From finite semantics to regular languages (and beyond) in second-order linear logic

Nguyễn Lê Thành Dũng (a.k.a. Tito) — nltd@nguyentito.eu
Laboratoire d'informatique de Paris Nord (LIPN), Université Paris 13
joint work with Thomas Seiller (CNRS, LIPN)

ELICA project final meeting, Paris, October 11th, 2018

Implicit complexity with proofs-as-programs

Curry-Howard approach to implicit complexity:

1. Define logic / programming language
2. Bound evaluation complexity (*soundness*)
3. Show language expressivity (*extensional completeness*)
4. Result: set of expressible functions = some complexity class

Step 1 requires creativity. Examples from linear logic:
LLL $\rightsquigarrow$ polytime (Girard), SBAL $\rightsquigarrow$ logspace (Schöpp)…

Implicit complexity with proofs-as-programs

Curry-Howard approach to implicit complexity:

1. Define logic / programming language
2. Bound evaluation complexity (*soundness*)
3. Show language expressivity (*extensional completeness*)
4. Result: set of expressible functions = some complexity class

Step 1 requires creativity. Examples from linear logic:
LLL $\rightsquigarrow$ polytime (Girard), SBAL $\rightsquigarrow$ logspace (Schöpp)…

This talk: instead, ask (2)–(4) for well-known systems:

- simply typed λ-calculus (STλ)
    - recall old methods and results
- and later Elementary Linear Logic (ELL)
    - new results inspired by STλ techniques

…or rather, (naturally) restricted situations within STλ/ELL.

Simply-typed λ-calculus and implicit complexity?

Recall k-$\mathsf{EXPTIME} = \mathsf{DTIME}$(tower of exponentials of height k), $\mathsf{ELEMENTARY} = \bigcup_{k \in \mathbb{N}} k$-$\mathsf{EXPTIME}$.

Claim: $\mathsf{ST}\lambda$ characterizes $\mathsf{ELEMENTARY}$.

Parameter controlling complexity: *functionality order*

$$\mathrm{ord}(\alpha \to \beta) = \max(\mathrm{ord}(\alpha) + 1, \mathrm{ord}(\beta))$$

- Soundness: $\forall k \in \mathbb{N}\ \exists f(k) \in \mathbb{N}$ s.t. normalization of λ-terms with order $\leq k$ subterms is in $f(k)$-$\mathsf{EXPTIME}$
- Extensional completeness: **naive attempt fails**

Church encodings of inputs in $\mathrm{ST}\lambda$

Church (or Böhm–Berarducci) encodings:

- For $w \in \{0,1\}^*$, $w : \mathsf{Str}[A]$ for any simple type A (meta-$\forall$)
  - $\mathsf{Str}[A] = (A \to A) \to (A \to A) \to (A \to A)$
  - $\overline{w} = \lambda f_0.\, \lambda f_1.\, \lambda x.\, f_{w[0]}\,(\ldots(f_{w[n-1]}\,x)\ldots)$
- $\mathsf{Bool} = o \to o \to o$ (o base type)

Choose a simple type A, and a term $t : \mathsf{Str}[A] \to \mathsf{Bool}$
$\longrightarrow$ defines language $\mathcal{L}(t) = \{w \in \{0,1\}^* \mid t\,\overline{w} \to_\beta^* \texttt{true}\}$.
Not all $\mathsf{ELEMENTARY}$ languages possible… but then what?

Church encodings of inputs in STλ

Church (or Böhm–Berarducci) encodings:

- For $w \in \{0,1\}^*$, $w : \mathsf{Str}[A]$ for any simple type A (meta-$\forall$)
  - $\mathsf{Str}[A] = (A \to A) \to (A \to A) \to (A \to A)$
  - $\overline{w} = \lambda f_0.\, \lambda f_1.\, \lambda x.\, f_{w[0]}\, (\dots (f_{w[n-1]}\, x) \dots)$
- $\mathsf{Bool} = o \to o \to o$ (o base type)

Choose a simple type A, and a term $t : \mathsf{Str}[A] \to \mathsf{Bool}$
$\longrightarrow$ defines language $\mathcal{L}(t) = \{w \in \{0,1\}^* \mid t\,\overline{w} \to_\beta^* \mathtt{true}\}$.
Not all ELEMENTARY languages possible… but then what?

Theorem (Hillebrand & Kanellakis, LICS'96)

The languages decided by STλ*-terms of type* $\mathsf{Str}[A] \to \mathsf{Bool}$ *are exactly the* regular *languages.*

Regular languages in STλ

Theorem (Hillebrand & Kanellakis, LICS'96)

For any type A and any STλ*-term $t : \mathsf{Str}[A] \to \mathsf{Bool}$, the language $\mathcal{L}(t) = \{w \in \{0,1\}^* \mid t\,\overline{w} \to_\beta^* \mathtt{true}\}$ is* regular*.*

Part 1 of proof.

Fix type A. Any *denotational semantics* $[\![-]\!]$ quotients words:

$$w \in \{0,1\}^* \rightsquigarrow \overline{w} : \mathsf{Str}[A] \rightsquigarrow [\![\overline{w}]\!]_{\mathsf{Str}[A]} \in [\![\mathsf{Str}[A]]\!]$$

When $[\![-]\!]$ *non-trivial* ($[\![\mathtt{true}]\!] \neq [\![\mathtt{false}]\!]$), $[\![\overline{w}]\!]_{\mathsf{Str}[A]}$ determines behavior of w w.r.t. all $\mathsf{Str}[A] \to \mathsf{Bool}$ terms:

$$w \in \mathcal{L}(t) \iff t\,\overline{w} \to_\beta^* \mathtt{true} \iff [\![t\,\overline{w}]\!] = [\![t]\!]([\![\overline{w}]\!]) = [\![\mathtt{true}]\!]$$

Goal: to decide $\mathcal{L}(t)$, compute $w \mapsto [\![\overline{w}]\!]$ in some model of STλ.

Regular languages in STλ

Theorem (Hillebrand & Kanellakis, LICS'96)

For any type A and any STλ*-term* $t : \mathsf{Str}[A] \to \mathsf{Bool}$, *the language* $\mathcal{L}(t) = \{w \in \{0,1\}^* \mid t\,\overline{w} \to_\beta^* \texttt{true}\}$ *is* regular.

Part 2 of proof.

We use $[\![-]\!] : \mathrm{ST}\lambda \to \mathsf{FinSet}$ to build a DFA with states $Q = [\![\mathsf{Str}[A]]\!]$, acceptation as $[\![t]\!](-) = [\![\texttt{true}]\!]$.

$$\begin{tikzcd} {} \arrow[r] & {[\![\overline{\varepsilon}]\!]} \arrow[r, "0"] & {[\![\overline{0}]\!]} \arrow[r, "1"] & {[\![\overline{01}]\!]} \arrow[r, "1"] & {[\![\overline{011}]\!]} \arrow[r] & \cdots \end{tikzcd}$$

$$w \in \mathcal{L}(t) \iff [\![t]\!]\Big([\![\overline{w}]\!]_{\mathsf{Str}[A]}\Big) = [\![\texttt{true}]\!] \iff w \text{ accepted}$$

$\longrightarrow$ *semantic evaluation* argument. $\square$

Regular languages in STλ

Theorem (Hillebrand & Kanellakis, LICS'96)

For any type A and any STλ*-term* $t : \mathsf{Str}[A] \to \mathsf{Bool}$, *the language* $\mathcal{L}(t) = \{w \in \{0,1\}^* \mid t\,\overline{w} \to_\beta^* \texttt{true}\}$ *is* regular.

Part 2 of proof.

We use $[\![-]\!] : \mathrm{ST}\lambda \to \mathsf{FinSet}$ to build a DFA with states $Q = [\![\mathsf{Str}[A]]\!]$, acceptation as $[\![t]\!](-) = [\![\texttt{true}]\!]$. ($|Q| < \infty$, e.g. $2^{2^{33}}$ when $A = \mathsf{Bool}$)

$$\begin{tikzcd} {} \arrow[r] & {[\![\varepsilon]\!]} \arrow[r, "0"] & {[\![0]\!]} \arrow[r, "1"] & {[\![01]\!]} \arrow[r, "1"] & {[\![011]\!]} \arrow[r] & \cdots \end{tikzcd}$$

$$w \in \mathcal{L}(t) \iff [\![t]\!]\left([\![\overline{w}]\!]_{\mathsf{Str}[A]}\right) = [\![\texttt{true}]\!] \iff w \text{ accepted}$$

$\longrightarrow$ *semantic evaluation* argument. $\square$

Moral of the story

Finite denotational semantics have complexity consequences.

- Analogous results for tree automata, and for propositional linear logic (using your favorite finite model)
- Another application to STλ *at fixed order*:

Theorem (Terui, RTA'12)

Normalizing an STλ*-term of type* Bool *w/ order* $\leq r$ *subterms is*

- k-EXPTIME-*complete for* $r = 2k + 2$
- k-EXPSPACE-*complete for* $r = 2k + 3$

Proof of membership in k-EXPTIME / k-EXPSPACE.

β-reduce to halve order, then evaluate in LL Scott model. $\square$

Extensional completeness for STλ

Need to change input representation!

Hillebrand, Kanellakis & Mairson, motivated by database queries, encode *finite relational (1st-order) structures* as inputs $\longrightarrow$ completeness for $\mathsf{ELEMENTARY}$.

- We'll come back to this later
- "β-convertibility of STλ terms $\notin \mathsf{ELEMENTARY}$" (Statman 1979) can be recovered from this
- Refined by H&K: characterization of k-$\mathsf{EXPTIME}$ / k-$\mathsf{EXPSPACE}$ in STλ+constants+equality at fixed order
  - Also using semantic evaluation for soundness!

Regular languages in Elementary Linear Logic

Elementary Linear Logic (ELL)

ELL = Multiplicative-Additive Linear Logic (MALL) + ?/! rules:

$$\frac{\vdash \Gamma, ?A, ?A}{\vdash \Gamma, ?A} \qquad \frac{\vdash \Gamma}{\vdash \Gamma, ?A} \qquad \frac{\vdash A_1, \ldots, A_n, B}{\vdash ?A_1, \ldots, ?A_n, !B}$$

!-intro: promotion and dereliction must come together.
$\longrightarrow$ enforces *stratification* by !-depth
(subproofs cannot change depth during cut-elimination)

Representable functions in 2nd-order ELL = $\mathsf{ELEMENTARY}$:

- Soundness: normalization in $f(\text{depth})$-$\mathsf{EXPTIME}$
  - depth in ELL $\simeq$ order in STλ
- Extensional completeness: Church encoding works
  - thanks to (impredicative) polymorphism!

Complexity in second-order ELL (1)

Data types:

- $\mathsf{Bool} = 1 \oplus 1$
- $\mathsf{Str} = \forall X. !(X \multimap X) \multimap !(X \multimap X) \multimap !(X \multimap X)$

Extensional completeness: all languages $L \in \mathsf{ELEMENTARY}$ expressible by ELL proofs of $!\mathsf{Str} \multimap !^k\mathsf{Bool}$ (k depends on L).

Soundness (reformulated):
proofs of $!^k\mathsf{Bool}$ can be normalized in $f(k)$-$\mathsf{EXPTIME}$.

Question: what do we get for a *fixed* depth k?

Complexity in second-order ELL (2)

Depth $k = 2$ case, in a variant of ELL:

Theorem (Baillot, APLAS'11)

The proofs of $!\mathsf{Str} \multimap !!\mathsf{Bool}$ *in 2nd order elementary affine logic* with recursive types *decide exactly the languages in* P.

Complexity in second-order ELL (2)

Depth $k = 2$ case, in a variant of ELL:

Theorem (Baillot, APLAS'11)

The proofs of $!\mathsf{Str} \multimap !!\mathsf{Bool}$ *in 2nd order elementary affine logic* with recursive types *decide exactly the languages in* P*.*

Recursive types are crucial for the above, as we show:

Theorem

The proofs of $!\mathsf{Str} \multimap !!\mathsf{Bool}$ *in 2nd order* ELL *decide exactly the* regular *languages.*

Proof idea.

Adapt Hillebrand & Kanellakis's STλ proof. Requires non-trivial *finite* semantics for *2nd order* MALL (MALL2). □

Finite semantics for MALL2

Choice of semantics: syntax/(*observational equivalence*).

Definition (Eqv. for propositional observations)

Let A be a MALL2 formula and $\pi, \pi' : A$. Define $\pi \sim_A \pi'$ as:

$$\forall B\,\text{MALL0},\ \forall \rho : (A \vdash B),\ \mathbf{cut}(\pi, \rho) \equiv \mathbf{cut}(\pi', \rho)$$

- MALL0 = propositional MALL
- $\equiv$ is usual proof equivalence on MALL0 (think $=_{\beta\eta}$)

Theorem

For any MALL2 *formula* A*, there are* finitely many *classes for* $\sim_A$.

Corollary

There exists a non-trivial finite semantics for MALL2.

New result of independent interest, cf. Pistone's talk 2 days ago.

Semantic evaluation in ELL, in a nutshell

Let $\pi : !\mathsf{Str} \multimap !!\mathsf{Bool}$. There exists

$$\widehat{\pi} : \mathsf{Str}[A_1] \multimap \ldots \multimap \mathsf{Str}[A_n] \multimap !\mathsf{Bool}$$

such that $\forall w.\ \pi(!\overline{w}) = !\widehat{\pi}(\overline{w}[A_1], \ldots, \overline{w}[A_n])$. $\quad (\mathsf{Str} = \forall X.\, \mathsf{Str}[X])$

Thanks to stratification, w.l.o.g. $A_1, \ldots, A_n \in$ MALL2.
Using finite MALL2 semantics $[\![-]\!]$, $\overline{w}[A]$ induces map

$$\|w\|_A : [\![A \multimap A]\!] \times [\![A \multimap A]\!] \to [\![A \multimap A]\!]$$

such that $\overline{w}[A](!f_1, !f_2) = !g \implies \|w\|_A([\![f_1]\!], [\![f_2]\!]) = [\![g]\!]$.

- Church encoding $\longrightarrow \|w\|_A$ computable by automaton
- $(\|w\|_{A_1}, \ldots, \|w\|_{A_n})$ determine $\widehat{\pi}(\overline{w}[A_1], \ldots, \overline{w}[A_n])$ and therefore $\pi(!\overline{w})$

What about higher depths?

We solved depth $k = 2$ case for ELL.
(First characterization of regular languages in a type system with impredicative quantification?)

When $k > 2$:

Theorem (Baillot, APLAS'11)

The proofs of $!\mathsf{Str} \multimap !^k\mathsf{Bool}$ *in* EAL+rectypes *decide exactly the languages in* $(k-2)$-$\mathsf{EXPTIME}$.

- For ELL without recursive types, we get a class between $(k-3)$-$\mathsf{EXPTIME}$ and $(k-2)$-$\mathsf{EXPTIME}$… which one exactly?
- Semantics probably has a role to play in the answer

Inputs as finite models:
towards logarithmic space in ELL?

Data represented as (totally ordered) *finite first-order structures* (a.k.a. *finite models*), over a signature of relation symbols.

Example

Signature for binary strings: $\langle \leq, S \rangle$.
Finite models are $(D, \leq^D, S^D)$, $|D| < \infty$. $S^D(d)$ = "d^{th} bit is 1".

Descriptive complexity: characterize a complexity class $\mathcal{C}$ as set of *queries* written in some logic $L_{\mathcal{C}}$, i.e. "is this $L_{\mathcal{C}}$ formula true in this finite model?". For instance:

Theorem (Fagin 1974)

Queries in existential second-order logic = NP.

Finite models in STλ and extensional completeness

With type d of elements (equipped with $\mathsf{Eq} : d \to d \to \mathsf{Bool}$),

- Represent k-ary relations as lists of k-tuples
  $\mathsf{Rel}_k[d, A] = (d^k \to A \to A) \to A \to A$
  (in the spirit of database theory: relation = set of records)
- Provide list of all domain elements ($\mathsf{List}[d, A] = \mathsf{Rel}_1[d, A]$)

Theorem (Hillebrand, Kanellakis & Mairson, LICS'93)

Terms $t : \mathsf{List}[d, A] \to \mathsf{Rel}_{k_1}[d, A_1] \to \ldots \to \mathsf{Rel}_{k_m}[d, A_m] \to \mathsf{Bool}$ *in* STλ *compute exactly* ELEMENTARY *queries over finite models.*

To feed input, instantiate $d = o^n \to o$ (n = domain size).
Program has "$\forall d\, \exists A$", input has "$\exists d\, \forall A$". Size of semantics depends on input, breaking earlier expressivity upper bound.

Finite models in ELL

We transpose this idea to second-order ELL:

- We use $\mathsf{Rel}_k[D] = D^{\otimes k} \multimap \mathsf{Bool}$, $\mathsf{List}[D] = \forall X.\, !(D \multimap X \multimap X) \multimap !(X \multimap X)$
- Allow non-linear use of D: $\mathsf{Cont}[D] = D \multimap D \otimes D$, $\mathsf{Wk}[D] = D \multimap 1$

$$\mathsf{Inp}_r = \exists D.\, !\mathsf{List}[D] \otimes \bigotimes_{i=1}^{n} !^r \mathsf{Rel}_{k_i}[D] \otimes !^r \mathsf{Cont}[D] \otimes !^r \mathsf{Wk}[D]$$

(choose r to satisfy stratification constraint)

- For size n domain, witness $D = 1 \oplus \ldots (n \text{ times}) \ldots \oplus 1$ (positive, therefore duplicable)

Towards logarithmic space in ELL? (1)

Theorem (Immerman 1983)

Queries in first-order logic with deterministic transitive closure = logarithmic space (L) *queries.*

Proposition

All L *queries on finite models for a given signature can be computed by an* ELL *proof of* $\mathsf{Inp}_2 \multimap !!\mathsf{Bool}$.

Proof idea: compute transitive closure of a relation $\mathcal{R} \subseteq D^k \times D^k$ by iterating $\varphi_{\mathcal{R}} : \mathcal{P}(D^k \times D^k) \to \mathcal{P}(D^k \times D^k)$.

Determinism of $\mathcal{R}$ ensures *linearity*: $\varphi_{\mathcal{R}} : \mathsf{Rel}_{2k} \multimap \mathsf{Rel}_{2k}$ in ELL.

This is remarkable enough to hope for:

Conjecture

Conversely, proofs of $\mathsf{Inp}_2 \multimap !!\mathsf{Bool}$ *only decide* L *queries.*

Towards logarithmic space in ELL? (2)

Conjecture

ELL *proofs of the following type only decide* L *queries:*

$$\left(\exists D.\,!\mathsf{List}[D] \otimes \bigotimes_{i=1}^{n} !!\mathsf{Rel}_{k_i}[D] \otimes !!\mathsf{Cont}[D] \otimes !!\mathsf{Wk}[D]\right) \multimap !!\mathsf{Bool}$$

In predicative case ($\forall/\exists$ range over propositional formulae):

- conjecture seems very likely
- already non-trivial… (maybe Geometry of Interaction works?)
- note that ext. completeness holds w/o impredicativity

In general case, I have no intuition or methods available ☹

Conclusion and future work

Conclusion

We brought methods from the STλ tradition to 2nd order ELL, showing that similar phenomena occur in both:

- *Church encodings* of inputs restrict expressivity
- *Semantic evaluation* can prove this (and lots of other stuff)
- To overcome this, one can represent inputs as *finite models*

Lemma (or Theorem, if you care about semantics)

The quotient of MALL2 *by propositional observations is* finite.

Theorem (or Corollary)

Proofs of $!\mathsf{Str} \multimap\, !!\mathsf{Bool}$ *in* ELL *decide* regular *languages*.

Moral: *geometry* (e.g. stratification) and *typing* jointly control complexity; semantics reflects the latter.

Open problems / future work

Logspace conjecture: what kind of techniques can solve this??

Classes characterized by higher fixed depths?
(For both $!\mathsf{Str} \multimap !^k\mathsf{Bool}$ and $\mathsf{Inp}_k \multimap !^k\mathsf{Bool}$…)

Related: complexity of normalizing a proof of $!^k\mathsf{Bool}$ in ELL?

- $k = 0$: P-complete
- $k = 1$: PSPACE-hard, in $\mathsf{EXPTIME}$
- $k \geq 2$: $(k-1)$-$\mathsf{EXPTIME}$-hard, in k-$\mathsf{EXPTIME}$

(For EAL+rectypes, k-$\mathsf{EXPTIME}$-complete by Baillot's results.)

On MALL2 semantics: further investigations ongoing,
j.w.w. P. Pistone and L. Tortora de Falco.