Type-Theory of Parametric Algorithms with Restricted Computations

Roussanka Loukanova

Institute of Mathematics and Informatics, Bulgarian Academy of Sciences, Bulgaria
Dept of Mathematics, Stockholm University, Sweden

ONLINE
17th International Conference on Distributed Computing and Artificial Intelligence | L'Aquila (Italy), 7th–9th October, 2020
https://www.dcai-conference.net

DCAI 5, 11:45am - 12:00pm, Oct 9 2020

- Moschovakis [2], 2006, introduced:
    - Type-Theory of Acyclic Algorithms, $\mathrm{L}_{\mathrm{ar}}^{\lambda}$
      by demonstrating it with examples for:
      Computational Semantics
      of Natural Language (NL), i.e., Human Language (HL)

- This paper and its presentation are about development of:
    - Type-Theory of Acyclic Algorithms, $\mathrm{L}_{\mathrm{ar}}^{\lambda}$:
      Typed Full Recursion without Acyclicity $\mathrm{L}_{\mathrm{r}}^{\lambda}$
      as a new approach to the mathematical notion of algorithm, via:
        - Moschovakis (acyclic) recursion for:
          - computations, by saving the algorithmic steps in memory locations (e.g., for use and reuse)
        - parametric algorithms that can be instantiated
        - a new restrictor operator for:
          - constrained computations
          - restricted memory locations, as generalised, restricted parameters

Gallin Types: $\sigma :\equiv \mathsf{e} \mid \mathsf{t} \mid \mathsf{s} \mid (\tau_1 \to \tau_2)$ (Gallin, 1975)
For all $\tau \in$ Types:
Constants: $\mathsf{Consts}_\tau = \{\mathsf{c}_0^\tau, \mathsf{c}_1^\tau, \ldots, \mathsf{c}_{k_\tau}^\tau\}$
Variables: $\mathsf{PureV}_\tau = \{v_0^\tau, v_1^\tau, \ldots\}$,
$\mathsf{MemoryV}_\tau = \mathsf{RecV}_\tau = \{p_0^\tau, p_1^\tau, \ldots\}$

Terms of $\mathrm{L}_{\mathrm{ar}}^\lambda$ ($\mathrm{L}_{\mathrm{r}}^\lambda$):

$$\mathsf{A} :\equiv \mathsf{c}^\tau : \tau \mid x^\tau : \tau \quad (\text{for } \mathsf{c}^\tau \in \mathsf{Consts}_\tau,\ x^\tau \in \mathsf{PureV}_\tau \cup \mathsf{RecV}_\tau\ ) \tag{1a}$$

$$\mid \mathsf{B}^{(\sigma\to\tau)}(\mathsf{C}^\sigma) : \tau \tag{1b}$$

$$\mid \lambda(v^\sigma)\,(\mathsf{B}^\tau) : (\sigma \to \tau) \quad (\text{for } v^\sigma \in \mathsf{PureV}_\sigma\ ) \tag{1c}$$

$$\mid \big[\mathsf{A}_0^{\sigma_0} \text{ where } \{\, p_1^{\sigma_1} := \mathsf{A}_1^{\sigma_1}, \ldots, p_i^{\sigma_i} := \mathsf{A}_i^{\sigma_i}, \ldots, p_n^{\sigma_n} := \mathsf{A}_n^{\sigma_n} \,\}\big] : \sigma_0 \tag{1d}$$

$$\mid \big[\mathsf{A}_0^{\sigma_0} \text{ such that } \{\, \mathsf{C}_1^{\tau_1}, \ldots, \mathsf{C}_m^{\tau_m} \,\}\big] : \sigma_0' \tag{1e}$$

- $\mathsf{B}, \mathsf{C} \in$ Terms, $p_i^{\sigma_i} \in \mathsf{RecV}_{\sigma_i}$, $A_i^{\sigma_i} \in \mathsf{Terms}_{\sigma_i}$
  $\mathsf{C}_j^{\tau_j} \in \mathsf{Terms}_{\tau_j}$ (for propositions): $\tau_j \equiv \mathsf{t}$ or $\tau_j \equiv \widetilde{\mathsf{t}} \equiv (\mathsf{s} \to \mathsf{t})$
- Acyclicity Constraint:
  $\{\, p_1^{\sigma_1} := \mathsf{A}_1^{\sigma_1}, \ldots, p_i^{\sigma_i} := \mathsf{A}_i^{\sigma_i}, \ldots, p_n^{\sigma_n} := \mathsf{A}_n^{\sigma_n} \,\}$ is acyclic iff:
  - there is a rank: $\{p_1, \ldots, p_n\} \to \mathbb{N}$ such that:
    if $p_j \in \mathsf{FreeVars}(A_i)$ then $\mathsf{rank}(p_i) > \mathsf{rank}(p_j)$

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

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

- The denotational semantics is by structural induction on the terms
- The algorithmic semantics is via the reduction calculus of $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ / $\mathrm{L}^{\lambda}_{\mathrm{r}}$
  1. The reduction rules define the reduction relation

     $$A \Rightarrow B \tag{2}$$

  2. The reduction calculus (by reduction rules) is effective:
     Every $A \in \mathsf{Terms}_{\sigma}$ can be reduced to its unique, up to congruence, canonical form $\mathsf{cf}(A) \in \mathsf{Terms}_{\sigma}$:

     $$A \Rightarrow_{\mathsf{cf}} \mathsf{cf}(A) \tag{3}$$

  3. For every algorithmically meaningful $A \in \mathsf{Terms}_{\sigma}$, $\mathsf{cf}(A)$ determines the algorithm $\mathsf{alg}(A)$ for computing $\mathsf{den}(A)$

- (4b)–(4c) determines the algorithm for computing den(A):

$$A \equiv (200 + 40)/6 \tag{4a}$$

$$\Rightarrow_{\text{cf}} \underbrace{n/d \text{ where } \{\, n := (a_1 + a_2)}_{\text{parametric part of an algorithm}}, \tag{4b}$$

$$\underbrace{a_1 := 200,\ a_2 := 40,\ d := 6}_{\text{algorithmic instantiation of memory slots}} \ \} \tag{4c}$$

- (5b)–(5c) determines the algorithm for computing den(B):

$$B \equiv (120 + 120)/6 \tag{5a}$$

$$\Rightarrow_{\text{cf}} \underbrace{n/d \text{ where } \{\, n := (a_1 + a_2)}, \tag{5b}$$

$$\underbrace{a_1 := 120,\ a_2 := 120,\ d := 6}\, \} \tag{5c}$$

- (6) determines the algorithm for computing den(C):

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

cf(A), cf(B), cf(C) designate algorithms for computing den(40):

$$\text{den}(A) = \text{den}(B) = \text{den}(C) = \text{den}(40) \quad \text{(decimal num. system)} \tag{7a}$$

$$\text{alg}(A) \neq \text{alg}(B) \neq \text{alg}(C) \tag{7b}$$

- Recursion terms with restrictor operator designated by such that:

$$D_1 \equiv \underbrace{\big(n/d \text{ such that } \{\, n, d \in \mathbb{N},\ d \neq 0 \,\}\big)}_{\text{restrictor term R}} \text{ where } \{ \tag{8a}$$

$$n := (a_1 + a_2),\ d := (d_1 \times d_2), \tag{8b}$$

$$a_1 := 200,\ a_2 := 40,\ d_1 := 2,\ d_2 := 3 \,\} \tag{8c}$$

- The restriction unsatisfied:

$$E_1 \equiv \underbrace{\big(n/d \text{ such that } \{\, n, d \in \mathbb{N},\ d \neq 0 \,\}\big)}_{\text{restrictor term R}} \text{ where } \{ \tag{9a}$$

$$n := (a_1 + a_2),\ d := (d_1 \times d_2), \tag{9b}$$

$$a_1 := 200,\ a_2 := 40,\ d_2 := 0 \,\} \tag{9c}$$

- cf(D_1) determines the algorithm alg(D_1)
- cf(E_1) determines the algorithm alg(E_1)

$$\text{alg}(D_1) \text{ computes } \text{den}(D_1) = \text{den}(40) \text{ (decimal)} \tag{10a}$$

$$\text{alg}(E_1) \text{ computes } \text{den}(E_1) = \mathit{Error} \equiv \mathit{er} \tag{10b}$$

- The constant such that designates a restrictor operator: $R \approx \mathsf{cf}(R)$, r designate parametric, restricted algorithms

$$R \equiv \underbrace{\left(n/d \text{ such that } \{\, (n \in \mathbb{N}),\ (d \in \mathbb{N}),\ (d \neq 0)\,\}\right)}_{\text{restrictor term } R} \tag{11a}$$

$$R_1 \equiv \Big[\underbrace{(a_0 \text{ such that } \{\, z_n, z_d, d_0 \,\})}_{\text{restricted memory variable } r_0} \text{ where } \{ \tag{11b}$$

$$\begin{aligned} &a_0 := n/d,\ z_n := (n \in \mathbb{N}),\ z_d := (d \in \mathbb{N}), \\ &d_0 := \neg p,\ p := (d = 0) \,\}\Big] \end{aligned} \tag{11c}$$

  - r_0, in (11b), and R_1, in (11b)–(11c), are restricted memory variables
  - R_1 instantiates r_0 via parametric (underspecified) assignments (11c)

- $D \in$ Terms instantiates the restrictor R_1 in (11b)–(11c)

$$D \equiv R_1 \text{ where } \{\, n := (a_1 + a_2),\ d := (d_1 \times d_2), \tag{12a}$$

$$a_1 := 200,\ a_2 := 40,\ d_1 := 2,\ d_2 := 3 \,\} \tag{12b}$$

- $\mathsf{cf}(D)$ designates the algorithm $\mathsf{alg}(D)$ for computing the value: $\mathsf{den}(D) = \mathsf{den}(40)$ (e.g., in decimal number system)

- $R_1 \approx \text{cf}(R_1)$ designate the parametric, restricted algorithm $\text{alg}(R_1)$ represented by $\text{cf}(R_1)$

$$R_1 \Rightarrow_{\text{cf}} \Big[\underbrace{(a_0 \text{ such that } \{ z_n, z_d, d_0 \})}_{\text{restricted memory variable } r_0} \text{ where } \{ \tag{13a}$$

$$\begin{aligned} & a_0 := n/d, \\ & z_n := (n \in \mathbb{N}),\ z_d := (d \in \mathbb{N}), \\ & d_0 := \neg p,\ p := (d = n_0),\ n_0 := 0 \,\} \Big] \end{aligned} \tag{13b}$$

- $D \in$ Terms instantiates the memory variables R_1, $\text{cf}(R_1)$, r

$$D \Rightarrow r \text{ where } \{ r := \Big[\underbrace{(a_0 \text{ such that } \{ z_n, z_d, d_0 \})}_{\text{restricted memory variable } r_0} \text{ where } \{ \tag{14a}$$

$$a_0 := n/d, \tag{14b}$$

$$z_n := (n \in \mathbb{N}),\ z_d := (d \in \mathbb{N}), \tag{14c}$$

$$d_0 := \neg p,\ p := (d = n_0),\ n_0 := 0 \,\} \Big], \tag{14d}$$

$$n := (a_1 + a_2),\ d := (d_1 \times d_2), \tag{14e}$$

$$a_1 := 200,\ a_2 := 40,\ d_1 := 2,\ d_2 := 3 \,\} \tag{14f}$$

- $\text{cf}(D)$ designates the algorithm $\text{alg}(D)$ for computing the value: $\text{den}(D) = \text{den}(40)$ (e.g., in decimal number system)

- The (same) parametric restrictor $R \approx \mathrm{cf}(R)$ and the restricted variable R_1 can be instantiated by a variety of algorithms

$$R \equiv \underbrace{\left(n/d \text{ such that } \{\, (n \in \mathbb{N}),\ (d \in \mathbb{N}),\ (d \neq 0) \,\}\right)}_{\text{restrictor term } R} \tag{15a}$$

$$R_1 \Rightarrow_{\mathrm{cf}} \Big[\underbrace{(a_0 \text{ such that } \{\, z_n, z_d, d_0 \,\})}_{\text{restricted memory variable } r_0} \text{ where } \{ \tag{15b}$$

$$a_0 := n/d,\ z_n := (n \in \mathbb{N}),\ z_d := (d \in \mathbb{N}),$$
$$d_0 := \neg p,\ p := (d = n_0),\ n_0 := 0 \,\} \Big] \tag{15c}$$

- E instantiates the restrictor R_1 without satisfying it:

$$E \equiv R_1 \text{ where } \{\, n := (a_1 + a_2),\ d := (d_1 \times d_2), \tag{16a}$$
$$a_1 := 200,\ a_2 := 40,\ d_1 := 2,\ d_2 := 0 \,\} \tag{16b}$$

- $\mathrm{cf}(E)$ determines the algorithm $\mathrm{alg}(E)$ for computing $\mathrm{den}(E) = er$

$$\text{issue: } \mathrm{den}(d_2) = 0,\ \mathrm{den}(d) = [\mathrm{den}(d_1) \times \mathrm{den}(d_2)] = 0 \tag{17a}$$
$$\text{(17a) contradicts the constraints } d_0 := \neg p,\ p := (d = n_0) \tag{17b}$$

- My focus is on:
  - Development of $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ and $\mathrm{L}^{\lambda}_{\mathrm{r}}$
  - Applications to formal and natural languages
    - Computational Semantics
    - Computational Syntax-Semantics Interfaces
    - Semantics of programming and specification languages
    - Theoretical foundations of compilers
- More to come

THANK YOU!

Some References I

Yiannis N Moschovakis.
The formal language of recursion.
Journal of Symbolic Logic, 54(04):1216–1252, 1989.
URL: https://doi.org/10.1017/S0022481200041086.

Yiannis N. Moschovakis.
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.