Semantics of Propositional Attitudes in Type-Theory of Algorithms

Roussanka Loukanova

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

June 15, 2024
Proof Systems for Mathematics and Verification
June 14-15, 2024, EPFL, Lausanne

Outline

Type-Theory of Acyclic / Full Algorithms: $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ / L^{λ}_{r}, introduced by Moschovakis [10] (2006)

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 of $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ / L^{λ}_{r}: defined by (10+) reduction rules $A \Rightarrow B$
- The reduction calculus of $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ / L^{λ}_{r} is effective (by a theorem):
  For every $A \in$ Terms, there is unique, up to congruence, canonical form cf(A), s.th.:
  $$A \Rightarrow_{\mathsf{cf}} \mathsf{cf}(A)$$
- 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)

Development of Type-Theory of Full / Acyclic Algorithms: $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ / L^{λ}_{r} / DTTSI

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

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

$$\overset{?}{\subsetneq} \text{DTTSitInfo} \tag{2}$$

- 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]
- This talk is derived from Loukanova [7, 9]:
  - $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ / L^{λ}_{r} terms of propositional attitudes, including statements
  - Operators of Algorithmic Scope
  - ToScope for an unspecified, open scope $B \in \text{Terms}(\mathrm{L}^{\lambda}_{r})$: ToScope$(B)$
  - $\mathcal{C}$ for the closure of a specified scope $A \in \text{Terms}(\mathrm{L}^{\lambda}_{r})$: $\mathcal{C}(A)$
  - Extended reduction calculus of $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ / L^{λ}_{r} for the scope operators
- *Note:* Dependent Type Theory of Situated Info (DTTSI) is not here: partial relations without currying; dependent types; situations, ...

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

- Gallin Types (1975)

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

- Abbreviations

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

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

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

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

$$\mathsf{Consts}_\sigma = K_\sigma = \{\mathsf{c}^\sigma_0, \mathsf{c}^\sigma_1, \dots\} \qquad \text{(4a)}$$

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

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

$$\mathsf{PureV}_\sigma = \{v^\sigma_0, v^\sigma_1, \dots\} \qquad \text{(4d)}$$

$$\mathsf{RecV}_\sigma = \mathsf{MemoryV}_\sigma = \{p^\sigma_0, p^\sigma_1, \dots\} \qquad \text{(4e)}$$

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

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

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

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

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

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

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

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

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

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

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

Terms of TTA $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ acyclic recursion (L^{λ}_{r} full recursion) / conditions on well-formed terms

- Acyclicity Constraint (AC), for $\mathrm{L}^{\lambda}_{\mathrm{ar}}$:

$$\{\, p_1^{\sigma_1} := A_1^{\sigma_1}, \ldots, p_n^{\sigma_n} := A_n^{\sigma_n} \,\} \quad (n \geq 0) \tag{6a}$$

is acyclic sequence (system) iff (6b)

there is some rank: $\{p_1, \ldots, p_n\} \rightarrow \mathbb{N}$ (6c)

if p_j occurs freely in A_i, then rank$(p_i) >$ rank(p_j) (6d)

- A term of the form (5b) is called acyclic recursion term, in case its system of assignments satisfies the AC
- $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ is type theory of acyclic algorithms, in case Def. 1 is restricted to acyclic terms (5b)
- L^{λ}_{r} is type theory of algorithms with full recursion: without requiring that AC holds for all recursion terms

Definition (Explicit and λ-*Calculus* Terms)

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

Definition (Immediate and Proper Terms)

- The set ImT of immediate terms is defined by recursion (7)

$$T :\equiv V \mid p(v_1)\ldots(v_m) \mid \lambda(u_1)\ldots\lambda(u_n)p(v_1)\ldots(v_m) \tag{7}$$

  for $V \in$ Vars, $p \in$ RecV, $u_i, v_j, \in$ PureV,
  $i = 1, \ldots, n$, $j = 1, \ldots, m$, $(m, n \geq 0)$
- Every $A \in$ Terms that is not immediate is proper

$$\text{PrT} = (\text{Terms} - \text{ImT}) \tag{8}$$

Immediate terms do not carry algorithmic sense.

$$\text{Some cube is large} \xrightarrow{\text{render}} T, \quad \mathit{large} \in \mathsf{Consts}_{((\widetilde{\mathsf{e}} \to \widetilde{\mathsf{t}}) \to (\widetilde{\mathsf{e}} \to \widetilde{\mathsf{t}}))} \tag{9a}$$

$$T \equiv \exists x \big[\mathit{cube}(x) \land \underbrace{\mathit{large}(\mathit{cube})(x)}_{\text{by predicate modification}} \big] \Rightarrow \ldots \tag{9b}$$

from (9b), by (ap) incl. to $\land$; (lq-comp); (rec-comp), (rec-comp) (9c)

$$\Rightarrow \exists x \big[(c_1 \land l) \text{ where } \{ c_1 := \mathit{cube}(x), \tag{9d}$$

$$l := \mathit{large}(c_2)(x),\ c_2 := \mathit{cube} \} \big] \tag{9e}$$

$$\Rightarrow \exists x (c_1'(x) \land l'(x)) \text{ where } \{ c_1' := \lambda(x)(\mathit{cube}(x)), \tag{9f}$$

$$l' := \lambda(x)(\mathit{large}(c_2'(x))(x)),\ c_2' := \lambda(x)\mathit{cube} \} \tag{9g}$$

$\equiv \mathsf{cf}(T)$ (9f)–(9g) is by (ξ) on (9d)–(9e)

$$\Rightarrow_{\gamma^*} \exists x (c_1'(x) \land l'(x)) \text{ where } \{ c_1' := \lambda(x)(\mathit{cube}(x)), \tag{9h}$$

$$l' := \lambda(x)(\mathit{large}(c_2)(x)),\ c_2 := \mathit{cube} \} \tag{9i}$$

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

$$\approx \exists x (c_1'(x) \land l'(x)) \text{ where } \{ c_1' := \mathit{cube}, \tag{9j}$$

$$l' := \lambda(x)(\mathit{large}(c_2)(x)),\ c_2 := \mathit{cube} \} \tag{9k}$$

Some cube is large $\xrightarrow{\text{render}} C$, without repeated calcs in $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ / L^{λ}_{r}

$$C \equiv \underbrace{\exists x\big[c(x) \wedge large(c)(x)\big]}_{E_0} \text{ where } \{\, c := cube \,\} \Rightarrow \ \ldots \tag{10a}$$

$$\Rightarrow \underbrace{\exists x\Big[\big(c(x) \wedge l\big) \text{ where } \{\, l := large(c)(x) \,\}\Big]}_{E_1} \quad \text{where } \{\, c := cube \,\} \tag{10b}$$

from (10a) by 2x(ap) to $\wedge$ of E_0; (lq-comp) to $\exists$;(rec-comp); (B-S); (head)

$$\Rightarrow \underbrace{\Big[\exists x\big(c(x) \wedge l'(x)\big) \text{ where } \{\, l' := \lambda(x)\big(large(c)(x)\big) \,\}\Big]}_{E_2} \quad \text{where } \{\, c := cube \,\} \tag{10c}$$

from (10b), by (ξ) to $\exists$

$$\Rightarrow \underbrace{\exists x\big(c(x) \wedge l'(x)\big)}_{C_0 \text{ an algorithmic pattern}} \quad \text{where } \{\, \underbrace{c := cube,\ l' := \lambda(x)\big(large(c)(x)\big)}_{\text{instantiations of memory } c,\, l'} \,\} \equiv \mathrm{cf}(C) \tag{10d}$$

from (10c), by (head); (cong)

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

$$C \equiv \underbrace{\exists x\big[c(x) \land l_1(c)(x)\big]}_{E_0} \text{ where } \{\, c := cube, l_1 := large \,\} \tag{11a}$$

$$\begin{aligned} &\Rightarrow \underbrace{\exists x\big[\big(c(x) \land l_2\big) \text{ where } \{\, l_2 := l_1(c)(x) \,\}\big]}_{E_1} \\ &\qquad\qquad \text{where } \{\, c := cube,\ l_1 := large \,\} \end{aligned} \tag{11b}$$

from (11a), by (ap) to $\land$ of E_0; (lq-comp); (rec-comp)

$$\begin{aligned} &\Rightarrow \big[\underbrace{\exists x\big(c(x) \land l_2'(x)\big) \text{ where } \{\, l_2' := \lambda(x)\big(l_1(c)(x)\big) \,\}}_{E_2}\big] \\ &\qquad\qquad \text{where } \{\, c := cube,\ l_1 := large \,\} \end{aligned} \tag{11c}$$

from (11b), by (ξ) to $\exists$ in E_1; (rec-comp)

$$\begin{aligned} &\Rightarrow \underbrace{\exists x\big(c(x) \land l_2'(x)\big)}_{C_0 \text{ an algorithmic pattern}} \\ &\qquad \text{where } \{\, \underbrace{l_2' := \lambda(x)\big(l_1(c)(x)\big),\ c := cube,\ l_1 := large \,\}}_{\text{instantiations of memory } c,\ l_1,\ l'} \} \end{aligned} \tag{11d}$$

$\equiv \mathsf{cf}(C)$ from (11c), by (head)

- attitude / statment verbs, taking sentential complements in Verb Phrases (VP)

$$\tau_{pa} \equiv (\widetilde{\mathsf{t}} \to (\widetilde{\mathsf{e}} \to \widetilde{\mathsf{t}})) \tag{12a}$$

$$\begin{aligned} &\text{claim} \xrightarrow{\text{render}} claim : \tau_{pa}, \quad \text{state} \xrightarrow{\text{render}} state : \tau_{pa}, \\ &\text{know} \xrightarrow{\text{render}} know : \tau_{pa}, \quad \text{believe} \xrightarrow{\text{render}} believe : \tau_{pa}, \ \ldots \end{aligned} \tag{12b}$$

$$\textcolor{red}{\text{ToScope}} : (\widetilde{\mathsf{t}} \to (\mathsf{s} \to \widetilde{\mathsf{t}})), \quad \textcolor{red}{\mathcal{C}} : (\mathsf{t} \to \widetilde{\mathsf{t}}) \qquad \text{(operators)} \tag{12c}$$

- Let ϕ, ψ be NL expressions, such that:

$$\psi \text{ is an attitude verb, e.g., } \psi \equiv state \tag{13a}$$

$$[\psi]_{\mathrm{V}} \xrightarrow{\text{render}} B : \tau_{pa} \qquad [\phi]_{\mathrm{S}} \xrightarrow{\text{render}} A : \widetilde{\mathsf{t}} \text{ (proposition)} \tag{13b}$$

For fresh variables $\mathfrak{c} \in \mathsf{RecV}_{\tau_{pa}}$, $s_c \in \mathsf{RecV}_{\mathsf{s}}$, we set:

$$[\psi \text{ (that) } \phi\ ]_{\mathrm{VP}} \xrightarrow{\text{render}} \mathfrak{c}\big(\mathsf{ToScope}(A)(s_c)\big) \text{ where } \{\, \mathfrak{c} := B \,\} \tag{14a}$$

$$\Rightarrow \mathfrak{c}\big(q\big) \text{ where } \{\, q := \mathsf{ToScope}(A)(s_c),\ \mathfrak{c} := B \,\} : (\widetilde{\mathsf{e}} \to \widetilde{\mathsf{t}}) \tag{14b}$$

$$\langle \dot{s}_c, \mathsf{Jim}_j \text{ states that some cube is large } \rangle \xrightarrow{\text{render}} A \tag{15a}$$

$$\begin{aligned}
A \equiv \mathfrak{c}\Big(\mathsf{ToScope}\big[\exists x\big(c(x) \land l_2'(x)\big)&\\
\text{where } \{\, l_2' := \lambda(x)\big(l_1(c)(x)\big),&\qquad (15\text{b})\\
c := cube,\ l_1 := large \,\}\big](s_c)\Big)(j)&\\
\text{where } \{\, j := jim,\ \mathfrak{c} := states \,\}&\qquad (15\text{c})
\end{aligned}$$

$$\begin{aligned}
\Rightarrow \mathfrak{c}(q)(j) \text{ where}&\\
\{\, q := \mathsf{ToScope}\big[\exists x\big(c(x) \land l_2'(x)\big)&\\
\text{where } \{\, l_2' := \lambda(x)\big(l_1(c)(x)\big),&\qquad (15\text{d})\\
c := cube,\ l_1 := large \,\}\big](s_c),&\\
j := jim,\ \mathfrak{c} := states \,\}&\qquad (15\text{e})
\end{aligned}$$

$$\begin{aligned}
\Rightarrow_{\mathsf{cf}} \mathfrak{c}(q)(j) \text{ where}&\\
\{\, q := \mathcal{C}\big[\exists x\big(c(x) \land l_2'(x)\big)(s_{r_1})&\\
\text{where } \{\, l_2' := \lambda(x)\big(l_1(c)(x)\big),&\qquad (15\text{f})\\
c := cube,\ l_1 := large \,\}\big],&\\
j := jim,\ \mathfrak{c} := states \,\} \equiv \mathsf{cf}(A_1)&\qquad (15\text{g})
\end{aligned}$$

from (15d)–(15e) by (ScopeR), (rec-comp)

Instantiations and Alternative Scoping: More Efficiency

$$\langle \dot{s}_c, \text{Jim}_j \text{ states that some cube is large } \rangle \xrightarrow{\text{render}} A_i (i = 1, 2) \tag{16}$$

$$\begin{aligned} A_1 \equiv \mathfrak{c}(q)(j) \text{ where } \\ \{ q := \mathcal{C}\big[\exists x \big(c(x) \land l_2'(x)\big)(s_{r_1}) \\ \text{where } \{ l_2' := \lambda(x)\big(l_1(c)(x)\big), \\ c := cube,\ l_1 := large \}\big], \end{aligned} \tag{17a}$$

$$j := jim,\ \mathfrak{c} := states \} \equiv \text{cf}(A_1) \tag{17b}$$

$$\begin{aligned} A_2 \equiv \mathfrak{c}(q)(j) \text{ where } \\ \{ q := \mathcal{C}\big[\exists x \big(c(x) \land l_2'(x)\big)(s_{r_2}) \\ \text{where } \{ l_2' := \lambda(x)\big(l_1(c)(x)\big), \\ l_1 := large \}\big], \end{aligned} \tag{18a}$$

$$c := cube,\ j := jim,\ \mathfrak{c} := states \} \equiv \text{cf}(A_2) \tag{18b}$$

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

$$\text{some, every} \xrightarrow{\text{render}} \mathit{some}, \mathit{every} \in \mathsf{Consts}_{[(\widetilde{\mathsf{e}} \to \widetilde{\mathsf{t}}) \to ((\widetilde{\mathsf{e}} \to \widetilde{\mathsf{t}}) \to \widetilde{\mathsf{t}})]} \tag{19}$$

$$[\text{some}_{\mathrm{DET}}\ \text{cube}_{\mathrm{N}}]_{\mathrm{NP}} \xrightarrow{\text{render}} \mathit{some}(\mathit{cube}) : ((\widetilde{\mathsf{e}} \to \widetilde{\mathsf{t}}) \to \widetilde{\mathsf{t}}) \tag{20}$$

$$\Rightarrow_{\mathsf{cf}} \big[\mathit{some}(d) \text{ where } \{\, d := \mathit{cube} \,\}\big] \tag{21}$$

$$\text{Some cube is large} \xrightarrow{\text{render}} A_0/A_1/A_2 \quad \text{(options)} \tag{22a}$$

$$A_0 \equiv \big(\mathit{some}(\mathit{cube})\big)(\mathit{large}_0) : \widetilde{\mathsf{t}} \qquad \textcolor{red}{\text{typical } \lambda\text{-calculi term}} \tag{22b}$$

$$\Rightarrow_{\mathsf{cf}} \underbrace{\mathit{some}(p_1)(p_2) \ \text{ where } \{p_1 := \mathit{cube},\ p_2 := \mathit{large}_0\}}_{\textcolor{red}{\text{recursion term}}} \tag{22c}$$

$$A_1 \equiv \mathit{some}(p_1)(p_2) \text{ where } \{p_1 := \mathit{cube},\ p_2 := \mathit{large}(p_1)\} \tag{22d}$$

$$A_2 \equiv \underbrace{Q(p_1)(p_2)}_{\textcolor{red}{\text{alg. pattern}}} \text{ where } \{\underbrace{Q := \mathit{some},\ p_1 := \mathit{cube},\ p_2 := \mathit{large}(p_1)}_{\textcolor{red}{\text{instantiations of memory}}}\} \tag{22e}$$

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: proved

Reduction Rules (to be continued)

[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)

- Congruence Relation, informally
  The *congruence* relation is the smallest (equivalence) relation between L_{ar}^{λ}-terms, such that it is:
  - reflexive, symmetric, transitive
  - closed under:
    - operators of term formation
    - renaming bound variables (pure and recursion), without causing collisions
    - re-ordering of the assignments within the recursion terms
    - re-ordering of the restriction sub-terms in the restriction terms

[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

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

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

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

Compositionality of Scope Operators (in the extended $\mathrm{L}^{\lambda}_{\mathrm{ar}}$)

If $A \Rightarrow A'$ then $\mathsf{ToScope}(A) \Rightarrow \mathsf{ToScope}(A')$ (S-comp)

If $A \Rightarrow A'$ then $\mathcal{C}(A) \Rightarrow \mathcal{C}(A')$ ($\mathcal{C}$-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,

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

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

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

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

$$\big(A_0 \text{ where } \{\, \overrightarrow{p} := \overrightarrow{A} \,\}\big)(B) \Rightarrow A_0(B) \text{ where } \{\, \overrightarrow{p} := \overrightarrow{A} \,\} \qquad \text{(recap)}$$

Reduction Rules (to be continued)

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

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

[λ and Pure Quantifier Rules] Let $\xi \in \{\, \lambda, \exists, \forall \,\}$.
Given fresh $p_i' \in \big[\,\mathsf{RecV} - \big(\,\mathsf{FV}(A) \cup \mathsf{BV}(A)\big)\big]$, $i = 1, \ldots, n$, for $A \equiv A_0 \text{ where } \{\, p_1 := A_1, \ldots, p_n := A_n \,\}$ and replacements A_i' in (27):

$$A_i' \equiv \Big[A_i\{\, p_1 :\equiv p_1'(u), \ldots, p_n :\equiv p_n'(u) \,\}\Big] \tag{27}$$

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

- each $R_i^{\tau_i} \in$ Terms in $\overrightarrow{R}$ is immediate and $\tau_i \in \{\, \mathsf{t}, \widetilde{\mathsf{t}} \,\}$
- each $C_j^{\tau_j} \in$ Terms is proper and $\tau_j \in \{\, \mathsf{t}, \widetilde{\mathsf{t}} \}$ $(j = 1, \dots, m,\ m \geq 0)$
- $a_0, c_j \in$ RecV $(j = 1, \dots, m)$ fresh

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

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

(st2) Rule A_0 is a proper term

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

γ- and γ^*-Rules — stronger reduction

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: (30a)–(30c):

$$A \equiv A_0 \text{ where } \{ \overrightarrow{a} := \overrightarrow{A}, \tag{30a}$$

$$p := \lambda(\overrightarrow{u})\lambda(v)P, \tag{30b}$$

$$\overrightarrow{b} := \overrightarrow{B} \} \tag{30c}$$

such that the following holds:

1. $v \notin \mathrm{FreeVars}(P)$
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 $\lambda(v)$ (preserves the free occurrences of v) modulo renaming the variables $\overrightarrow{u}, v$:

(γ^*)-rule

$$A \equiv A_0 \text{ where } \{ \overrightarrow{a} := \overrightarrow{A}, \tag{31a}$$

$$p := \lambda(\overrightarrow{u})\lambda(v)P, \tag{31b}$$

$$\overrightarrow{b} := \overrightarrow{B} \} \tag{31c}$$

$$\Rightarrow_{(\gamma^*)} A_0' \text{ where } \{ \overrightarrow{a} := \overrightarrow{A}', \tag{31d}$$

$$p' := \lambda(\overrightarrow{u})P, \tag{31e}$$

$$\overrightarrow{b} := \overrightarrow{B'} \} \tag{31f}$$

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 (31a)–(31f), modulo renaming the variables $\overrightarrow{u}, v$

Reduction Rules

[Scope Rule] Closure of part of the unspecified, open scope

$$\begin{aligned}
&\mathsf{ToScope}\,\underbrace{\big(B_0 \text{ where } \{\, \overrightarrow{c} := \overrightarrow{C}, \underbrace{\overrightarrow{q} := \overrightarrow{B}}\,\}\big)}_{\text{unspecified, open scope}}(s_c) \\
&\Rightarrow \mathcal{C}\,\underbrace{\big(B_0(s_r) \text{ where } \{\, \overrightarrow{c} := \overrightarrow{C}\,\}\big)}_{\text{closed scope}} \text{ where } \{\; \underbrace{\overrightarrow{q} := \overrightarrow{B}}_{\text{outside } \mathcal{C} \text{ scope}} \;\}
\end{aligned} \tag{ScopeR}$$

Derived Reduction Rules

Lemma (Bekič-Scott Scope)

$$\begin{aligned}
& A_0 \text{ where } \{ p := \mathsf{ToScope}\big(B_0 \text{ where } \{ \overrightarrow{c} := \overrightarrow{C}, \overrightarrow{q} := \overrightarrow{B} \}\big)(s_c), \\
& \qquad \overrightarrow{p} := \overrightarrow{A} \} \\
\Rightarrow\ & A_0 \text{ where } \{ p := \mathcal{C}\big(B_0(s_r) \text{ where } \{ \overrightarrow{c} := \overrightarrow{C} \}\big), \\
& \qquad \overrightarrow{q} := \overrightarrow{B},\ \overrightarrow{p} := \overrightarrow{A} \}
\end{aligned} \tag{B-S-sc}$$

Proof.

By the reduction rules:

- (ScopeR)
- compositionality of recursion (rec-comp)
- Bekič-Scott Rule (B-S)

□

The (head) rule is similarly generalized to the derived rule for scopes.

Theorem (Canonical Form Theorem)

For every $A \in$ Terms, *there is a unique up to congruence, irreducible term* cf$(A) \in$ Terms, *such that:*

1. $A \Rightarrow \text{cf}(A)$
2. *for every* B, *if* $A \Rightarrow B$ *and* B *is irreducible, then* $B \equiv_c \text{cf}(A)$, *i.e.,* cf(A) *is the unique, up to congruence, term to which A can be reduced*
3. *(a)* $\text{FreeRecV}(\text{cf}(A)) = \text{FreeRecV}(A)$
   (b) $\text{FreePureV}(\text{cf}(A)) = \text{FreePureV}(A)$
4. $\text{Consts}(\text{cf}(A)) = \text{Consts}(A)$

Proof.

The proof is by induction on term structure of A, (5a)–(5e), (5h), 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. $\square$

Algorithmic Semantic of L^{λ}_{ar} / L^{λ}_{r}

How is the algorithmic semantics provided?

- For each proper (i.e., non-immediate) $A \in \mathsf{Terms}$, $\mathsf{cf}(A)$ determines the algorithm $\mathsf{alg}(A)$ for computing $\mathsf{den}(A)$
- By the Canonical Form Theorem 6:

Theorem (Effective Reduction Calculi)

For every term $A \in \mathsf{Terms}$, *its canonical form* $\mathsf{cf}(A)$ *is effectively computed, by the extended reduction calculus:*

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

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

- L^{λ}_{ar} / L^{λ}_{r}/ DTTSI provide Computational Semantics with:
  - denotations
  - algorithms for computing denotations
- 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
- Representation of mathematical statements proven or verified, or neither
  - L^{λ}_{ar} / L^{λ}_{r} with logical operators and pure quantifiers
  - L^{λ}_{ar} / L^{λ}_{r} can be used for proof-theoretic computational reasoning and inferences of semantic information
- L^{λ}_{ar} / L^{λ}_{r} in Dependent-Type Theory of Situated Info (DTTSI)

  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).
  DOI 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).
DOI 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`