf{c}} \big[A_0\{\, q :\equiv p \,\} \text{ where } \{\, p := A, p_1 := A_1\{\, q :\equiv p \,\}, \ldots, p_n := A_n\{\, q :\equiv p \,\} \,\}\big] \tag{24b}$$

Theorem (γ^*-Canonical Form Theorem)

For each $A \in \text{Terms}$, *there is a unique up to congruence,* γ^**-irreducible* $\text{cf}_{\gamma^*}(A) \in \text{Terms}$, *s.th.:*

1. *for some explicit,* γ^**-irreducible* $A_0, \dots, A_n \in \text{Terms}$ *(*$n \geq 0$*)*

$$\text{cf}_{\gamma^*}(A) \equiv A_0 \text{ where } \{p_1 := A_1, \dots, p_n := A_n\}$$

2. $A \Rightarrow^*_{\gamma^*} \text{cf}_{\gamma^*}(A)$
3. *for every* B, *such that* $A \Rightarrow^*_{\gamma^*} B$ *and* B *is* γ^**-irreducible, it holds that* $B \equiv_c \text{cf}_{\gamma^*}(A)$
   i.e., $\text{cf}_{\gamma^*}(A)$ *is unique, up to congruence*
4. $\text{Consts}(\text{cf}_{\gamma^*}(A)) = \text{Consts}(A)$ *and*
5. $\text{FreeV}(\text{cf}_{\gamma^*}(A)) = \text{FreeV}(A)$

Proof.

The proof is by induction on term structure of A, (5a)–(5e), (5f), using reduction rules, definitions, and properties of reduction.
The reduction rules and their applications do not remove and do not add any constants and free variables. □

Algorithmic Semantics of $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ / L^{λ}_{r}

How is the algorithmic meaning / semantics of a proper (non-immediate) $A \in$ Terms determined?

- For every term $A \in$ Terms, by the Canonical Form Theorem 5:

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

$$A \Rightarrow_{\gamma^*} \mathsf{cf}_{\gamma^*}(A)$$

- For each proper (i.e., non-immediate) $A \in$ Terms, $\mathsf{cf}(A)$ / $\mathsf{cf}_{\gamma^*}(A)$ determines the algorithm $\mathsf{alg}(A)$ for computing $\mathsf{den}(A)$

Theorem (Effective Reduction Calculi)

For every term $A \in$ Terms, its canonical forms $\mathsf{cf}(A)$ and $\mathsf{cf}_{\gamma^*}(A)$ are effectively computed, by the extended reduction calculus.

Definition (of Algorithmic Equivalence / Synonymy)

Two terms $A, B \in$ Terms are algorithmically equivalent, $A \approx B$, in a given semantic structure $\mathfrak{A}$, i.e., referentially synonymous in $\mathfrak{A}$, iff

- A and B are both immediate, or
- A and B are both proper

and there are explicit, irreducible terms (of appropriate types), $A_0, \ldots, A_n, B_0, \ldots, B_n \in$ Terms, $(n \geq 0)$ such that:

1. $A \Rightarrow_{\mathsf{cf}} A_0$ where $\{ p_1 := A_1, \ldots, p_n := A_n \} \equiv \mathsf{cf}(A)$
2. $B \Rightarrow_{\mathsf{cf}} B_0$ where $\{ p_1 := B_1, \ldots, p_n := B_n \} \equiv \mathsf{cf}(B)$
3. for all $i \in \{ 0, \ldots, n \}$
   a. for every $x \in \mathsf{PureV} \cup \mathsf{RecV}$,
   $$x \in \mathsf{FreeV}(A_i) \quad \text{iff} \quad x \in \mathsf{FreeV}(B_i) \tag{25}$$
   b. $\mathsf{den}(A_i) = \mathsf{den}(B_i)$

Type Theory L^{λ}_{ar} / L^{λ}_{r} is more expressive than Gallin TY2

Theorem (Moschovakis [17] 2006, §3.24 (mild adjustment))

1. *For any explicit (λ-calculus) $A \in$ Terms, there is no (assignment) memory location, bound via where in its canonical form, such that it occurs in more than one of its parts A_i $(0 \leq i \leq n)$ of* $\mathsf{cf}(A)$ / $\mathsf{cf}_{\gamma^*}(A)$
2. *Assume that $A \in$ Terms is such that an assignment location $p \in$ RecV, bound by where in its canonical form* $\mathsf{cf}(A)$ / $\mathsf{cf}_{\gamma^*}(A)$, *occurs in (at least) two assignment parts, and the denotations of those parts depend essentially on p:*
   Then, there is no explicit (λ-calculus) term $B \in$ Terms, such that B is algorithmically equivalent to A, $B \approx A$,
   i.e., for all λ-calculus $B \in$ Terms, $B \not\approx A$.

The proof is by Moschovakis [17] (2006). I provide it for the extended L^{λ}_{ar} / L^{λ}_{r}

Reductions with Pure Quantifier Rules: Algorithmic Patterns and Instantiations

- Assume $cube, large_0 \in \mathsf{Consts}_{(\widetilde{e} \to \widetilde{t})}$, in the typical Aristotelian form:

$$\text{Some cube is large} \xrightarrow{\text{render}} B \equiv \exists x(cube(x) \land large_0(x)) \tag{26a}$$

$$B \Rightarrow \exists x((c \land l) \text{ where } \{\, c := cube(x),\ l := large_0(x) \,\}) \tag{26b}$$

by 2 x (ap) (ap-comp), (recap), (wh-comp), (head), (lq-comp)

$$\Rightarrow \underbrace{\exists x(c'(x) \land l'(x))}_{B_0 \text{ algorithmic pattern}} \text{ where } \{ \tag{26c}$$

$$\underbrace{c' := \lambda(x)(cube(x)),\ l' := \lambda(x)(large_0(x))}_{\text{instantiations of memory slots } c',\ l'} \} \equiv \mathsf{cf}(B) \tag{26d}$$

from (26c), by (ξ) to $\exists$

$$\approx \underbrace{\exists x(c'(x) \land l'(x))}_{B_0 \text{ algorithmic pattern}} \text{ where } \{ \underbrace{c' := cube,\ l' := large_0}_{\text{instantiations of memory slots } c',\ l'} \} \equiv B' \tag{26e}$$

$$\begin{aligned} &\text{by Def. 7 from (26c)–(26d)},\ \mathsf{den}(\lambda(x)(cube(x))) = \mathsf{den}(cube), \\ &\qquad \mathsf{den}(\lambda(x)(large_0(x))) = \mathsf{den}(large_0) \end{aligned} \tag{26f}$$

$$\text{Some cube is large} \xrightarrow{\text{render}} T, \quad \mathit{large} \in \mathsf{Consts}_{((\tilde{\mathsf{e}}\to\tilde{\mathsf{t}})\to(\tilde{\mathsf{e}}\to\tilde{\mathsf{t}}))} \tag{27a}$$

$$T \equiv \exists x\big[\mathit{cube}(x) \wedge \underbrace{[\mathit{large}(\mathit{cube})](x)}_{\text{by predicate modification}}\big] \Rightarrow \dots \tag{27b}$$

$$\Rightarrow \exists x\big[(c_1 \wedge l) \text{ where } \{\, c_1 := \mathit{cube}(x), \tag{27c}$$

$$l := \mathit{large}(c_2)(x),\ c_2 := \mathit{cube}\,\}\big] \tag{27d}$$

$$\Rightarrow \exists x(c_1'(x) \wedge l'(x)) \text{ where } \{\, c_1' := \lambda(x)(\mathit{cube}(x)), \tag{27e}$$

$$l' := \lambda(x)(\mathit{large}(c_2'(x))(x)),\ c_2' := \lambda(x)\mathit{cube}\,\} \tag{27f}$$

$$\equiv \mathsf{cf}(T) \qquad \text{(27e)–(27f) is by } (\xi) \text{ on (27c)–(27d)}$$

$$\Rightarrow_{\gamma^*} \exists x(c_1'(x) \wedge l'(x)) \text{ where } \{\, c_1' := \lambda(x)(\mathit{cube}(x)), \tag{27g}$$

$$l' := \lambda(x)(\mathit{large}(c_2)(x)),\ c_2 := \mathit{cube}\,\} \tag{27h}$$

$$\equiv \mathsf{cf}_{\gamma^*}(T)$$

$$\approx \exists x(c_1'(x) \wedge l'(x)) \text{ where } \{\, c_1' := \mathit{cube}, \tag{27i}$$

$$l' := \lambda(x)(\mathit{large}(c_2)(x)),\ c_2 := \mathit{cube}\,\} \tag{27j}$$

$$\text{Some cube is large} \xrightarrow{\text{render}} C, \quad \mathit{large} \in \mathsf{Consts}_{((\widetilde{\mathsf{e}}\to\widetilde{\mathsf{t}})\to(\widetilde{\mathsf{e}}\to\widetilde{\mathsf{t}}))}$$

$$C \equiv \underbrace{\exists x\big[c'(x) \land \mathit{large}(c')(x)\big]}_{E_0} \text{ where } \{\, c' := \mathit{cube} \,\} \tag{28a}$$

$$\Rightarrow \underbrace{\exists x\Big[\big(c'(x) \land l\big) \text{ where } \{\, l := \mathit{large}(c')(x) \,\}\Big]}_{E_1} \text{ where } \{\, c' := \mathit{cube} \,\} \tag{28b}$$

from (28a), by (ap) to $\land$ of E_0; (lq-comp); (wh-comp)

$$\Rightarrow \Big[\underbrace{\exists x\big(c'(x) \land l'(x)\big) \text{ where } \{\, l' := \lambda(x)\big(\mathit{large}(c')(x)\big) \,\}}_{E_2}\Big] \text{ where } \{\, c' := \mathit{cube} \,\} \tag{28c}$$

from (28b), by (ξ) to $\exists$

$$\Rightarrow \underbrace{\exists x\big(c'(x) \land l'(x)\big)}_{C_0 \text{ an algorithmic pattern}} \text{ where } \{ \underbrace{c' := \mathit{cube},\ l' := \lambda(x)\big(\mathit{large}(c')(x)\big)}_{\text{instantiations of memory } c',\, l'} \} \equiv \mathsf{cf}(C) \tag{28d}$$

from (28c), by (head); (cong)

Proposition

1. *The $\mathrm{L}_{\mathrm{ar}}^{\lambda}$-terms $C \approx \mathsf{cf}(C)$ in (28a)–(28d), and many other $\mathrm{L}_{\mathrm{ar}}^{\lambda}$-terms, are not algorithmically equivalent to any explicit terms*
2. *$\mathrm{L}_{\mathrm{ar}}^{\lambda}$ is a strict, proper extension of TY_2, Gallin [3]*
3. *and of a la Montague semantics via inclusion of Montague IL in TY_2*

Outline of a proof:
(1) follows by Theorem 8
(2) follows by Theorem 8, and (1)
(3) Gallin [3] provides an interpretation of Montague IL (MIL) [19] into TY_2. Suitable interpretation of MIL can be given directly in $\mathrm{L}_{\mathrm{ar}}^{\lambda}$ (L_{r}^{λ}).

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

$$\text{Montague IL} \subsetneq \text{Gallin } \mathrm{TY}_2 \subsetneq \text{Moschovakis } \mathrm{L}_{\mathrm{ar}}^{\lambda} \subsetneq \text{Moschovakis } \mathrm{L}_{r}^{\lambda} \tag{29}$$

Generalised Two-Argument Quantifiers: $Q : ((\widetilde{\mathsf{e}} \to \widetilde{\mathsf{t}}) \to ((\widetilde{\mathsf{e}} \to \widetilde{\mathsf{t}}) \to \widetilde{\mathsf{t}}))$

$$\text{some, every} \xrightarrow{\text{render}} \mathit{some}, \mathit{every} \in \mathsf{Consts}_{[(\widetilde{\mathsf{e}} \to \widetilde{\mathsf{t}}) \to ((\widetilde{\mathsf{e}} \to \widetilde{\mathsf{t}}) \to \widetilde{\mathsf{t}})]} \tag{30}$$

$$[\text{some}_{\mathrm{DET}}\ \text{cube}_{\mathrm{N}}]_{\mathrm{NP}} \xrightarrow{\text{render}} \mathit{some}(\mathit{cube}) : ((\widetilde{\mathsf{e}} \to \widetilde{\mathsf{t}}) \to \widetilde{\mathsf{t}}) \tag{31}$$

$$\Rightarrow_{\mathsf{cf}} \big[\mathit{some}(d) \text{ where } \{ d := \mathit{cube} \}\big] \tag{32}$$

$$\text{Some cube is large} \xrightarrow{\text{render}} A_0/A_1/A_2 \quad \text{(options)} \tag{33a}$$

$$A_0 \equiv \big(\mathit{some}(\mathit{cube})\big)(\mathit{large}_0) : \widetilde{\mathsf{t}} \qquad \text{typical } \lambda\text{-term} \tag{33b}$$

$$\Rightarrow_{\mathsf{cf}} \underbrace{\mathit{some}(p_1)(p_2) \ \text{ where } \{p_1 := \mathit{cube},\ p_2 := \mathit{large}_0\}}_{\text{recursion term}} \tag{33c}$$

$$A_1 \equiv \mathit{some}(p_1)(p_2) \text{ where } \{p_1 := \mathit{cube},\ p_2 := \mathit{large}(p_1)\} \tag{33d}$$

$$A_2 \equiv \underbrace{Q(p_1)(p_2)}_{\text{alg. pattern}} \text{ where } \{ \underbrace{Q := \mathit{some},\ p_1 := \mathit{cube},\ p_2 := \mathit{large}(p_1)}_{\text{instantiations of memory}} \} \tag{33e}$$

Alternatives: $Q := \mathit{every}$, $Q := \mathit{one}$, $Q := \mathit{two}$, $Q := \mathit{most}$, etc.

No explicit terms are algorithmically equivalent to A_1 and A_2, by Th. 8.

$$[[\mathsf{K}]_{\mathrm{NP}}\ [\mathsf{is}\ [\mathsf{larger}_{\mathrm{ADJ}}\ \mathsf{than}\ [\mathsf{some}_{\mathrm{DET}}\ \mathsf{number}_{\mathrm{N}}]_{\mathrm{NP}}]_{\mathrm{ADJP}}]_{\mathrm{VP}}]_{\mathrm{S}} \xrightarrow{\mathsf{render}} A \tag{34a}$$

$$A \equiv \Big[\lambda y \big[\big[some(number)\big]\big(\lambda x_d\, larger(x_d)(y)\big)\big]\Big]_{\mathrm{VP}}(K) \Rightarrow \ \ldots \tag{34b}$$

$$\Rightarrow \Big[\lambda(y_k)\Big(some\big(d'(y_k)\big)\big(h(y_k)\big)\Big) \text{ where } \{\, d' := \lambda(y_k)number,\ h := \lambda(y_k)\lambda(x_d)larger(x_d)(y_k)\,\}\Big](K) \tag{34c}$$

$$\cdots \Rightarrow_{\mathsf{cf}} \mathsf{cf}(A) \equiv \tag{34d}$$

$$\Big[\lambda(y_k)\Big(some\big(d'(y_k)\big)\big(h(y_k)\big)\Big)\Big](k) \text{ where } \{\, h := \lambda(y_k)\lambda(x_d)larger(x_d)(y_k),\ \color{red}{d' := \lambda(y_k)number},\ k := K\,\} \tag{34e}$$

$$\Rightarrow_{\gamma^*} \mathsf{cf}_{\gamma^*}(A) \equiv \tag{34f}$$

$$\big[\lambda(y_k)some(d)\big(h(y_k)\big)\big](k) \text{ where } \{\, h := \lambda(y_k)\lambda(x_d)larger(x_d)(y_k),\ \color{red}{d := number},\ k := K\,\} \tag{34g}$$

$$[[\mathsf{K}]_{\mathrm{NP}}\ [\mathsf{is}\ [\mathsf{larger}_{\mathrm{ADJ}}\ \mathsf{than}\ [\mathsf{some}_{\mathrm{DET}}\ \mathsf{number}_{\mathrm{N}}]_{\mathrm{NP}}]_{\mathrm{ADJP}}]_{\mathrm{VP}}]_{\mathrm{S}} \xrightarrow{\text{render}} \mathsf{cf}_{\gamma^*}(A) \tag{35a}$$

$$\mathsf{cf}_{\gamma^*}(A) \equiv \tag{35b}$$

$$\big[\lambda(y_k)some(d)\big(h(y_k)\big)\big](k) \text{ where } \{\, h := \lambda(y_k)\lambda(x_d)larger(x_d)(y_k),\ d := number,\ k := K\,\} \tag{35c}$$

$$\not\approx A' \equiv \big[some(d)\big(h(k)\big)\big] \text{ where } \{\, h := \lambda(y_k)\lambda(x_d)larger(x_d)(y_k),\ d := number,\ k := K\,\} \tag{35d}$$

$$\Rightarrow \mathsf{cf}_{\gamma^*}(A') \equiv \big[some(d)(h')\big] \text{ where } \{\, h' := h(k),\ h := \lambda(y_k)\lambda(x_d)larger(x_d)(y_k),\ d := number,\ k := K\,\} \tag{35e}$$

from (35d) by (ap), since $h(k)$ is not immediate, see Def. 2; (wh-comp), (head)

$\therefore$ Tentative β-repl. in (35c) does not preserve algorithmic equivalence by Def. 7

$$[K\ [\text{is}\ [\text{larger}_{\mathrm{ADJ}}\ \text{than}\ [\text{some}_{\mathrm{DET}}\ \text{number}_{\mathrm{N}}]_{\mathrm{NP}}]_{\mathrm{ADJP}}]_{\mathrm{VP}}]_{\mathrm{S}} \xrightarrow{\text{render}} A_3 \tag{36a}$$

$$A_3 \equiv \Big[\lambda y_k \big[\big[Q(number)\big]\big(\lambda x_d\, larger(x_d)(y_k)\big) \text{ where } \{ Q := some\}\big]\Big](K) \Rightarrow \ \dots \tag{36b}$$

$$\Rightarrow_{\gamma^*}\ \mathsf{cf}_{\gamma^*}(A_3) \equiv \big[\lambda(y_k)Q(d)\big(h(y_k)\big)\big](k) \text{ where } \{\, Q := some,\ h := \lambda(y_k)\lambda(x_d) larger(x_d)(y_k),\ d := number,\ k := K\,\} \tag{36c}$$

$$\not\approx A_3' \equiv \big[Q(d)\big({\color{red}h(k)}\big)\big] \text{ where } \{\, Q := some,\ h := \lambda(y_k)\lambda(x_d) larger(x_d)(y_k),\ d := number,\ k := K\,\} \tag{36d}$$

$$\Rightarrow\ \mathsf{cf}_{\gamma^*}(A_3') \equiv \big[Q(d)\big({\color{red}h'}\big)\big] \text{ where } \{\, Q := some,\ d := number,\ k := K,\ {\color{red}h' := h(k)},\ h := \lambda(y_k)\lambda(x_d) larger(x_d)(y_k)\,\} \tag{36e}$$

from (36d) by (ap), since ${\color{red}h(k)}$ is not immediate, see Def. 2; (wh-comp), (head)

de dicto and *de re* renderings of quantifiers share algorithmic pattern

$$\text{Every cube is larger than some dodeca} \xrightarrow{\text{render}} \qquad \text{(de dicto)}$$

$$\begin{aligned}
R_3 \text{ where } \{R_3 &:= every(p)(R_2), && (37a)\\
R_2 &:= \lambda(x_2)some(b)(R_1(x_2)), && (37b)\\
R_1 &:= \lambda(x_2)\lambda(x_1)larger(x_1)(x_2), && (37c)\\
p &:= cube,\ b := dodeca\} && (37d)
\end{aligned}$$

$$\text{Every cube is larger than some dodeca} \xrightarrow{\text{render}} \qquad \text{(de re)}$$

$$\begin{aligned}
R_3 \text{ where } \{R_3 &:= some(b)(R_1), && (38a)\\
R_1 &:= \lambda(x_1)every(p)(R_2(x_1)), && (38b)\\
R_2 &:= \lambda(x_1)\lambda(x_2)larger(x_1)(x_2), && (38c)\\
p &:= cube,\ b := dodeca\} && (38d)
\end{aligned}$$

de dicto term S_{21}

$$
\begin{aligned}
S_{21} \equiv R_3 \text{ where } \{\, R_3 &:= Q_2(R_2), && \text{(39a)}\\
R_2 &:= \lambda(x_2) Q_1(R_1^1(x_2)) && \text{(39b)}\\
R_1^1 &:= \lambda(x_2)\lambda(x_1) h(x_1)(x_2), && \text{(39c)}\\
Q_1 &:= q_1(d_1),\ Q_2 := q_2(d_2), && \text{(39d)}\\
q_2 &:= \mathit{every},\ d_2 := \mathit{cube}, && \text{(39e)}\\
q_1 &:= \mathit{some},\ d_1 := \mathit{dodeca},\ h := \mathit{larger}\,\} && \text{(39f)}
\end{aligned}
$$

de re term S_{12}

$$
\begin{aligned}
S_{12} \equiv R_3 \text{ where } \{\, R_3 &:= Q_1(R_1), && \text{(40a)}\\
R_1 &:= \lambda(x_1) Q_2(R_2^1(x_1)), && \text{(40b)}\\
R_2^1 &:= \lambda(x_1)\lambda(x_2) h(x_1)(x_2), && \text{(40c)}\\
Q_1 &:= q_1(d_1),\ Q_2 := q_2(d_2), && \text{(40d)}\\
q_2 &:= \mathit{every},\ d_2 := \mathit{cube}, && \text{(40e)}\\
q_1 &:= \mathit{some},\ d_1 := \mathit{dodeca},\ h := \mathit{larger}\,\} && \text{(40f)}
\end{aligned}
$$

Constrained Underspecified Terms

$$
\begin{aligned}
U \equiv R_3 \text{ where } \{\, & l_1 := Q_1(R_1),\ l_2 := Q_2(R_2), && \text{(41a)}\\
& Q_1 := q_1(d_1),\ Q_2 := q_2(d_2), && \text{(41b)}\\
& q_1 := some,\ q_2 := every, && \text{(41c)}\\
& h := larger,\ d_1 := dodeca,\ d_2 := cube \,\} && \text{(41d)}\\
\text{s.t. } \{\, & Q_i \text{ binds the } i\text{-th argument of } h, && \text{(41e)}\\
& R_3 \text{ binds (dominates) each } Q_i\ (i = 1, 2) \,\} && \text{(41f)}
\end{aligned}
$$

- U is underspecified (per se), while restricted:
  R_3, $R_i (i = 1, 2)$ are free recursion variables
- the specifications in U have to satisfy the constraints on argument bindings

de dicto rendering of the quantifiers after specification of the underspecified pattern U

U can be specified to *de dicto* term:

$$
\begin{aligned}
U_{21} \equiv R_3 \text{ where } \{\, & R_3 := l_2,\ l_2 := Q_2(R_2), && \text{(42a)}\\
& R_2 := \lambda(x_2) l_1^1(x_2),\ l_1^1 := \lambda(x_2) Q_1(R_1^1(x_2)), && \text{(42b)}\\
& R_1^1 := \lambda(x_2)\lambda(x_1) h(x_1)(x_2), && \text{(42c)}\\
& Q_1 := q_1(d_1),\ Q_2 := q_2(d_2), && \text{(42d)}\\
& q_2 := \mathit{every},\ d_2 := \mathit{cube}, && \text{(42e)}\\
& q_1 := \mathit{some},\ d_1 := \mathit{dodeca},\ h := \mathit{larger}\,\} && \text{(42f)}
\end{aligned}
$$

U_{21} can be simplified to the similar, chain-free term by (chain) rule:

$$
\begin{aligned}
S_{21} \equiv R_3 \text{ where } \{\, & R_3 := Q_2(R_2),\ R_2 := \lambda(x_2) Q_1(R_1^1(x_2)), && \text{(43a)}\\
& R_1^1 := \lambda(x_2)\lambda(x_1) h(x_1)(x_2), && \text{(43b)}\\
& Q_1 := q_1(d_1),\ Q_2 := q_2(d_2), && \text{(43c)}\\
& d_2 := \mathit{cube},\ d_1 := \mathit{dodeca},\ h := \mathit{larger}, && \text{(43d)}\\
& q_2 := \lambda(U)\lambda(V)\forall(x)\big[U(x) \rightarrow V(x)\big], && \text{(43e)}\\
& q_1 := \lambda(U)\lambda(V)\exists(x)\big[U(x) \wedge V(x)\big]\,\} && \text{(43f)}
\end{aligned}
$$

U can be specified to the *de re* term:

$$\begin{aligned} U_{12} \equiv R_3 \text{ where } \{\, & R_3 := l_1,\ l_1 := Q_1(R_1), && \text{(44a)}\\ & R_1 := \lambda(x_1) l_2^1(x_1),\ l_2^1 := \lambda(x_1) Q_2(R_2^1(x_1)), && \text{(44b)}\\ & R_2^1 := \lambda(x_1)\lambda(x_2) h(x_1)(x_2), && \text{(44c)}\\ & Q_1 := q_1(d_1),\ Q_2 := q_2(d_2), && \text{(44d)}\\ & q_2 := \mathit{every},\ d_2 := \mathit{cube}, && \text{(44e)}\\ & q_1 := \mathit{some},\ d_1 := \mathit{dodeca},\ h := \mathit{larger}\,\} && \text{(44f)} \end{aligned}$$

U_{12} can be simplified to the similar, chain-free term by (chain) rule:

$$\begin{aligned} S_{12} \equiv R_3 \text{ where } \{\, & R_3 := Q_1(R_1),\ R_1 := \lambda(x_1) Q_2(R_2^1(x_1)), && \text{(45a)}\\ & R_2^1 := \lambda(x_1)\lambda(x_2) h(x_1)(x_2), && \text{(45b)}\\ & Q_1 := q_1(d_1),\ Q_2 := q_2(d_2), && \text{(45c)}\\ & d_2 := \mathit{cube},\ d_1 := \mathit{dodeca},\ h := \mathit{larger}, && \text{(45d)}\\ & q_2 := \lambda(U)\lambda(V)\forall(x)\big[U(x) \rightarrow V(x)\big], && \text{(45e)}\\ & q_1 := \lambda(U)\lambda(V)\exists(x)\big[U(x) \wedge V(x)\big]\,\} && \text{(45f)} \end{aligned}$$

Combinatorial Permutations of Quantifier Scopes

$$
\begin{aligned}
Q &\equiv R_{(n+1)} \text{ where } \{ && (46a)\\
R^{(n-1)}_{\pi(n)} &:= \lambda(x_{\pi(1)}) \ldots \lambda(x_{\pi(n)}) h(x_1) \ldots (x_n), && (46b)\\
&\ldots \\
R^{(j-1)}_{\pi(j)} &:= \lambda(x_{\pi(1)}) \ldots \lambda(x_{\pi(j)}) Q_{\pi(j+1)}[&& (46c)\\
&\quad \lambda(x_{\pi(j+1)}) R^{j}_{\pi(j+1)}(x_{\pi(1)}) \ldots (x_{\pi(j)})(x_{\pi(j+1)})] && (46d)\\
&\quad \mid \text{ for } j = (n-1), \ldots, 1, \\
&\ldots \\
R_{\pi(1)} &:= \lambda(x_{\pi(1)}) Q_{\pi(2)}[\lambda(x_{\pi(2)}) R^{1}_{\pi(2)}(x_{\pi(1)})(x_{\pi(2)})], && (46e)\\
R_{(n+1)} &:= Q_{\pi(1)}[\lambda(x_{\pi(1)}) R_{\pi(1)}(x_{\pi(1)})], && (46f)\\
Q_i &:= [q_i(d_i) \text{ s.t. } \{ Q_i \text{ binds the } i\text{-th argument of } h \ \}] && \\
&\qquad \mid \text{ for } i = 1, \ldots, n;\ n \geq 1 \, \} && (46g)
\end{aligned}
$$

Underspecified Quantifiers

$$
\begin{aligned}
V \equiv R_4 \text{ where } \{ l_i &:= [Q_i(R_i), && \text{(47a)}\\
&\text{s.t. } \{ Q_i\ \lambda\text{-binds the } i\text{-th argument of } h \text{ via } R_i, && \text{(47b)}\\
&\quad h : (\widetilde{\mathsf{e}} \to (\widetilde{\mathsf{e}} \to \ldots (\widetilde{\mathsf{e}} \to \widetilde{\mathsf{t}}))), && \text{(47c)}\\
&\quad R_4 \text{ is assigned to a closed subterm with} && \text{(47d)}\\
&\quad \text{fully scope specified } Q_i, &&\\
&\quad R_4 \text{ dominates each } Q_i \text{ (for } i = 1, \ldots, n) \}] &&\\
Q_i &:= q_i(d_i) && \text{(47e)}\\
&\quad \mid \text{for } i = 1, \ldots, n;\ n \geq 1 \} && \text{(47f)}
\end{aligned}
$$

$$\Phi \equiv \text{The cube is large} \tag{48}$$

- First Order Logic (FOL) A

$$\Phi \xrightarrow{\text{render}} A \equiv \exists x \big[\underbrace{\forall y (cube(y) \leftrightarrow x = y)}_{uniqueness} \ \wedge \ isLarge(x)\big] \tag{49a}$$

$$S \equiv \exists x \big[\underbrace{\forall y (P(y) \leftrightarrow x = y)}_{uniqueness} \ \wedge \ Q(x)\big] \tag{49b}$$

In FOL, A in (49a) has the following features:

- Existential quantification as the direct, topmost predication
- Uniqueness of the existing entity
- There is no referential force to the object denoted by the descriptor NP: $[\text{the cube}]_{\text{NP}}$
- There is no compositional analysis, i.e., no "derivation", of A from the components of Φ

- Higher Order Logic (HOL): Henkin (1950) and Mostowski (1957)
  a significant, positive step of compositional analyses, via β
  but no referential force!

$$\text{the} \xrightarrow{\text{render}} T \equiv \left[\lambda P \lambda Q \left[\exists x \left[\underbrace{\forall y (P(y) \leftrightarrow x = y)}_{uniqueness} \; \wedge \; Q(x)\right]\right]\right] \tag{50a}$$

$$\text{the cube} \xrightarrow{\text{render}} C \equiv T(cube)$$

$$C \equiv \left[\lambda P \lambda Q \left[\exists x \left[\underbrace{\forall y (P(y) \leftrightarrow x = y)}_{uniqueness} \; \wedge \; Q(x)\right]\right]\right](cube) \tag{50b}$$

$$\models\mid D \equiv \lambda Q \left[\exists x \left[\underbrace{\forall y (cube(y) \leftrightarrow x = y)}_{uniqueness} \; \wedge \; Q(x)\right]\right] \tag{50c}$$

(from (50b) by β-reduction)

$$\Phi \equiv \text{The cube is large} \xrightarrow{\text{render}} B \equiv D(isLarge) \tag{51a}$$

$$B \equiv \left[\lambda Q \left[\exists x \left[\underbrace{\forall y (cube(y) \leftrightarrow x = y)}_{uniqueness} \; \wedge \; Q(x)\right]\right]\right](isLarge) \tag{51b}$$

$$\models\mid \exists x \left[\underbrace{\forall y (cube(y) \leftrightarrow x = y)}_{uniqueness} \; \wedge \; isLarge(x)\right] \tag{51c}$$

(from (51b) by β-reduction)

Example: rendering of the definite article "the" Option 1

- Rendering the definite article "the" to a constant:

$$\text{the} \xrightarrow{\text{render}} \mathit{the} \in \mathsf{Consts}_{((\widetilde{\mathsf{e}} \to \widetilde{\mathsf{t}}) \to \widetilde{\mathsf{e}})} \tag{52}$$

- together with the following denotation of the constant the requiring "uniqueness" of the denoted object having the property $\bar{p}$ in s_0:

$$\big[\big(\mathsf{den}(\mathit{the})\big)(g)\big](\bar{p})(s_0) = \begin{cases} y, & \text{if } y \text{ is the unique } y \in \mathbb{T}_{\mathsf{e}}, \\ & \quad \text{for which } \bar{p}(s \mapsto y)(s_0) = 1 \\ \mathsf{er}, & \text{otherwise} \\ & \quad \text{i.e., there is no unique entity} \\ & \quad \text{that has the property } \bar{p} \text{ in } s_0 \end{cases} \tag{53}$$

for every $\bar{p} \in \mathbb{T}_{(\widetilde{\mathsf{e}} \to \widetilde{\mathsf{t}})}$ and every $s_0 \in \mathbb{T}_{\mathsf{s}}$

There are other possibilities for rendering the definite article "the", e.g., see Loukanova [13].

Option 3: the definite determiner "the" and descriptors: Underspecification

We can render "the" to A_1 or $\mathrm{cf}(A_1)$, underspecified for p:

$$\text{the} \xrightarrow{\text{render}} A_1 \equiv \big(q \text{ s.t. } \{\, unique(p)(q) \,\}\big) : \widetilde{\mathsf{e}} \tag{54a}$$

$$\text{the} \xrightarrow{\text{render}} \mathrm{cf}(A_1) \equiv \big(q \text{ s.t. } \{\, U \,\}\big) \text{ where } \{\, U := unique(p)(q) \,\} \tag{54b}$$

$$p \in \mathrm{RecV}_{(\widetilde{\mathsf{e}} \to \widetilde{\mathsf{t}})},\ q \in \mathrm{RecV}_{\widetilde{\mathsf{e}}},\ \text{by (st1) from (54a)} \tag{54c}$$

- q is the unique object having the property p, whoever it turns to be, by the predication $unique(p)(q)$

$$\text{the cube} \xrightarrow{\text{render}} \mathrm{cf}(A_2) : \widetilde{\mathsf{e}} \tag{55a}$$

$$A_2 \equiv \big(q \text{ s.t. } \{\, unique(p)(q) \,\}\big) \text{ where } \{\, p := cube \,\} \tag{55b}$$

$$\Rightarrow_{\mathrm{cf}} \mathrm{cf}(A_2) \equiv \big(q \text{ s.t. } \{\, U \,\}\big) \text{ where } \{\, U := unique(p)(q),\ p := cube \,\} \tag{55c}$$

by (st1), (head), from (55b)

$$\text{The cube is large} \xrightarrow{\text{render}} \mathsf{cf}(A_3) : \widetilde{\mathsf{t}} \tag{56a}$$

$$A_3 \equiv large(p)\Big(\big(q \text{ s.t. } \{\, unique(p)(q) \,\}\big) \text{ where } \{\, p := cube \,\}\Big) \tag{56b}$$

$$\begin{aligned} &\Rightarrow large(p)(Q) \text{ where } \{ \\ &\quad Q := \big[\big(q \text{ s.t. } \{\, unique(p)(q) \,\}\big) \text{ where } \{\, p := cube \,\}\big] \,\} \end{aligned} \tag{56c}$$

by (ap), from (56b)

$$\begin{aligned} \Rightarrow_{\mathsf{cf}} \mathsf{cf}(A_3) \equiv large(p)(Q) \text{ where } \{\, &Q := (q \text{ s.t. } \{\, U \,\}), \\ &U := unique(p)(q),\ p := cube \,\} \end{aligned} \tag{56d}$$

by (st1), (wh-comp), (B-S), from (55c), (56c)

Algorithmic Pattern: definite descriptors in predicative statements: Opt3

$$A \equiv L(Q) \text{ where } \{\, Q := (q \text{ s.t. } \{\, U \,\}), U := unique(p)(q) \,\} \tag{57a}$$

$$p, q, L \in \mathsf{FreeV}(A),\ p \in \mathsf{RecV}_{(\widetilde{\mathsf{e}} \to \widetilde{\mathsf{t}})},\ q \in \mathsf{RecV}_{\widetilde{\mathsf{e}}}, \tag{57b}$$

$$Q \in \mathsf{RecV}_{\widetilde{\mathsf{e}}},\ U \in \mathsf{RecV}_{\widetilde{\mathsf{t}}},\ L \in \mathsf{RecV}_{(\widetilde{\mathsf{e}} \to \widetilde{\mathsf{t}})} \tag{57c}$$

$$\text{The number } n \text{ is odd} \xrightarrow{\text{render}} \mathsf{cf}(A_4) : \widetilde{\mathsf{t}} \tag{58a}$$

$$A_4 \equiv isOdd\Big((q \text{ s.t. } \{\, unique(N)(q),\ p(q)\,\})\ \text{where}\ \{ \\ q := n,\ p := number,\ N := named\text{-}n\,\}\Big) \tag{58b}$$

$$\Rightarrow_{\mathsf{cf}}\ \mathsf{cf}(A_4) \equiv isOdd(Q)\ \text{where}\ \{\, Q := (q \text{ s.t. } \{\, U, C\,\}), \\ U := unique(N)(q),\ C := p(q), \\ q := n,\ p := number,\ N := named\text{-}n\,\} \tag{58c}$$

- direct reference, by assignment; uniqueness and existence are consequences

$$\text{The number } n \text{ is large} \xrightarrow{\text{render}} \mathsf{cf}(A_5) : \widetilde{\mathsf{t}} \tag{59a}$$

$$A_5 \equiv isOdd\Big((q \text{ s.t. } \{\, p(q)\,\})\ \text{where}\ \{ \\ q := n,\ p := number\,\}\Big) \tag{59b}$$

$$\Rightarrow_{\mathsf{cf}}\ isOdd(Q)\ \text{where}\ \{\, Q := (q \text{ s.t. } \{\, C\,\}),\ C := p(q), \\ q := n,\ p := number\,\} \tag{59c}$$

Predication via Coordination in VP: e.g., a class of coordinated ADJ, V, VP, etc.

$$[\Phi_j]_{\text{NP}}\ \big[[\Theta_L \text{ and } \Psi_N]\ [W_w]_{\text{NP}}\big]_{\text{VP}} \tag{60a}$$

$$\xrightarrow{\text{render}} \underbrace{\lambda x_j \big[\lambda y_w \big(L(y_w)(x_j) \land N(y_w)(x_j)\big)(w)\big](j)}_{\text{algorithmic pattern with memory parameters } L,\ N,\ w,\ j} \tag{60b}$$

$$[\ [\text{The cube}]_j\ [\text{is}\ [\underbrace{\text{larger than and next to}}_{\text{adjectival coordination}} \tag{61a}$$

$$[\ [\text{its}]_j\ \text{predecessor}]_w]_{\text{ADJP}}]_{\text{VP}}]_{\text{S}} \xrightarrow{\text{render}} A \tag{61b}$$

$$A \equiv \lambda x_j \big[\lambda y_w \big(larger(y_w)(x_j) \land nextTo(y_w)(x_j)\big) \\ \big(predecessor(x_j)\big)\big](the(cube)) \tag{62a}$$

$$\Rightarrow_{\gamma^*} \underbrace{\lambda x_j \big[\lambda y_w \big(L''(x_j)(y_w) \land N''(x_j)(y_w)\big)(w'(x_j))\big](j)}_{\text{algorithmic pattern with memory parameters } L'',\ N'',\ w',\ j} \tag{62b}$$

$$\text{where } \{\, L'' := \lambda x_j \lambda y_w\, larger(y_w)(x_j), \\ N'' := \lambda x_j \lambda y_w\, nextTo(y_w)(x_j), \\ w' := \lambda x_j predecessor(x_j),\ j := the(c),\ c := cube \,\} \tag{62c}$$

2 x (ap) to $\land$, (ξ) for λ, (wh-comp), (recap), 2 x (ap), (B-S)

- The algorithmic semantics of (61a)–(61b), (62b)–(62c) is predication via VP
- The sentence (63a)–(63b) is a conjunction of propositions, i.e., a propositional conjunction
- The algorithmic semantics of (63a)–(63b) can be represented by $\mathrm{cf}_{\gamma^*}(B)$, in (64b)–(64c), i.e., by a conjunction of propositions: $(L \wedge N)$

$$\text{[The cube]}_j \text{ is larger than [[its]}_j \text{ predecessor]}_w \tag{63a}$$

$$\text{and [it]}_j \text{ is next to [it]}_w \xrightarrow{\text{render}} B \tag{63b}$$

$$B \equiv \big[\mathit{larger}(w)(j) \wedge \mathit{nextTo}(w)(j)\big] \text{ where } \{ \\ j := \mathit{the}(\mathit{cube}),\, w := \mathit{predecessor}(j)\,\} \tag{64a}$$

$$\Rightarrow_{\mathrm{cf}_{\gamma^*}} (L \wedge N) \text{ where } \{L := \mathit{larger}(w)(j),\, N := \mathit{nextTo}(w)(j), \\ w := \mathit{predecessor}(j), \\ j := \mathit{the}(c),\ c := \mathit{cube}\,\} \tag{64b}$$

$$\equiv \mathrm{cf}_{\gamma^*}(B) \tag{64c}$$

Computational Syntax-Semantics of NL by using L_{ar}^{λ} in GCBLG

For syntax-semantics interfaces of Natural Language (NL), I employ:

- Generalised Constraint-Based Lexicalized Grammar (GCBLG), see [7]
  GCBLG covers a variety of computational grammars, by representing major, common syntactic characteristics of a class of approaches to computational grammar, e.g.:
    - Head-Driven Phrase Structure Grammar (HPSG) [18]
      `https://langsci-press.org/catalog/book/478`
    - Lexical Functional Grammar (LFG) [1]
    - Categorial Grammar (CG) [2, 15]
    - Grammatical Framework (GF) [4] (tentatively)

Algorithms for Computing Denotations of Terms

Algorithmic Syntax-Semantics of L_{ar}^{λ} (L_{r}^{λ}) and Natural Language

$$\underbrace{\text{Syntax of } L_{ar}^{\lambda}\ (L_{r}^{\lambda}) \Longrightarrow \text{Algorithms for Computations} \Longrightarrow \text{Denotations}}_{\text{Semantics of } L_{ar}^{\lambda}(L_{r}^{\lambda})} \tag{65}$$

$$\underbrace{\text{Computational Syntax of NL}}_{\text{Computational Grammar}} \xrightarrow{\text{render}} L_{ar}^{\lambda} \tag{66}$$

$$\underbrace{\text{Computational Syntax of NL} \xrightarrow{\text{render}} L_{ar}^{\lambda}}_{\text{Computational Grammar: Syntax-Semantics Interface}} \tag{67}$$

Computational Syntax-Semantics of NL by using L^{λ}_{ar} in GCBLG

Generalised Constraint-Based Lexicalized Grammar (GCBLG) covers major syntactic categories of natural language, by linguistically motivated generalizations.

- The syntactic information is distributed among a hierarchy of types
- typed feature-value descriptions: Feature-Value Logics; Attribute-Value (ATV) Matrices
- The semantic representation in syntax-semantics composition and interface, is by the feature SEM and its recursive values

  SEM has typed values that encode recursion terms of L^{λ}_{ar}, alternatively, of DTTSitInfo
- Efficient and effective, computational rendering of NL expressions to γ^*-canonical forms, see Loukanova [6, 9, 8]

Computational Syntax-Semantics of NL by using L_{ar}^{λ} in GCBLG

Computational Grammar with Syntax-Semantics and Underspecification

For a given NL expression ϕ, its grammar analysis Φ, includes syntax-semantics interface, throughout its constituents

$$\Phi \xrightarrow{\text{render}} A \equiv \mathsf{cf}_{\gamma}(A) \tag{68}$$

Motivation for Type Theory $\mathrm{L}_{\mathrm{ar}}^{\lambda}$ and Outlook

- $\mathrm{L}_{\mathrm{ar}}^{\lambda}$ provides Computational Semantics with:
  - greater semantic distinctions than type-theoretic semantics by λ-calculi, e.g., in Montagovian grammars
- $\mathrm{L}_{\mathrm{ar}}^{\lambda}$ provides Parametric Algorithms
  Parameters can be instantiated depending on:
  - classes and sets of specific parts of speech, e.g., nouns, NPs, verbs, properties, relations, etc.
  - representing major semantic ambiguities and underspecification [6], at the object level of the formal language of $\mathrm{L}_{\mathrm{ar}}^{\lambda}$, without meta-language variables
- $\mathrm{L}_{\mathrm{ar}}^{\lambda}$ with logical operators and pure quantifiers can be used for:
  - Proof-theoretic computational semantics and reasoning
  - Inferences of semantic information
  - Canonical forms can be used by automatic provers and proof assistants

Looking Forward!

Some References I

- Bresnan, J.: Lexical-Functional Syntax.
  Blackwell Publishers, Oxford (2001)

- Buszkowski, W.: Mathematical Linguistics and Proof Theory.
  In: J. van Benthem, A. ter Meulen (eds.) Handbook of Logic and Language, pp. 683–736. North-Holland, Amsterdam (1997).
  DOI https://doi.org/10.1016/B978-044481714-3/50016-3.
  URL https://www.sciencedirect.com/science/article/pii/B9780444817143500163

- Gallin, D.: Intensional and Higher-Order Modal Logic: With Applications to Montague Semantics.
  North-Holland Publishing Company, Amsterdam and Oxford, and American Elsevier Publishing Company (1975).
  URL https://doi.org/10.2307/2271880

Some References II

The Grammatical Framework GF.
http://www.grammaticalframework.org.
Accessed 5 Jul 2024

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 III

- Loukanova, R.: An approach to functional formal models of constraint-based lexicalized grammar (CBLG).
  Fundamenta Informaticae **152**(4), 341–372 (2017).
  URL `https://doi.org/10.3233/FI-2017-1524`

- 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).

Some References IV

URL https://doi.org/10.1007/978-3-030-05453-3_18

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 V

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 VI

- 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

- Moortgat, M.: Categorial Type Logics.
  In: J. van Benthem, A. ter Meulen (eds.) Handbook of Logic and Language, pp. 93–177. Elsevier, Amsterdam (1997).
  URL https://doi.org/10.1016/B978-044481714-3/50005-9

- Moschovakis, Y.N.: The formal language of recursion.
  Journal of Symbolic Logic **54**(4), 1216–1252 (1989).
  URL https://doi.org/10.1017/S0022481200041086

Some References VII

- 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

- Müller, S., Abeillé, A., Borsley, R.D., Koenig, J.P. (eds.): Head-Driven Phrase Structure Grammar. No. 9 in Empirically Oriented Theoretical Morphology and Syntax. Language Science Press, Berlin (2024). DOI 10.5281/zenodo.13637708

- Thomason, R.H. (ed.): Formal Philosophy: Selected Papers of Richard Montague. Yale University Press, New Haven, Connecticut (1974)