Type Theory of Situated Algorithms and Information

Roussanka Loukanova

Stockholm University, Sweden

Logic and Algorithms in Computational Linguistics 2017
(LACompLing2017)
Stockholm, August 16–19, 2017

Outline

Outline

- Moschovakis (1994): untyped theory of Moschovakis algorithms / recursion
- L^{λ}_{ar}, Moschovakis (2006), and L^{λ}_{r} (both ongoing, open work) are classes of:
  higher-order Simply-Typed Theories of Recursion
- L^{λ}_{ar} and L^{λ}_{r} have two-fold semantics:
  - denotations in typed domains
    - state-dependent objects (à la Gallin intensionality), i.e., functions from states to objects:

$$\mathbb{T}_{(s\to\tau)} = \{f \mid f : \mathbb{T}_s \longrightarrow \mathbb{T}_\tau\}$$

    - pure objects: entities and function values do not depend on states
  - algorithmic semantics: algorithms for computing denotations

Algorithmic Meaning

$$\underbrace{\text{Syntax of } L^{\lambda}_{ar}\ (L^{\lambda}_{r}) \Longrightarrow \text{Algorithmic Meanings (Computations)} \Longrightarrow \text{Denotations}}_{\text{Semantics of } L^{\lambda}_{ar}(L^{\lambda}_{r})}$$

What is Moschovakis Type-Theory of Recursion?

A class of higher-order Typed Theory of Recursion with

- formal syntax
- reduction calculi
- denotational semantics; algorithmic semantics

Algorithmic Meaning

$$\underbrace{\text{Formal Syntax} \Longrightarrow \text{Algorithmic Meanings (Computations)}}_{\text{Semantics}} \Longrightarrow \text{Denotations}$$

- L^{λ}_{ar} is Simply-Typed Theory of Acyclic Recursion: computations close-off after finite number of steps
- L^{ta}_{ar} is Polymorphic Typed Theory of Acyclic Recursion: computations close-off after finite number of steps
- L^{λ}_{r} is Simply-Typed Theory of Full Recursion
- L^{ta}_{r} is Polymorphic Typed Theory of Full Recursion
- Adding terms for constraints

Placement of L^{ta}_{ar}, L^{ta}_{r} in a class of type theories:

$$IL \subsetneq TY_2 \subsetneq L^{\lambda}_{ar} \subsetneq L^{\lambda}_{ra} \subsetneq L^{\lambda}_{r} \subsetneq L^{ta}_{r} \quad (1)$$

$$L^{\lambda}_{ar} \subsetneq L^{\lambda}_{ra} \subsetneq L^{ta}_{ar} \subsetneq L^{ta}_{r} \subsetneq \mathit{TTSitInfo} \subsetneq L^{ST}_{GP} \quad (2)$$

- Montague IL for PTQ (1970-73)
- Gallin TY_2 (1975)
- L^{λ}_{ar}, L^{λ}_{r} (2006–now) and L^{ta}_{ar}, L^{ta}_{r} are

  Type-Theories of Algorithms with
  1. Recursion Operator (forming Recursion Terms)
  2. Restrictor Operator (forming Restriction Terms):
     can be added to all versions: L^{λ}_{ar}, L^{λ}_{r} and L^{ta}_{ar}, L^{ta}_{r}, L^{ST}_{GP}
  3. Reduction Calculi

Reduction Calculi: reduction rules

- for reducing every term A to a canonical form $cf(A)$
- $cf(A)$ of a meaningful term A determines the algorithm for computing $den(A)$, in a step-by-step mode, from the simplest components of A

$$A \equiv (200 + 40)/6 \tag{3a}$$
$$\Rightarrow n/d \text{ where } \{\, n := (a_1 + a_2), \tag{3b}$$
$$a_1 := 200,\ a_2 := 40,\ d := 6 \,\} \tag{3c}$$

$$B \equiv (120 + 120)/6 \tag{4a}$$
$$\Rightarrow n/d \text{ where } \{\, n := (a_1 + a_2), \tag{4b}$$
$$a_1 := 120,\ a_2 := 120,\ d := 6 \,\} \tag{4c}$$

$$C \equiv n/d \text{ where } \{\, n := (a + a),\ a := 120,\ d := 6 \,\} \tag{5}$$

$$\text{den}(A) = \text{den}(B) = \text{den}(C) = 40 \tag{6a}$$
$$A \not\approx B \not\approx C \tag{6b}$$

$$A \equiv (200 + 40)/6 \tag{7a}$$
$$\Rightarrow n/d \text{ where } \{\, n := (a_1 + a_2), \tag{7b}$$
$$a_1 := 200,\ a_2 := 40,\ d := 6 \,\} \tag{7c}$$

Recursion terms with restrictor:

$$D \equiv \big(n/d \text{ such that } \{\, d \neq 0 \,\}\big) \text{ where } \{ \tag{8a}$$
$$n := (a_1 + a_2), \tag{8b}$$
$$a_1 := 200,\ a_2 := 40,\ d := 6 \,\} \tag{8c}$$

$$E \equiv \big(n/d \text{ such that } \{\, d \neq 0 \,\}\big) \text{ where } \{ \tag{9a}$$
$$n := (a_1 + a_2), \tag{9b}$$
$$a_1 := 200,\ a_2 := 40,\ d := 0 \,\} \tag{9c}$$

$$\text{den}(A) = \text{den}(D) = 40; \qquad \text{den}(E) = er \tag{10a}$$
$$A \not\approx D \not\approx E \tag{10b}$$

Syntax of L^{λ}_{ar} - acyclic recursion (L^{λ}_{r} full recursion without acyclicity)

Gallin Types: $\sigma :\equiv e \mid t \mid s \mid (\tau_1 \to \tau_2)$

Constants: $\mathrm{Const}_\tau = \{c^\tau_0, c^\tau_1, \dots\}$

Variables: $\mathrm{PureVars}_\tau = \{v^\tau_0, v^\tau_1, \dots\}$, $\mathrm{RecVars}_\tau = \{p^\tau_0, p^\tau_1, \dots\}$

Terms of L^{λ}_{ar} (L^{λ}_{r})

$$A :\equiv c^\tau : \tau \mid x^\tau : \tau \tag{11a}$$

$$\mid B^{(\sigma\to\tau)}(C^\sigma) : \tau \tag{11b}$$

$$\mid \lambda(v^\sigma)(B^\tau) : (\sigma \to \tau) \tag{11c}$$

$$\mid \left[A^\sigma_0 \text{ where } \{p^{\sigma_1}_1 := A^{\sigma_1}_1, \dots, p^{\sigma_n}_n := A^{\sigma_n}_n\}\right] : \sigma \tag{11d}$$

given that $p_i \in \mathrm{RecVars}^{\sigma_i}$, $A_i \in \mathrm{Terms}^{\sigma_i}$ satisfy Acyclicity Constraint:

- $\{p^{\sigma_1}_1 := A^{\sigma_1}_1, \dots, p^{\sigma_n}_n := A^{\sigma_n}_n\}$ is acyclic, i.e., exists a function

$$\mathrm{rank}\colon \{p_1, \dots, p_n\} \to \mathbb{N}$$

s.th. for all $i, j \in \{1, \dots, n\}$:
if p_j occurs freely in A_i, then $\mathrm{rank}(p_i) > \mathrm{rank}(p_j)$

Denotational Semantics of L^{λ}_{ar}

For any *semantic structure* $\mathfrak{A}(\mathrm{Const}) = \langle\, \mathbb{T}, \mathcal{I}(\mathrm{Const}) \,\rangle$, where

$\mathbb{T} = \{\mathbb{T}_\sigma \mid \sigma \in \mathit{Types}\}$ is a frame of typed objects,
$\mathcal{I}\colon\ \mathrm{Const}_\sigma \longrightarrow \mathbb{T}_\sigma$ is the *interpretation function*,
with $G = \{g \mid g\colon\ \mathrm{PureVars} \cup \mathrm{RecVars} \longrightarrow \mathbb{T}\}$ — all variable valuations,
the *denotation function*, $\mathrm{den}\colon \mathit{Terms} \longrightarrow \{f \mid f\colon G \longrightarrow \mathbb{T}\}$
is defined by structural recursion:

(D1) $\mathrm{den}(x)(g) = g(x);\ \ \mathrm{den}(c)(g) = \mathcal{I}(c)$

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

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

The Denotation of the Recursion Terms (continuation)

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

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

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

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

Intuitively:

- $\mathrm{den}(A_1)(g), \ldots, \mathrm{den}(A_n)(g)$ are computed recursively and stored in $p_1, \ldots, p_n$, respectively
- the denotation $\mathrm{den}(A_0)(g)$ may depend on the values stored in $p_1, \ldots, p_n$

Example

$$\text{Kim} \xrightarrow{\text{render}} kim : \widetilde{\text{e}} \tag{12a}$$

$$\text{Maja} \xrightarrow{\text{render}} maja : \widetilde{\text{e}} \tag{12b}$$

$$\text{runs} \xrightarrow{\text{render}} runs : (\widetilde{\text{e}} \to \widetilde{\text{t}}) \tag{12c}$$

$$\text{hugs} \xrightarrow{\text{render}} hugs : (\widetilde{\text{e}} \to (\widetilde{\text{e}} \to \widetilde{\text{t}})) \tag{12d}$$

$$\text{Kim hugs Maja.} \tag{13a}$$

$$\xrightarrow{\text{render}} \left[hugs^{(\widetilde{\text{e}} \to (\widetilde{\text{e}} \to \widetilde{\text{t}}))}(maja^{\widetilde{\text{e}}}) \right]^{(\widetilde{\text{e}} \to \widetilde{\text{t}})}(kim^{\widetilde{\text{e}}}) : \widetilde{\text{t}} \tag{13b}$$

$$\equiv \left[hugs(maja) \right](kim) : \widetilde{\text{t}} \tag{13c}$$

$$\Rightarrow_{\text{cf}} \left[hugs(m)(k) \text{ where } \{ k := kim,\ m := maja \} \right] : \widetilde{\text{t}} \tag{13d}$$

Example

$$\text{John likes Mary's father.} \tag{14a}$$

$$\xrightarrow{\text{render}} \left[like\left(father\_of\left(mary\right)\right)\right]\left(john\right) \ : \ \widetilde{\mathrm{t}} \tag{14b}$$

$$\equiv like\left(father\_of\left(mary\right), john\right) \ : \ \widetilde{\mathrm{t}} \tag{14c}$$

$$\Rightarrow_{\mathrm{cf}} like(f)(j) \text{ where } \{j := john,\ m := mary,\ f := father\_of(m)\} \tag{14d}$$

$$\equiv like(f,j) \text{ where } \{j := john,\ m := mary,\ f := father\_of(m)\} \tag{14e}$$

Example

A term with coordination:

$$\text{Mary runs and smiles.} \xrightarrow{\text{render}} \tag{15a}$$

$$A \Rightarrow_{\text{cf}} \lambda x\Big[r(x) \;\&\; s(x)\Big](m) \text{ where } \{\, r := run,\, s := smile, \tag{15b}$$
$$m := mary \,\}$$

$$\not\approx \Big[r(m) \;\&\; s(m)\Big] \text{ where } \{\, r := run,\, s := smile, \tag{15c}$$
$$m := mary \,\}$$

This is justified because

- (15a) and (15b) denote predication, while (15c) is a conjunction
- β-reduction does not apply (in full) to algorithmic synonymy
- β-reduction and the established results of λ-calculus are valid denotationally

Type Theory of Restricted Algorithms L_{ra}^{λ}

$$A :\equiv \mathsf{c}^{\tau} : \tau \mid x^{\tau} : \tau \mid B^{(\sigma\to\tau)}(C^{\sigma}) : \tau \mid \lambda(v^{\sigma})(B^{\tau}) : (\sigma \to \tau) \quad (16a)$$

$$\mid \left(A_0^{\sigma_0} \text{ where } \{p_1^{\sigma_1} := A_1^{\sigma_1}, \ldots, p_n^{\sigma_n} := A_n^{\sigma_n}\}\right) : \sigma_0 \quad (16b)$$

$$\mid \left(A_0^{\sigma_0} \text{ such that } \{ C_1^{\tau_1}, \ldots, C_n^{\tau_n} \}\right) : \sigma_0' \quad (16c)$$

In (16b):

$p_i \in \text{RecVars}^{\sigma_i}$, $A_i \in \text{Terms}^{\sigma_i}$ satisfy Acyclicity Constraint:

- $\{p_1^{\sigma_1} := A_1^{\sigma_1}, \ldots, p_n^{\sigma_n} := A_n^{\sigma_n}\}$ is acyclic, i.e., exists a function $\text{rank}\colon \{p_1, \ldots, p_n\} \to \mathbb{N}$ s.th. if p_j occurs freely in A_i, then $\text{rank}(p_i) > \text{rank}(p_j)$

In (16c):

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

$$\sigma_0' \equiv \begin{cases} \sigma_0, & \text{if } \tau_i \equiv \mathsf{t}, \text{ for all } i \in \{1, \ldots, n\} \\ \widetilde{\sigma_0} \equiv (\mathsf{s} \to \sigma_0), & \text{if } \tau_i \equiv \widetilde{\mathsf{t}}, \text{ for some } i \in \{1, \ldots, n\} \end{cases} \quad (17)$$

Abbreviations

- $er \equiv error$
- $\widetilde{\tau} \equiv (s \to \tau)$, where $\tau \in$ Types (the type of state dependent objects of type σ)
- Abbreviated sequences and mutually recursive assignments:

$$\overrightarrow{X} \equiv X_1, \ldots, X_n, \text{ where } X_i \in \text{Terms for all } i \in \{1, \ldots, n\} \text{ or } X_i \in \text{Types for all } i \in \{1, \ldots, n\} \tag{18}$$

$$\left(A_0^{\sigma_0} \text{ such that } \{ C_1^{\tau_1} \ldots, C_n^{\tau_n} \}\right) \equiv \left(A_0^{\sigma_0} \text{ such that } \{ \overrightarrow{C} \}\right) \tag{19a}$$

$$\equiv \left(A_0^{\sigma_0} \text{ such that } \overrightarrow{C}\right) \tag{19b}$$

$$\overrightarrow{p} := \overrightarrow{A} \equiv \left[p_1 := A_1, \ \ldots, \ p_n := A_n\right] \quad (n \geq 0) \tag{20}$$

$$\begin{aligned} &(A_0 \text{ s.t. } \{ \overrightarrow{C} \}) \text{ where } \{ \overrightarrow{p} := \overrightarrow{A} \} \\ &\equiv (A_0 \text{ s.t. } \overrightarrow{C}) \text{ where } \{ \overrightarrow{p} := \overrightarrow{A} \} \end{aligned} \tag{21}$$

(D5)

Case 1: for all $i \in \{1, \ldots, n\}$, $C_i : \mathrm{t}$
For every $g \in G$:

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

Case 2: for some $i \in \{1, \ldots, n\}$, $C_i : \tilde{\mathrm{t}}$ (state dependent proposition)

For every $g \in G$, and every state $s \in \mathbb{T}_s$:

$$\mathrm{den}\left(A_0^{\sigma_0} \text{ s.t. } \{ \overrightarrow{C} \}\right)(g)(s) = \begin{cases} \mathrm{den}(A_0)(g), & \text{if } \mathrm{den}(C_i)(g) = 1, \\ & \quad \text{for all } i \text{ s.th. } C_i : \mathrm{t}, \text{ and} \\ & \mathrm{den}(C_i)(g)(s) = 1, \\ & \quad \text{for all } i \text{ s.th. } C_i : \tilde{\mathrm{t}} \\ er, & \text{if } \mathrm{den}(C_i)(g) = er \text{ or} \\ & \mathrm{den}(C_i)(g) = 0, \\ & \quad \text{for some } i \text{ s.th. } C_i : \mathrm{t} \\ er, & \text{if } \mathrm{den}(C_i)(g)(s) = er \text{ or} \\ & \mathrm{den}(C_i)(g)(s) = 0, \\ & \quad \text{for some } i \text{ s.th. } C_i : \tilde{\mathrm{t}} \end{cases}$$

Immediate terms have no algorithmic meaning.

Definition (The set ImT of immediate terms)

$$\mathsf{ImT}^{\tau} :\equiv X^{\tau} \mid Y^{(\tau_1 \to \cdots \to (\tau_m \to \tau))}(v_1^{\tau_1}) \ldots (v_m^{\tau_m}) \quad \text{(immediate applicative terms)} \tag{22a}$$

$$\mathsf{ImT}^{(\sigma_1 \to \cdots \to (\sigma_n \to \tau))} :\equiv \lambda(u_1^{\sigma_1}) \ldots \lambda(u_n^{\sigma_n}) Y^{(\tau_1 \to \cdots \to (\tau_m \to \tau))}(v_1^{\tau_1}) \ldots (v_m^{\tau_m}) \quad \text{(immediate } \lambda\text{-terms)} \tag{22b}$$

where $n \geq 0$, $m \geq 0$;
$u_i, v_j \in \mathsf{PureVars}$,
$X \in \mathsf{Vars}$, $Y \in \mathsf{RecVars}$

Definition (Proper terms)

$$\mathsf{PrT} = (\mathsf{Terms} - \mathsf{ImT}) \tag{23}$$

Reduction rules of L^{λ}_{ar} / L^{λ}_{ra} (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)

[Compositionality:]

- If $A \Rightarrow A'$ and $B \Rightarrow B'$, then $A(B) \Rightarrow A'(B')$ (app-comp / rep1)
- If $A \Rightarrow B$, then $\lambda(u)(A) \Rightarrow \lambda(u)(B)$ (λ-comp / rep2)
- If $A_i \Rightarrow B_i$, for $i = 0, \ldots, n$, then

$$A_0 \text{ where } \{ p_1 := A_1, \ldots, p_n := A_n \} \quad \text{(wh-comp / rep3)}$$
$$\Rightarrow B_0 \text{ where } \{ p_1 := B_1, \ldots, p_n := B_n \}$$

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

$$\left(A_0 \text{ such that } \{ C_1, \ldots, C_n \}\right) \quad \text{(st-comp / rep4)}$$
$$\Rightarrow \left(B_0 \text{ such that } \{ R_1, \ldots, R_n \}\right)$$

Reduction rules of L^{λ}_{ar} / L^{λ}_{ra} (to be continued)

[Head Rule:] given that 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 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

$$\Big(A_0 \text{ where } \{\overrightarrow{p} := \overrightarrow{A}\}\Big)(B) \qquad (27)$$
$$\Rightarrow A_0(B) \text{ where } \{\overrightarrow{p} := \overrightarrow{A}\} \qquad \text{(recap)}$$

Reduction rules of L_{ar}^{λ} / L_{ra}^{λ} (to be continued)

[Application Rule:] given that $B \in \mathrm{PrT}$ is a proper term, and $p \in \Big[\mathrm{RecVars} - \big(\mathrm{FV}\big(A(B)\big) \cup \mathrm{BV}\big(A(B)\big)\big)\Big]$ is fresh

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

[λ-rule:] given fresh $p'_i \in \Big[\mathrm{RecVars} - \big(\mathrm{FV}(A) \cup \mathrm{BV}(A)\big)\Big]$, $i = 1, \ldots, n$, for $A \equiv A_0$ where $\{\, p_1 := A_1, \ldots, p_n := A_n \,\}$

$$\lambda(u)\Big(A_0 \text{ where } \big\{\, p_1 := A_1, \ldots, p_n := A_n \,\big\}\Big) \tag{λ}$$

$$\Rightarrow \Big[\lambda(u)A'_0 \text{ where } \big\{\, p'_1 := \lambda(u)A'_1, \ldots, p'_n := \lambda(u)A'_n \,\big\}\Big]$$

where, for all $i = 0, \ldots, n$,

$$A'_i \equiv \Big[A_i\big\{p_1 :\equiv p'_1(u), \ldots, p_n :\equiv p'_n(u)\big\}\Big] \tag{29}$$

[st / Restriction Rule:]

1. given that:
   - A_0, C_i $(i = 1, \ldots, n,\ n \geq 0)$ are proper terms, and
   - $\overrightarrow{R}$ (if not empty) are immediate
   - $a_0, c_i \in \mathsf{RecVars}$ $(i = 1, \ldots, n)$ are fresh

$$
\begin{aligned}
&(A_0 \text{ such that } \{\, C_1, \ldots, C_n, \overrightarrow{R} \,\}) && \text{(st1)}\\
\Rightarrow\ &(a_0 \text{ such that } \{\, c_1, \ldots, c_n, \overrightarrow{R} \,\})\\
&\qquad \text{where } \{\, a_0 := A_0,\ c_1 := C_1,\ \ldots, c_n := C_n \,\}
\end{aligned}
$$

2. given that:
   - C_i $(i = 1, \ldots, n,\ n \geq 0)$ are proper terms
   - A_0, $\overrightarrow{R}$ (if not empty) are immediate, and
   - $c_i \in \mathsf{RecVars}$ $(i = 1, \ldots, n)$ are fresh

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

Proposition (Basic Restricted Memory Variables)

Assume that, for $n \geq 1$:

- $\overrightarrow{R_j}$ *are immediate terms, and*
- $p_i \in \mathit{RecVars}$, $i = 2, \ldots, n$, *are fresh with respect to* p_1, $\overrightarrow{R_j}$ $(j = 1, \ldots, n)$

Then:

$$((\ldots((p_1 \text{ s.t. } \overrightarrow{R_1}) \text{ s.t. } \overrightarrow{R_2}) \ldots) \text{ s.t. } \overrightarrow{R_n}) \tag{32a}$$

$$\Rightarrow (p_n \text{ s.t. } \overrightarrow{R_n}) \text{ where } \{\, p_n := (p_{n-1} \text{ s.t. } \overrightarrow{R_{n-1}}), \tag{32b}$$

$$\ldots, \tag{}$$

$$p_3 := (p_2 \text{ s.t. } \overrightarrow{R_2}), \tag{32c}$$

$$p_2 := (p_1 \text{ s.t. } \overrightarrow{R_1}) \,\} \tag{32d}$$

Proof: by induction on n.

Case1: $n = 1$
$(p_1 \text{ s.t. } \overrightarrow{R_1}) \Rightarrow (p_1 \text{ s.t. } \overrightarrow{R_1})$ is trivially true

Case2: Assume (32a)–(32d), for $n \geq 1$. Then, we reduce the term (33a) to the canonical form (33h)–(33j), by applying the reduction rules (compositionally). □

$$(\underbrace{((\ldots((p_1 \text{ s.t. } \overrightarrow{R_1}) \text{ s.t. } \overrightarrow{R_2})\ldots) \text{ s.t. } \overrightarrow{R_n})}_{p_{n+1}} \text{ s.t. } \overrightarrow{R_{n+1}}) \tag{33a}$$

$$\text{(st1)} \quad \Rightarrow (p_{n+1} \text{ s.t. } \overrightarrow{R_{n+1}}) \text{ where } \{ \tag{33b}$$

$$p_{n+1} := \underbrace{((\ldots((p_1 \text{ s.t. } \overrightarrow{R_1}) \text{ s.t. } \overrightarrow{R_2})\ldots) \text{ s.t. } \overrightarrow{R_n})}\ \} \tag{33c}$$

$$\text{(ind.hyp.; wh-comp)} \tag{33d}$$

$$\Rightarrow (p_{n+1} \text{ s.t. } \overrightarrow{R_{n+1}}) \text{ where } \{ \tag{33e}$$

$$p_{n+1} := \Big[(p_n \text{ s.t. } \overrightarrow{R_n}) \text{ where } \{\, p_n := (p_{n-1} \text{ s.t. } \overrightarrow{R_{n-1}}), \tag{33f}$$

$$\ldots,$$

$$p_3 := (p_2 \text{ s.t. } \overrightarrow{R_2}),\ p_2 := (p_1 \text{ s.t. } \overrightarrow{R_1})\ \}\Big] \tag{33g}$$

$$\text{(B-S)} \quad \Rightarrow (p_{n+1} \text{ s.t. } \overrightarrow{R_{n+1}}) \text{ where } \{ \tag{33h}$$

$$p_{n+1} := (p_n \text{ s.t. } \overrightarrow{R_n}),\ p_n := (p_{n-1} \text{ s.t. } \overrightarrow{R_{n-1}}), \tag{33i}$$

$$\ldots,$$

$$p_3 := (p_2 \text{ s.t. } \overrightarrow{R_2}),\ p_2 := (p_1 \text{ s.t. } \overrightarrow{R_1})\ \} \tag{33j}$$

Proposition (Restricted Memory Variables)

Assume that, for $n \geq 1$:

- $\overrightarrow{R_j}$ *are proper terms, and* $\overrightarrow{I_j}$ *are immediate*
- $p_i \in \mathit{RecVars}$ $(i = 2, \ldots, n)$ *and* $r_j \in \mathit{RecVars}$ $(j = 1, \ldots, n)$ *are fresh with respect to* p_1, $\overrightarrow{R_j}$, $\overrightarrow{I_j}$ $(j = 1, \ldots, n)$

Then:

$$((\ldots((p_1 \text{ s.t. } \{\overrightarrow{R_1}, \overrightarrow{I_1}\}) \text{ s.t. } \{\overrightarrow{R_2}, \overrightarrow{I_2}\})\ldots) \text{ s.t. } \{\overrightarrow{R_n}, \overrightarrow{I_n}\}) \tag{34a}$$

$$\Rightarrow (p_n \text{ s.t. } \{\overrightarrow{r_n}, \overrightarrow{I_1}\}) \text{ where } \{\, p_n := (p_{n-1} \text{ s.t. } \{\overrightarrow{r_{n-1}}, \overrightarrow{I_{n-1}}\}), \tag{34b}$$

$$\ldots,$$

$$p_3 := (p_2 \text{ s.t. } \{\overrightarrow{r_2}, \overrightarrow{I_2}\}), \tag{34c}$$

$$p_2 := (p_1 \text{ s.t. } \{\overrightarrow{r_1}, \overrightarrow{I_1}\}), \tag{34d}$$

$$\overrightarrow{r} := \overrightarrow{R}\, \} \tag{34e}$$

Proof: by induction on $n \geq 1$.

□

Theorem (Canonical Form Theorem)

For each $A \in \mathsf{Terms}$, *there is a unique up to congruence, irreducible* $\mathsf{cf}(A) \in \mathsf{Terms}$ *s.th.:*

1. $\mathsf{cf}(A) \equiv A_0$ where $\{p_1 := A_1, \ldots, p_n := A_n\}$
   for some explicit, irreducible $A_0, \ldots, A_n \in \mathsf{Terms}$ $(n \geq 0)$
2. $A \Rightarrow \mathsf{cf}(A)$

- Every term A reduces to $\mathrm{cf}(A)$
- How is the algorithmic semantics of an algorithmically meaningful term A determined?

Algorithmic Semantics: Recursors, i.e., Recursive Computations

For every proper term A, such that

$$A \Rightarrow \mathrm{cf}(A) \equiv A_0 \text{ where } \{p_1 := A_1, \ldots, p_n := A_n\} \tag{35}$$

$\mathrm{alg}(A)$ is the recursor, i.e., the algorithm, for computing $\mathrm{den}(A)$:

$$\mathrm{alg}(A) = \Big(\mathrm{den}(A_0), \mathrm{den}(A_1), \ldots, \mathrm{den}(A_n)\Big) \tag{36}$$

which computes $\mathrm{den}(A_i)(g)$, for every $g \in G$,

$$\overline{\mathrm{alg}(A)}(g) = \mathrm{den}(A)(g) = \mathrm{den}(A_0)(g) \tag{37}$$

according to $\mathrm{rank}(p_i)$, $i = 1, \ldots, n$

Algorithmic Equivalence

Intuitively: L^{λ}_{r} is a formalization of the mathematical notion of algorithm, for computing values of recursive functions, by recursion terms and terms in canonical forms.
I.e., the concept of algorithm is defined formally, at the object level of its syntax.

Theorem (of Algorithmic Synonymy)

Two terms $A, B \in$ Terms *are algorithmically equivalent*, $A \approx B$, *iff there are explicit, irreducible terms* $A_0, A_1, \ldots, A_n$, $B_0, B_1, \ldots, B_n$ $(n \geq 0)$ *s.th.:*

- $A \Rightarrow_{cf} A_0$ where $\{p_1 := A_1, \ldots, p_n := A_n\}$
- $B \Rightarrow_{cf} B_0$ where $\{p_1 := B_1, \ldots, p_n := B_n\}$
- $\models A_i = B_i \quad (i = 0, \ldots, n)$, *i.e.,*

$$\mathrm{den}(A_i)(g) = \mathrm{den}(B_i)(g), \quad \textit{for all } g \in G \tag{38}$$

Example

Given that *dog* and *runs* are constants, the following terms are algorithmically, synonymous, but can not be reduced to each other:

$$\lambda x\, dog(x) \approx dog : (\widetilde{\mathrm{e}} \to \widetilde{\mathrm{t}}) \tag{39}$$

$$\lambda x\, runs(x) \approx runs : (\widetilde{\mathrm{e}} \to \widetilde{\mathrm{t}}) \tag{40}$$

$$\lambda\, x\, some(x) \approx some : [(\widetilde{\mathrm{e}} \to \widetilde{\mathrm{t}}) \to ((\widetilde{\mathrm{e}} \to \widetilde{\mathrm{t}}) \to \widetilde{\mathrm{t}})] \tag{41}$$

$$\text{some dog} \xrightarrow{\text{render}} some^{(\widetilde{\mathrm{e}}\to\widetilde{\mathrm{t}})\to((\widetilde{\mathrm{e}}\to\widetilde{\mathrm{t}})\to\widetilde{\mathrm{t}})}(dog^{(\widetilde{\mathrm{e}}\to\widetilde{\mathrm{t}})}) : \big((\widetilde{\mathrm{e}} \to \widetilde{\mathrm{t}}) \to \widetilde{\mathrm{t}}\big) \tag{42a}$$

$$\equiv some(dog) : \big((\widetilde{\mathrm{e}} \to \widetilde{\mathrm{t}}) \to \widetilde{\mathrm{t}}\big) \tag{42b}$$

$$\Rightarrow_{\mathrm{cf}} \big[some(d) \text{ where } \{\, d := dog \,\}\big] : \big((\widetilde{\mathrm{e}} \to \widetilde{\mathrm{t}}) \to \widetilde{\mathrm{t}}\big) \tag{42c}$$

recursion term

$$\text{Some dog runs} \xrightarrow{\text{render}} \tag{43a}$$

$$\big(\mathit{some}(\mathit{dog})\big)(\mathit{runs}) : \widetilde{\mathrm{t}} \tag{43b}$$

$$\equiv \mathit{some}(\mathit{dog}, \mathit{runs}) \qquad (\text{rel. not.}) \tag{43c}$$

$$\Rightarrow_{\mathrm{cf}} \big[\big(\mathit{some}(p_1)\big)(p_2) \ \text{ where } \{p_1 := \mathit{dog},\ p_2 := \mathit{runs}\}\big] : \widetilde{\mathrm{t}} \tag{43d}$$

$$\approx \big[\big(\mathit{some}(p_1)\big)(p_2) \ \text{ where } \{p_1 := \lambda x\, \mathit{dog}(x),\ p_2 := \lambda x\, \mathit{runs}(x)\}\big] : \widetilde{\mathrm{t}} \tag{43e}$$

$$\text{Some dog runs} \xrightarrow{\text{render}} \big[\big(Q(p_1)\big)(p_2) \ \text{ where } \{Q := \mathit{some},\ p_1 := \mathit{dog},\ p_2 := \mathit{runs}\}\big] \tag{44a}$$

$$\not\approx \big[\big(\mathit{some}(p_1)\big)(p_2) \ \text{ where } \{p_1 := \mathit{dog},\ p_2 := \mathit{runs}\}\big] \tag{44b}$$

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

We may consider rendering the definite article "the" to a constant:

$$\text{the} \xrightarrow{\text{render}} \mathit{the} \in \text{Const}_{((\widetilde{\text{e}}\to\widetilde{\text{t}})\to\widetilde{\text{e}})} \tag{45}$$

and the following denotation of the constant *the*:

$$\big[\big(\text{den}(\mathit{the})\big)(g)\big](\bar{p})(s_0) = \begin{cases} y, & \text{in case (i.e., iff) } y \text{ is} \\ & \text{the unique } y \in \mathbb{T}_{\text{e}}, \\ & \text{for which } \bar{p}(s \mapsto y)(s_0) = 1 \\ \text{er}, & \text{otherwise} \\ & \text{i.e., there is no unique entity} \\ & \text{which has the property } \bar{p} \text{ in } s_0 \end{cases} \tag{46}$$

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

There are other possibilities for rendering the definite article "the", e.g., with complex terms of generalized quantifiers or by using the restrictor.

Example: possible rendering of the definite article "the" — Option 2

We may consider rendering the definite article "the" to a term:

$$\text{the} \xrightarrow{\text{render}} A \equiv \lambda(x)\big[\big(q \text{ such that } \{\, \mathit{unique}(p)(q) \,\}\big) \tag{47a}$$

$$\text{where } \{\, p := x \,\}\big] \tag{47b}$$

$$: ((\widetilde{\mathrm{e}} \to \widetilde{\mathrm{t}}) \to \widetilde{\mathrm{e}}) \tag{47c}$$

for $q \in \mathrm{RecVars}_{\mathrm{e}}$, $p \in \mathrm{RecVars}_{(\widetilde{\mathrm{e}} \to \widetilde{\mathrm{t}})}$,
$\mathit{unique} \in \mathrm{Const}_{(((\widetilde{\mathrm{e}} \to \widetilde{\mathrm{t}}) \to \mathrm{e}) \to \widetilde{\mathrm{t}})}$,
and the following denotation (by ignoring the variable assignment g) of the constant $\mathit{unique} : (((\widetilde{\mathrm{e}} \to \widetilde{\mathrm{t}}) \to \mathrm{e}) \to \widetilde{\mathrm{t}})$:

$$\big[\,\mathrm{den}(\mathit{unique})\big](\bar{p})(y)(s_0) = \begin{cases} 1, & \text{in case (i.e., iff) } y \text{ is} \\ & \text{the unique } y \in \mathbb{T}_{\mathrm{e}}, \\ & \text{for which } \bar{p}(s \mapsto y)(s_0) = 1 \\ \mathrm{er}, & \text{otherwise} \end{cases} \tag{48}$$

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

Example: possible rendering of the definite article "the" — Option 3

We may consider rendering the definite article "the" to a term that is underspecified for the property p:

$$\text{the} \xrightarrow{\text{render}} A \equiv \big(q \text{ such that } \{\, \mathit{unique}(p)(q) \,\}\big) \tag{49a}$$
$$: \widetilde{\mathsf{e}} \tag{49b}$$

for $q \in \mathsf{RecVars}_{\mathsf{e}}$, $p \in \mathsf{RecVars}_{(\widetilde{\mathsf{e}} \to \widetilde{\mathsf{t}})}$,
$\mathit{unique} \in \mathsf{Const}_{(((\widetilde{\mathsf{e}} \to \widetilde{\mathsf{t}}) \to \mathsf{e}) \to \widetilde{\mathsf{t}})}$

- Then, p gets specified, by NPs:

$$\text{the dog} \xrightarrow{\text{render}} A \equiv \big(q \text{ such that } \{\, \mathit{unique}(p)(q) \,\}\big) \tag{50a}$$
$$\text{where } \{\, p := \mathit{dog} \,\} \tag{50b}$$
$$: \widetilde{\mathsf{e}} \tag{50c}$$

Neural Networks by Type Theory of Restricted Algorithms L^{λ}_{ra}

The term N in (51a)–(51b) represents a neural network:

$$N \equiv \Big(\Big[A_0^{\sigma_0}(\overrightarrow{q}) \text{ such that } \{ C_1^{\tau_1}, \ldots, C_m^{\tau_m} \} \Big] \tag{51a}$$

$$\text{where } \{ p_1^{\sigma_1} := A_1^{\sigma_1}, \ldots, p_n^{\sigma_n} := A_n^{\sigma_n} \} \Big) \tag{51b}$$

- $A_0^{\sigma_0}(\overrightarrow{q})$ in (51a) can represent
  - the nucleus of N, which has structure over the dependent components $\overrightarrow{q}$
  - the structure of N is determined by the entire term (51a)–(51b)
  - the membrane of N
- (51b) represents procedural memory of N
- $\{ C_1^{\tau_1}, \ldots, C_m^{\tau_m} \}$ represents declarative memory of N

Early work on Situation Theory

- Barwise and Perry [1] (1983): Situation Theory as model theory of Information, incl. the idea of restricted parameters
- Loukanova (1990ties) introduced Situation Theory (SitT) as math structures with:
  - situated types as semantic objects (not as a formal language)
  - early mathematics of recursiely defined semantic parameters that are restricted with situated types
  - Cooper Storage by Situation Theory: underspecified semantic quantification
  - restricted parameters: Cooper and Loukanova, 1994
  - restricted parameters to represent space-time locations as components in information
  - restricted parameters to represent reference to individuals by proper names

A New Approach to Situation Theory: Type Theory of Algorithms & Information Content

▷ Loukanova [4] (2014) is an intro to set-theoretic foundations of SitT by the following ideas

- information in context and agents
- primitive and complex parameters
  - represent objects, by partially available information
  - represent objects that are undeveloped or in developmental stage (e.g., objects in nature)

Typed Theory of Situated Algorithms and Information

- formal syntax of situated algorithms & information

Some key points:

- theory of typed recursion (new) on relations and functions
- two kinds of typed variables, across types:
  - pure variables (corresponding to classic variables): for λ-abstractions
  - memory / recursion variables (new):
    designating typed semantic parameters and memory locations
- recursion terms (new): designating calculations, i.e., algorithms;
- restrictor terms: designating constraints
- generalized, memory networks (new) with constraints
- recursion terms with constraints: designating situated algorithms operating over structured information and objects in situations, in space-time locations

Memory Variables — Some Key Intuitions

- Memory variables designate semantic parameters (per se semantic entities that are underspecified objects)
- Values can be assigned to memory variables by recursive computations.
- The recursive computations are determined by recursion terms in the formal language L_{GP}^{ST}.
- Memory variables are constrained to satisfy situation-theoretic restrictions, expressed by recursion terms with constraints / restrictions
- Memory variables can be underspecified.
- Recursion terms can be underspecified.
- The semantic parameters designated by memory variables can be "anchored" to more specific values, which can be still parametric.
- "Anchoring" corresponds to recursive assignments and satisfies restrictions.

Computational Neuroscoence of languge and memory

From the perspective of neuroscience:

- The generalized recursion terms of TTofSitAlg and TTSitInfo represent neural nets of recursively linked memory cells for processing and saving information
- My inspiration for this:
  Kandel et al. [3] and Squire and Kandel [5]
- TTofSitAlg in the functional settings is insufficient
- I extend TTofSitAlg, to TTSitInfo by formalizing Situation Theory and extending it
- Extending the formal language of TTofSitAlg and TTSitInfo to a formal language LNNets L_{GP}^{ST}.

Primitive (basic) types of L_{GP}^{ST}: a (relatively small) set of constants

$$\begin{aligned}\text{BTypes} = \{ &\textsc{ind}, \textsc{rel}, \textsc{fun}, \textsc{ArgR}, \textsc{loc}, \textsc{pol}, \textsc{par}, \\ &\textsc{infon}, \textsc{sit}, \textsc{prop}, \textsc{set}, \textsc{type}, \models, \ldots \}\end{aligned} \tag{52}$$

- IND: for primitive and complex individuals;
- REL: for primitive and complex relations;
- FUN: for functions, primitive and complex;
- ARoles: for primitive and complex argument roles;
- LOC: for space-time locations;
- POL: for polarities 0 and 1 (these are not truth values);
- PAR: for primitive and complex parameters;
- INFON: for basic or complex information units;
- SIT: for situations;
- PROP: for basic or complex propositions;
- TYPE: for basic and complex types;
- $\models$ is a designated type
- SET: the type of sets

- $\models$ is a special type called "supports" ("holds"), e.g., used in the type of propositions that a situation s and an infon σ are of the type "supports", i.e., "s supports σ":

$$(s \models \sigma) \qquad \text{(a proposition)}$$
$$s \models \sigma \qquad \text{(a verified proposition)}$$

- A class of primitive and complex types
  - Complex types are constructed at stages, e.g., as needed (not necessarily all)

$$\text{Types}_0, \text{Types}_1, \ldots, \text{Types}_n, \ldots \tag{54a}$$
$$\text{where } \text{Types}_i \subseteq \text{Types}_{i+1}, \text{ for } i \geq 0 \tag{54b}$$
$$\tau : \text{TYPE} \iff \tau \in \text{Types}_i, \quad \text{for some } i \geq 0 \tag{54c}$$

Vocabulary of L_{GP}^{ST}

- Typed constants $K = \bigcup_{\tau \in \text{Types}} K_\tau$, where

$$K_\tau = \{ \mathsf{c}_0^\tau, \mathsf{c}_1^\tau, \ldots, \mathsf{c}_{k_\tau}^\tau \}, \text{ for } \tau \in \text{Types} \tag{55}$$

- Typed pure and memory (recursion) variables
  - pure variables (for λ-abstractions) $\text{PureVars}_\tau = \text{PV}_\tau = \{ \mathsf{v}_0^\tau, \mathsf{v}_1^\tau, \ldots \}$, for $\tau \in \text{Types}$
  - memory (recursion) variables (for memory "slots") $\text{MV}_\tau = \text{RV}_\tau = \{ \mathsf{p}_0^\tau, \mathsf{p}_1^\tau, \ldots \}$, for $\tau \in \text{Types}$
    - basic restricted memory variables (sometimes marked by dots), for saving information
- Notations for typed constants, variables, and terms (the class Terms to be defined)

$$A : \tau \iff A^\tau \in \text{Terms} \iff A \in \text{Terms}_\tau \tag{56}$$

Relations and Types with Argument Roles

- Every expression γ, denoting a relation, a function, or a type, comes with a set $\textsc{ArgR}(\gamma)$ of expressions arg_i for argument roles and types for appropriateness constraints:

$$\begin{aligned} \textsc{ArgR}(\gamma) &= \{\, T_1 : \text{arg}_1, \ldots, T_n : \text{arg}_n \,\} \\ \text{for all } \gamma &\in K_{\text{REL}} \cup K_{\text{TYPE}} \cup \text{Vars}_{\text{REL}} \cup \text{Vars}_{\text{TYPE}} \end{aligned} \tag{57}$$

$$\mathit{ArgR}(\text{smile}) = \{T_a : \text{smiler}\} \tag{58a}$$

$$\mathit{ArgR}(\text{read}) = \{T_a : \text{reader}, T_o : \text{readed}\} \tag{58b}$$

$$\mathit{ArgR}(\text{read-to}) = \{T_{a_1} : \text{reader}, \; T_m : \text{readed}, \; T_{a_2} : \text{readee}\} \tag{58c}$$

$$\mathit{ArgR}(\mathit{give}) = \{T_a : \text{giver}, \; T_r : \text{receiver}, \; T_g : \text{given}\} \tag{58d}$$

- Every function constant and function variable γ, i.e., $\gamma \in K_{\text{FUN}} \cup \text{Vars}_{\text{FUN}}$, is associated with two sets of typed expressions:

$$\text{ARGR}(\gamma) = \{ T_1 : \text{arg}_1, \ldots, T_n : \text{arg}_n \} \tag{59a}$$

$$\mathit{Value}(\gamma) = \{ T_{n+1} : \text{arg}_{n+1} \} \tag{59b}$$

- In classic notations from λ-calculi:

$$\gamma : ((T_1 \times \cdots \times T_n) \to T_{n+1}) \tag{60a}$$

$$\gamma : (T_1 \to \ldots (T_n \to T_{n+1})) \qquad \text{(currying)} \tag{60b}$$

- The full class of typed terms is defined by recursion (I do not present the full definition).
- Next I shall introduce some of the terms.

- Every function constant and function variable γ, i.e., $\gamma \in K_{\text{FUN}} \cup \text{Vars}_{\text{FUN}}$, is associated with two sets of typed expressions:

$$\text{ARGR}(\gamma) = \{ T_1 : \text{arg}_1, \ldots, T_n : \text{arg}_n \} \tag{59a}$$
$$\mathit{Value}(\gamma) = \{ T_{n+1} : \text{arg}_{n+1} \} \tag{59b}$$

- In classic notations from λ-calculi:

$$\gamma : ((T_1 \times \cdots \times T_n) \to T_{n+1}) \tag{60a}$$
$$\gamma : (T_1 \to \ldots (T_n \to T_{n+1})) \qquad \text{(currying)} \tag{60b}$$

- The full class of typed terms is defined by recursion (I do not present the full definition).
- Next I shall introduce some of the terms.

- Every function constant and function variable γ, i.e., $\gamma \in K_{\text{FUN}} \cup \text{Vars}_{\text{FUN}}$, is associated with two sets of typed expressions:

$$\text{ARGR}(\gamma) = \{ T_1 : \text{arg}_1, \ldots, T_n : \text{arg}_n \} \tag{59a}$$
$$\mathit{Value}(\gamma) = \{ T_{n+1} : \text{arg}_{n+1} \} \tag{59b}$$

- In classic notations from λ-calculi:

$$\gamma : ((T_1 \times \cdots \times T_n) \to T_{n+1}) \tag{60a}$$
$$\gamma : (T_1 \to \ldots (T_n \to T_{n+1})) \qquad \text{(currying)} \tag{60b}$$

- The full class of typed terms is defined by recursion (I do not present the full definition).
- Next I shall introduce some of the terms.

- Every function constant and function variable γ, i.e., $\gamma \in K_{\text{FUN}} \cup \text{Vars}_{\text{FUN}}$, is associated with two sets of typed expressions:

$$\text{ARGR}(\gamma) = \{ T_1 : \text{arg}_1, \ldots, T_n : \text{arg}_n \} \tag{59a}$$
$$\mathit{Value}(\gamma) = \{ T_{n+1} : \text{arg}_{n+1} \} \tag{59b}$$

- In classic notations from λ-calculi:

$$\gamma : ((T_1 \times \cdots \times T_n) \to T_{n+1}) \tag{60a}$$
$$\gamma : (T_1 \to \ldots (T_n \to T_{n+1})) \qquad \text{(currying)} \tag{60b}$$

- The full class of typed terms is defined by recursion (I do not present the full definition).
- Next I shall introduce some of the terms.

Infons: information pieces

Infon Terms: The class of expressions of the form

$$\begin{aligned} \ll \rho, & T_1 : \mathsf{arg}_1 : \xi_1, \ldots, \\ & T_n : \mathsf{arg}_n : \xi_n, \\ & \mathrm{LOC} : \mathit{Loc} : \tau, \ \mathrm{POL} : \mathit{Pol} : t \gg : \mathrm{INFON} \end{aligned}$$

for any given

- $\rho \in \mathsf{Terms}_{\mathrm{REL}}$, s.t.

$$\textsc{ArgR}(\rho) = \{\ T_1 : \mathsf{arg}_1, \ldots, T_n : \mathsf{arg}_n\ \} \tag{62}$$

- $\xi_1 \in \mathsf{Terms}_{T_1}, \ldots, \xi_n \in \mathsf{Terms}_{T_n}$
- $\tau \in \mathsf{Terms}_{\mathrm{LOC}}$
- $t \in \mathsf{Terms}_{POL}$

Example (infons: specific or parametric)

- c_a reads c_b to c_c at the space-time location l (specific objects)

$$\begin{aligned} \ll \text{read-to},\ & T_{a_1} : \text{reader} : c_a, \\ & T_m : \text{readed} : c_b, \\ & T_{a_2} : \text{readee} : c_c, \\ & \text{LOC} : Loc : l; \\ & \text{POL} : Pol : 1 \gg \end{aligned} \tag{63}$$

- c_a reads c_b to the unknown z at the unknown location $\dot{l}$

$$\begin{aligned} \ll \text{read-to},\ & T_{a_1} : \text{reader} : c_a, && \text{(specific)} \\ & T_m : \text{readed} : c_b, && \text{(specific)} \\ & T_{a_2} : \text{readee} : z, && \text{(par)} \\ & \text{LOC} : \text{Loc} : \dot{l};\ \text{POL} : \text{Pol} : 1 \gg && \text{(par)} \end{aligned}$$

Example (infons in linear notations)

- c_a reads (the unknown y to unknown z at $\dot{l}$)

$$\begin{aligned} \ll \text{read-to}, \ & T_{a_1} : \text{reader} : c_a, && \text{(specific)} \\ & T_m : \text{readed} : y, \ T_{a_2} : \text{readee} : z, && \text{(param.)} \\ & \text{LOC} : \text{Loc} : \dot{l}; \ \text{POL} : \text{Pol} : 1 \gg && \text{(param.)} \end{aligned}$$

- the info whether c_a either reads or does not available with unknown polarity $\dot{p}$

$$\begin{aligned} \ll \text{read-to}, \ & T_{a_1} : \text{reader} : c_a, && \text{(specific)} \\ & T_m : \text{readed} : y, \ T_{a_2} : \text{readee} : z, && \text{(param.)} \\ & \text{LOC} : \text{Loc} : \dot{l}; \ \text{POL} : \text{Pol} : \dot{p} \gg && \text{(param.)} \end{aligned}$$

Proposition Terms

For every term (basic or complex) $\gamma \in \text{Terms}_{\text{TYPE}}$, with argument roles

$$\text{ARGR}(\gamma) = \{\ T_1 : \text{arg}_1, \ldots, T_n : \text{arg}_n\ \} \tag{67}$$

and every $\xi_1 \in \text{Terms}_{T_1}, \ldots, \xi_n \in \text{Terms}_{T_n}$,

- a proposition term in postfix notation

$$(\{\ T_1 : \text{arg}_1 : \xi_1, \ldots, T_n : \text{arg}_n : \xi_n\}\ :\ \gamma)\ \in\ \text{Terms}_{\text{PROP}} \tag{68a}$$
$$(\{\ T_1 : \text{arg}_1 : \xi_1, \ldots, T_n : \text{arg}_n : \xi_n\}\ :\ \gamma)\ :\ \text{PROP} \qquad \text{(not-n)} \tag{68b}$$

- a proposition term in prefix notation

$$(\gamma,\ \{\ T_1 : \text{arg}_1 : \xi_1, \ldots, T_n : \text{arg}_n : \xi_n\ \}) : \text{PROP} \tag{69a}$$
$$(\gamma,\ T_1 : \text{arg}_1 : \xi_1, \ldots, T_n : \text{arg}_n : \xi_n) : \text{PROP} \tag{69b}$$

Definition (Terms of Situated Propositions)

- The type $\models$ ("supports"):

$$ArgR(\models) = \{\text{SIT} : arg_{\text{SIT}},\ \text{INFON} : arg_{\text{INFON}}\} \tag{70}$$

- Terms of situated propositions: for any

$$s \in \text{PureVars}_{\text{SIT}} \cup \text{MV}_{\text{SIT}} \tag{71a}$$
$$\sigma \in \text{Terms}_{\text{INFON}} \tag{71b}$$

  - In prefix notation

$$(\models, s, \sigma) \tag{72}$$

  - In infix notation

$$(s \models \sigma) \tag{73}$$

Example (The proposition that s supports a positive infon)

$$(s \models \ll book, \text{IND} : arg : b, \tag{74a}$$
$$\text{LOC} : Loc : l; \text{POL} : Pol : 1 \gg) \tag{74b}$$

Example (The proposition that s supports a negative infon)

$$(s \models \ll book, \text{IND} : arg : b, \tag{75a}$$
$$\text{LOC} : Loc : l; \text{POL} : Pol : 0 \gg) \tag{75b}$$

Example (The situation s does not support a positive infon)

$$(s \not\models \ll \mathit{book}, \textsc{ind} : \mathit{arg} : b, \tag{76a}$$
$$\textsc{loc} : \mathit{Loc} : l; \textsc{pol} : \mathit{Pol} : 1 \gg) \tag{76b}$$

Example (The situation s does not support a negative infon)

$$(s \not\models \ll \mathit{book}, \textsc{ind} : \mathit{arg} : b, \tag{77a}$$
$$\textsc{loc} : \mathit{Loc} : l; \textsc{pol} : \mathit{Pol} : 0 \gg) \tag{77b}$$

Example (A situation s can "carry" partial information)

$$(s \not\models \ll book,\ b,\ l;\ 1 \gg) \tag{78a}$$
$$(s \not\models \ll book,\ b,\ l;\ 0 \gg) \tag{78b}$$

Both propositions (78a) and (78b) can be true.

Terms for complex propositions can be formed by using the usual logic connectives, $\neg$, $\wedge$, $\vee$, etc.

Example (conjunctive propositions)

- the situation is the same

$$(s \models \ll \mathit{smiles},\ \text{IND} : \mathit{arg} : a,\ l; 1 \gg) \tag{79a}$$
$$(s \models \ll \mathit{cries},\ \text{IND} : \mathit{arg} : a,\ l_2; 1 \gg) \tag{79b}$$
$$\wedge\, (s \models \ll \mathit{animate},\ \text{IND} : \mathit{arg} : a,\ l_1; 1 \gg) \tag{79c}$$
$$\wedge\, (l \circ l_1) \wedge (l \circ l_2) \tag{79d}$$

- different situations

$$(s \models \ll \mathit{smiles},\ \text{IND} : \mathit{arg} : a,\ l; 1 \gg) \tag{80a}$$
$$(s_2 \models \ll \mathit{cries},\ \text{IND} : \mathit{arg} : a,\ l_2; 1 \gg) \tag{80b}$$
$$\wedge\, (s_1 \models \ll \mathit{animate},\ \text{IND} : \mathit{arg} : a,\ l_1; 1 \gg) \tag{80c}$$
$$\wedge\, (l \circ l_1) \wedge (l \circ l_2) \tag{80d}$$

Terms for complex propositions can be formed by using the usual logic connectives, $\neg$, $\wedge$, $\vee$, etc.

Example (conjunctive propositions)

- the situation is the same

$$(s \models \ll \mathit{smiles},\ \text{IND} : \mathit{arg} : a,\ l; 1 \gg) \tag{79a}$$
$$(s \models \ll \mathit{cries},\ \text{IND} : \mathit{arg} : a,\ l_2; 1 \gg) \tag{79b}$$
$$\wedge\, (s \models \ll \mathit{animate},\ \text{IND} : \mathit{arg} : a,\ l_1; 1 \gg) \tag{79c}$$
$$\wedge\, (l \circ l_1) \wedge (l \circ l_2) \tag{79d}$$

- different situations

$$(s \models \ll \mathit{smiles},\ \text{IND} : \mathit{arg} : a,\ l; 1 \gg) \tag{80a}$$
$$(s_2 \models \ll \mathit{cries},\ \text{IND} : \mathit{arg} : a,\ l_2; 1 \gg) \tag{80b}$$
$$\wedge\, (s_1 \models \ll \mathit{animate},\ \text{IND} : \mathit{arg} : a,\ l_1; 1 \gg) \tag{80c}$$
$$\wedge\, (l \circ l_1) \wedge (l \circ l_2) \tag{80d}$$

Example (conjunctive information)

- a conjunctive infon in a proposition

$$(s \models \quad \ll \mathit{smiles},\ \textsc{ind} : \mathit{arg} : a,\ \textsc{loc} : \mathit{Loc} : l; 1 \gg \tag{81a}$$
$$\wedge \ \ll \mathit{animate},\ \textsc{ind} : \mathit{arg} : a,\ l_1; 1 \gg \tag{81b}$$
$$\wedge \ l \circ l_1) \tag{81c}$$

- a conjunctive proposition

$$(s \models \ll \mathit{smiles},\ \textsc{ind} : \mathit{arg} : a,\ l; 1 \gg) \tag{82a}$$
$$\wedge (s \models \ll \mathit{animate},\ \textsc{ind} : \mathit{arg} : a,\ l_1; 1 \gg) \tag{82b}$$
$$\wedge (l \circ l_1) \tag{82c}$$

Example (conjunctive information)

- a conjunctive infon in a proposition

$$(s \models \quad \ll \mathit{smiles},\ \textsc{ind} : \mathit{arg} : a,\ \textsc{loc} : \mathit{Loc} : l; 1 \gg \tag{81a}$$
$$\wedge \ \ll \mathit{animate},\ \textsc{ind} : \mathit{arg} : a,\ l_1; 1 \gg \tag{81b}$$
$$\wedge \ l \circ l_1) \tag{81c}$$

- a conjunctive proposition

$$(s \models \ll \mathit{smiles},\ \textsc{ind} : \mathit{arg} : a,\ l; 1 \gg) \tag{82a}$$
$$\wedge (s \models \ll \mathit{animate},\ \textsc{ind} : \mathit{arg} : a,\ l_1; 1 \gg) \tag{82b}$$
$$\wedge (l \circ l_1) \tag{82c}$$

Example

- The propositional content of the sentence (83) might be expressed by the proposition (84a)–(84c), with some (great) approximation.

$$\text{The book } b \text{ is read} \tag{83}$$

$$\begin{aligned}
(s \models \quad & \ll \mathit{read},\ \mathit{reader} : \dot{x},\ \mathit{readed} : b,\ \mathit{readee} : \dot{y}, && \text{(84a)}\\
& \mathit{Loc} : l; 1 \gg \\
\wedge \quad & \ll \mathit{book},\ \mathit{arg} : b,\ \mathit{Loc} : l_1; 1 \gg) && \text{(84b)}\\
\wedge \quad & (l \subset l_1) && \text{(84c)}
\end{aligned}$$

(84b) and (84c) are presented as parts of the propositional content of (83). There are other ways to include this information (later).

λ-Terms

For every term (basic or complex) $\Phi \in$ Terms and any pure variables $\xi_1, \ldots, \xi_n \in$ PV, the expression $\lambda\{\, \xi_1, \ldots, \xi_n \,\}\, \Phi$ is a λ-*abstraction term*, i.e.:

$$\lambda\{\, \xi_1, \ldots, \xi_n \,\}\, \Phi \in \text{Terms} \tag{85a}$$

$[\xi_1], \ldots, [\xi_n]$ are expressions for the argument roles of $\lambda\{\, \xi_1, \ldots, \xi_n \,\}\Phi$, and are associated with corresponding *appropriateness constraints* as follows:

$$\begin{aligned} \textsc{ArgR}(\lambda\{\, \xi_1, \ldots, \xi_n \,\}\, \Phi) = \\ \{\, T_1 : [\xi_1], \ldots, T_n : [\xi_n] \,\} \end{aligned} \tag{86}$$

where, for each $i \in \{\, 1, \ldots, n \,\}$, T_i is the union of all sets (of types) that are the appropriateness constraints of all the argument roles that occur in Φ, and such that ξ_i fills up them, without being bound. (Note that ξ_i may fill more than one argument role in Φ.)

Complex Relations

Case 1: complex relations with complex argument roles. In the case when $\Phi \in \text{Terms}_{\text{INFON}}$ the expression $\lambda\{\xi_1, \ldots, \xi_n\}\,\Phi$ is a *complex-relation term*, i.e.:

$$\lambda\{\xi_1, \ldots, \xi_n\}\,\Phi \in \text{Terms}_{\text{REL}} \tag{87a}$$

$$\lambda\{\xi_1, \ldots, \xi_n\}\,\Phi : \text{REL} \tag{87b}$$

Example

$$R_1 \equiv \lambda(x)\Big[\ll \text{read-to}, \text{reader} : \text{x}, \text{readed} : \text{b}, \text{readee} : z, \text{Loc} : \text{l}; \text{Pol} : 1 \gg \Big] \tag{88a}$$

$$R_2 \equiv \lambda(x, z)\Big[\ll \text{read-to}, \text{reader} : \text{x}, \text{readed} : \text{b}, \text{readee} : z, \text{Loc} : \text{l}; \text{Pol} : 1 \gg \Big] \tag{88b}$$

Example (Underspecified complex infons in linear notation)

- $\mathsf{b}, z \in \mathrm{RV}_{\mathrm{IND}}$ are recursion (memory) variables
- $\mathsf{l} \in \mathrm{RV}_{\mathrm{LOC}}$ is a recursion (memory) variable for space-time location
- $\mathsf{x} \in \mathrm{PV}_{\mathrm{IND}}$ is a pure variable for an individual

$$I \equiv \ll \mathsf{book}, \mathsf{arg} : \mathsf{b}, \mathsf{Loc} : \mathsf{l}; \mathsf{Pol} : 1 \gg \wedge \tag{89a}$$

$$\ll \mathsf{read\text{-}to}, \mathsf{reader} : \mathsf{x}, \mathsf{readed} : \mathsf{b}, \mathsf{readee} : z, \tag{89b}$$

$$\mathsf{Loc} : \mathsf{l}; \mathsf{Pol} : 1 \gg$$

$$R \equiv \lambda(x, z)\Big[\ll \mathsf{book}, \mathsf{arg} : \mathsf{b}, \mathsf{Loc} : \mathsf{l}; \mathsf{Pol} : 1 \gg \wedge \tag{90a}$$

$$\ll \mathsf{read\text{-}to}, \mathsf{reader} : \mathsf{x}, \mathsf{readed} : \mathsf{b}, \mathsf{readee} : z, \tag{90b}$$

$$\mathsf{Loc} : \mathsf{l}; \mathsf{Pol} : 1 \gg \Big]$$

Note: (89a)–(89b) is a complex infon, not a proposition!

Complex Types

Case 2: complex types with complex argument roles.
In the case when $\Phi \in \text{Terms}_{\text{PROP}}$, the expression $\lambda\{\, \xi_1, \ldots, \xi_n \,\}\, \Phi$ is a *complex-type term*, i.e.:

$$\lambda\{\, \xi_1, \ldots, \xi_n \,\}\, \Phi \in \text{Terms}_{\text{TYPE}} \tag{91a}$$
$$\lambda\{\, \xi_1, \ldots, \xi_n \,\}\, \Phi : \text{TYPE} \tag{91b}$$

- Notations:

$$[T_1 : \xi_1, \ldots, T_n : \xi_n \mid \Phi] \in \text{Terms}_{\text{TYPE}} \tag{92a}$$
$$[T_1 : \xi_1, \ldots, T_n : \xi_n \mid \Phi] : \text{TYPE} \tag{92b}$$

The expression in (93) is a recursion term of type σ_0:

$$\left[\mathsf{A}_0 \text{ where } \{\, \mathsf{p}_1 := \mathsf{A}_1, \ldots, \mathsf{p}_n := \mathsf{A}_n \,\}\right] : \sigma_0 \tag{93}$$

for any terms $\mathsf{A}_i : \sigma_i$, $i = 0, \ldots, n$ $(n \geq 0)$, and pairwise different memory variables $\mathsf{p}_i \in \mathrm{MV}_{\sigma_i}$ $i = 1, \ldots, n$, such that the set of the assignments (94)

$$\{\, \mathsf{p}_1 := \mathsf{A}_1, \ldots, \mathsf{p}_n := \mathsf{A}_n \,\} \tag{94}$$

satisfies the Acyclicity Constraint RT,

Definition (Acyclicity Constraint RT)

The assignments $\{\, \mathsf{p}_1 := \mathsf{A}_1, \ldots, \mathsf{p}_n := \mathsf{A}_n \,\}$ are *acyclic* iff there is a rank function

$$\mathrm{rank} \colon \{\bigcup \{\, \mathsf{p}_i \,\}_{i=1}^{n}\} \longrightarrow \mathbb{N} \tag{95}$$

such that:

- if p_j occurs freely in A_i, then $\mathrm{rank}(\mathsf{p}_j) < \mathrm{rank}(\mathsf{p}_i)$

The expression in (96) is a restriction (constraned) term of type σ_0:

$$\Big[\mathsf{A}_0 \text{ such that } \{\, (\mathsf{q}_{1,1}, \ldots, \mathsf{q}_{1,l_1} : \mathsf{C}_1), \ldots, (\mathsf{q}_{m,1}, \ldots, \mathsf{q}_{m,l_m} : \mathsf{C}_m) \,\}\Big] : \sigma_0 \tag{96}$$

for

- any type terms $\mathsf{C}_k : \textsc{type}$, $k = 1, \ldots, m$ $(m \geq 0)$ that have argument roles:

$$\textsc{ArgR}(\mathsf{C}_k) = \{\, T_{k,1} : arg_{k,l_1}, \ldots, T_{k,l_k} : arg_{k,l_k} \,\} \tag{97}$$

- any memory (recursion) variables $\mathsf{q}_{k,j} \in \mathsf{MV}_{T_{k,j}}$, $j = 1, \ldots, l_k$,
- and any term $\mathsf{A}_0 : \sigma_0$,
  such that the set of propositions (98)

$$\{\, (\mathsf{q}_{1,1}, \ldots, \mathsf{q}_{1,l_1} : \mathsf{C}_1), \ldots, (\mathsf{q}_{m,1}, \ldots, \mathsf{q}_{m,l_m} : \mathsf{C}_m) \,\} \tag{98}$$

  satisfies the Acyclicity Constraint CT (99), p. 68,

Note: If the acyclicity is dropped, we have a formal language with 'cyclic' constraints.

Definition (Acyclicity Constraint CT)

A set of propositions (99):

$$\{ (q_{1,1}, \ldots, q_{1,l_1} : C_1), \ldots, (q_{m,1}, \ldots, q_{m,l_m} : C_m) \} \tag{99}$$

is *acyclic* iff there is a rank function

$$\text{rank}: \{ \bigcup_{k=1}^{m} \{ q_{k,j} \}_{j=1}^{l_k} \} \longrightarrow \mathbb{N} \tag{100}$$

such that:

- if $q_{k',j'}$ occurs freely in C_k, then $\text{rank}(q_{k',j'}) < \text{rank}(q_{k,j})$

The expression in (101) is a restricted-recursion term of type σ_0:

$$\begin{aligned}\big[\big[\mathsf{A}_0 \text{ such that } \{\, &(\mathsf{q}_{1,1},\ldots,\mathsf{q}_{1,l_1} : \mathsf{C}_1),\ldots,\\ &(\mathsf{q}_{m,1},\ldots,\mathsf{q}_{m,l_m} : \mathsf{C}_m)\,\}\ \big]\\ \text{where } \{\, &\mathsf{p}_1 := \mathsf{A}_1,\ldots,\mathsf{p}_n := \mathsf{A}_n\,\}\big] : \sigma_0\end{aligned} \tag{101}$$

for

- type terms $\mathsf{C}_k : \textsc{type}$, $k = 1,\ldots,m$ $(m \geq 0)$ with argument roles:

$$\textsc{Argr}(\mathsf{C}_k) = \{\, T_{k,1} : arg_{k,l_1},\ldots,T_{k,l_k} : arg_{k,l_k}\,\} \tag{102}$$

- memory (recursion) variables (not necessarily pairwise different): $\mathsf{q}_{k,j} \in \mathsf{MV}_{T_{k,j}}$, $j = 1,\ldots,l_k$,
- terms $\mathsf{A}_i : \sigma_i$, $i = 0,\ldots,n$ $(n \geq 0)$,
- pairwise different memory (recursion) variables: $\mathsf{p}_i \in \mathsf{MV}_{\sigma_i}$ $i = 1,\ldots,n$,
  - such that the sequences (103a) and (103b) jointly satisfy the Acyclicity Constraint-CRT

$$\{\, (\mathsf{q}_{1,1},\ldots,\mathsf{q}_{1,l_1} : \mathsf{C}_1),\ldots,(\mathsf{q}_{m,1},\ldots,\mathsf{q}_{m,l_m} : \mathsf{C}_m)\,\} \tag{103a}$$

$$\{\, \mathsf{p}_1 := \mathsf{A}_1,\ldots,\mathsf{p}_n := \mathsf{A}_n\,\} \tag{103b}$$

Definition (Acyclicity Constraint-CRT)

$$\{ (q_{1,1}, \ldots, q_{1,l_1} : C_1), \ldots, (q_{m,1}, \ldots, q_{m,l_m} : C_m) \} \tag{104a}$$

$$\{ p_1 := A_1, \ldots, p_n := A_n \} \tag{104b}$$

(104a) and (104b) are (jointly) *acyclic* iff there is a rank function

$$\text{rank} : \{ \bigcup_{k=1}^{m} \{ q_{k,j} \}_{j=1}^{l_k} \cup \bigcup \{ p_i \}_{i=1}^{n} \} \longrightarrow \mathbb{N} \tag{105}$$

such that:

1. if $q_{k',j'}$ occurs freely in C_k, then $\text{rank}(q_{k',j'}) < \text{rank}(q_{k,j})$
2. if p_j occurs freely in A_i, then $\text{rank}(p_j) < \text{rank}(p_i)$
3. if p_i occurs freely in C_k, then $\text{rank}(p_i) < \text{rank}(q_{k,j})$
4. if $q_{k,j}$ occurs freely in A_i, then $\text{rank}(q_{k,j}) < \text{rank}(p_i)$

By dropping any of the acycliciy conditions, we obtain a corresponding class of formal languages.

Example (restricted-memory variables: for denoting semantic parameters)

Restricted-memory variables are a special case of restricted-recursion terms.

- q is a restricted-memory variable

$$\{q\} \text{ such that } \{ (C : q) \} \tag{106a}$$
$$\text{where} \quad \{ C := \lambda(y)\big(s \models\!\ll \text{read-to}, \tag{106b}$$
$$T_{a_1} : \text{reader} : c_a, \tag{106c}$$
$$T_m : \text{readed} : y, \tag{106d}$$
$$T_{a_2} : \text{readee} : z, \tag{106e}$$
$$\text{LOC} : \text{Loc} : l; \tag{106f}$$
$$\text{POL} : \text{Pol} : 1 \gg \big) \} \tag{106g}$$

The language $\mathrm{L}_{\mathrm{GP}}^{\mathrm{ST}}$ / LNNets has terms for propositions:

$$(\{\, arg_1 : p_1, \ldots, arg_m : p_m \,\} : T) \tag{107a}$$
$$\text{for } T : \mathrm{TYPE}, \tag{107b}$$
$$\mathrm{ARGR}(T) = \{\, arg_1, \ldots, arg_m \,\}, \tag{107c}$$
$$p_1, \ldots, p_m : \mathrm{MV} \tag{107d}$$

The proposition term (107a) designates simultaneous restrictions over sets of objects: $p_1, \ldots, p_m$.

Definition (Memory Network / Restricted Memory Net)

$$\{\mathsf{q}_{k_1}, \ldots, \mathsf{q}_{k_l}\} \text{ s.t. } \{\, (\mathsf{q}_1, \ldots, \mathsf{q}_k : \mathsf{C}), \; (\overrightarrow{\mathsf{q}} : \overrightarrow{\mathsf{C}}) \,\} \tag{108}$$
$$\text{where } \{\, \overrightarrow{\mathsf{p}} := \overrightarrow{\mathsf{A}} \,\}$$

Reduction Calculus

Reduction Calculus: an effective system of reduction rules

- for reducing every term A to a canonical form $\mathrm{cf}(A)$
- $\mathrm{cf}(A)$ of a meaningful term A determines the algorithm for computing $\mathrm{den}(A)$, in a step-by-step mode, from the simplest components.
- The algorithm designated by a term of restricted recursion respects the restrictions.

Generalized, restricted recursion variables: memory variables

Example (recursion variables: for denoting semantic parameters)

- $\{ q, \dot{l} \}$ is a generalized, restricted recursion variable, i.e., a restricted memory net

$$\{q, \dot{l}\} \text{ s.t. } \{ (C : (q, \dot{l})) \} \tag{109a}$$

$$\text{where } \{ C := \lambda(y, l)\big(s \models\!\ll \text{read-to}, \tag{109b}$$

$$T_{a_1} : \text{reader} : c_a, \tag{109c}$$

$$T_m : \text{readed} : y, \tag{109d}$$

$$T_{a_2} : \text{readee} : z, \tag{109e}$$

$$\text{LOC} : \text{Loc} : l; \tag{109f}$$

$$\text{POL} : \text{Pol} : 1 \gg \big) \} \tag{109g}$$

Complex Relations and Propositions

Example (A proposition with a basic infon having a complex relation)

$$
\begin{aligned}
&(s \models && && (110a)\\
&\quad \ll \lambda\, x, y, z \,\big[\ll \text{read-to}, \text{reader} : x, \text{readed} : y, && && (110b)\\
&\qquad\qquad \text{readee} : z, \text{Loc} : l; 1 \gg \wedge && \\
&\qquad\quad \ll \text{book}, \text{arg} : y, \text{Loc} : l_1; 1 \gg \ \wedge && && (110c)\\
&\qquad\quad \ll \text{listen}, \text{arg} : z, \text{Loc} : l_2; 1 \gg \ \wedge && && (110d)\\
&\qquad\quad l \subseteq l_1 \ \wedge \ l_2 \subseteq_t l \big], && && (110e)\\
&\quad [x] : a,\ [y] : b,\ [z] : c,\ \text{Loc} : l;\ 1 \gg) && && (110f)
\end{aligned}
$$

The complex relation (110b)–(110e) has new, its own, argument-roles denoted by $[x]$, $[y]$, and $[z]$, which are filled up by the objects denoted by a, b, and c, respectively.

$$P \text{ where } \{ \tag{111a}$$

$$P := (s \models [\tag{111b}$$

$$\ll r_1, [x'] : a, [y'] : b, [z'] : c; 1 \gg \ \wedge \tag{111c}$$

$$\ll r_2, [y'] : b; 1 \gg \ \wedge \tag{111d}$$

$$\ll r_3, [z'] : c; 1 \gg \ \wedge \tag{111e}$$

$$i_4 \ \wedge \ i_5]), \tag{111f}$$

$$r_1 := \lambda x', y', z' \ \ll \text{read-to}, \text{reader} : x', \text{readed} : y', \text{readee} : z', \text{Loc} : l; 1 \gg, \tag{111g}$$

$$r_2 := \lambda y' \ll \text{book}, \text{arg} : y', \text{Loc} : l_1; 1 \gg, \tag{111h}$$

$$r_3 := \lambda z' \ \ll \text{listen}, \text{arg} : z', \text{Loc} : l_2; 1 \gg, \tag{111i}$$

$$i_4 := (l \subseteq l_1), \tag{111j}$$

$$i_5 := (l_2 \subseteq_t l) \} \tag{111k}$$

- T_0 is the proposition that a, b, c are in relation $R \in \text{RV}$ in the situation $s \in \text{Vars}$.

$$T_0 \equiv \left(\text{s} \models\!\ll R,\ [\text{x}] : \text{a},\ [\text{y}] : \text{b},\ [\text{z}] : \text{c},\ \text{Loc} : \text{l};\ 1 \gg\right) \tag{112}$$

- T_1 is the proposition that a, b, c are in relation $R \in \text{RV}$ in the situation $s \in \text{Vars}$, where R is a complex, parametric relation with r_1, r_2, r_3 as sub-relations in a conjunctive infon.

$$T_1 \equiv \left(\text{s} \models\!\ll R,\ [\text{x}] : \text{a},\ [\text{y}] : \text{b},\ [\text{z}] : \text{c},\ \text{Loc} : \text{l};\ 1 \gg\right) \tag{113a}$$

$$\text{where } \{\, R := \ \ll \lambda\, \text{x}, \text{y}, \text{z} \left[\right. \tag{113b}$$

$$\ll \text{r}_1, [\text{x}'] : \text{x}, [\text{y}'] : \text{y}, [\text{z}'] : \text{z}; 1 \gg \ \wedge \tag{113c}$$

$$\ll \text{r}_2, [\text{x}'] : \text{x}, [\text{y}'] : \text{y}, [\text{z}'] : \text{z}; 1 \gg \ \wedge \tag{113d}$$

$$\ll \text{r}_3, [\text{x}'] : \text{x}, [\text{y}'] : \text{y}, [\text{z}'] : \text{z}; 1 \gg \ \wedge \tag{113e}$$

$$\left. \text{i}_4 \ \wedge \ \text{i}_5 \right] \,\} \tag{113f}$$

- (114a)–(114f) is a complex proposition containing restrictions
  - via propositional constraints
  - assignments

$$T \text{ s.t. } \{ (\mathsf{l} : \textsc{loc}), (\mathsf{l}_1 : \textsc{loc}), (\mathsf{l}_2 : \textsc{loc}), (\mathsf{a} : \mathsf{T}), (\mathsf{c} : \mathsf{T}), \tag{114a}$$

$$\mathsf{l} \subseteq \mathsf{l}_1,\ \mathsf{l}_2 \subseteq_t \mathsf{l} \} \tag{114b}$$

$$\text{where } \{\mathsf{r}_1 := \lambda\, \mathsf{x}', \mathsf{y}', \mathsf{z}' \ll \text{read-to}, \text{reader} : \mathsf{x}', \text{readed} : \mathsf{y}', \tag{114c}$$

$$\text{readee} : \mathsf{z}', \text{Loc} : \mathsf{l}; 1 \gg,$$

$$\mathsf{r}_2 := \lambda\, \mathsf{x}', \mathsf{y}', \mathsf{z}' \ll \text{book}, \text{arg} : \mathsf{y}', \text{Loc} : \mathsf{l}_1; 1 \gg, \tag{114d}$$

$$\mathsf{r}_3 := \lambda\, \mathsf{x}', \mathsf{y}', \mathsf{z}' \ll \text{listen}, \text{arg} : \mathsf{z}', \text{Loc} : \mathsf{l}_2; 1 \gg, \tag{114e}$$

$$\mathsf{T} := \lambda\, \mathsf{u}(\mathsf{s} \models\!\ll \text{person}, \mathsf{u}, \mathsf{l}, 1 \gg) \} \tag{114f}$$

Existing and potential applications

- Typed syntax-semantics interfaces for information representation
  - programming languages
  - algorithm specifications: higher-order type theory of algorithms
  - data science / database
- Computational semantics
- Syntax-semantics interface in computational grammar of human language
- Applications to:
  - Language Processing / Technology
  - AI
  - Neuroscience
  - Life sciences

THANKS!

Some References I


Jon Barwise and John Perry.
Situations and Attitudes.
Cambridge, MA:MIT press, 1983.
Republished as [2].


Jon Barwise and John Perry.
Situations and Attitudes.
The Hume Series. CSLI Publications, Stanford, California, 1999.


Eric Kandel, Thomas Jessell, Steven Siegelbaum, James Schwartz, and A. J. Hudspeth, editors.
Principles of neural science.
McGraw-Hill, Health Professions Division, 2000.
URL: `http://www.principlesofneuralscience.com`.

Some References II


Roussanka Loukanova.
Situation Theory, Situated Information, and Situated Agents.
In Ngoc Thanh Nguyen, Ryszard Kowalczyk, Ana Fred, and Filipe Joaquim, editors, *Transactions on Computational Collective Intelligence XVII*, volume 8790 of *Lecture Notes in Computer Science*, pages 145–170. Springer Berlin Heidelberg, 2014.
URL: http://dx.doi.org/10.1007/978-3-662-44994-3_8, doi:10.1007/978-3-662-44994-3_8.


Larry Squire and Eric Kandel.
Memory: From Mind to Molecules.
Roberts & Co., 2009.