Logic Programming and Logarithmic Space

Clément Aubert[←], Marc Bagnol[←],
Paolo Pistone[←], Thomas Seiller[⇋]

[←] *Institut de Mathématiques de Marseille*
[⇋] *Institut des Hautes Études Scientifiques*

November 19, 2014

Inspiration: Proof Theory, GoI and Implicit Complexity

Subsystems of LL that capture complexity classes.

GoI is a semantics of cut-elimination, allows to study it abstractly.

$\rightarrow$ What can GoI say about ICC?

Resolution-based Geometry of Interaction

Resolution-based GoI: more syntactical flavour, better suited for complexity analysis, related to logic programming.

Within the *"resolution semiring"* used to build the GoI model, find a suitable *"semiring of logspace"*.

Related work

- Baillot, Pedicini: *Elementary complexity and geometry of interaction* (2001)
- Girard: *Normativity in Logic* (2010)
- Aubert, Seiller: *Characterizing coNL by a Group Action* (2012), *Logarithmic Space and Permutations* (2013)
- Aubert, Bagnol: *Unification and logarithmic space* (2014)

The Resolution Semiring

- An algebraic view of logic programs
- Enables vocabulary and tools from abstract algebra

Flows

Flow: a pair $t \leftharpoondown u$ of (first-order) terms with $\mathtt{var}(t) \subseteq \mathtt{var}(u)$.

(considered up to renaming of variables)

Think of $t \leftharpoondown u$ as ‘`match ... with u -> t`’ in a ML-style language, or as a (safe) clause $t \dashv u$ in logic programming.

Product:

$$(u \leftharpoondown v)(t \leftharpoondown w) := u\theta \leftharpoondown w\theta$$

where $\theta = MGU(v, t)$, may be undefined. (*resolution rule* of LP)

Examples: *($\bullet$ is a binary symbol written in infix notation)*

$\big(\mathtt{g}(x) \leftharpoondown \mathtt{f}(x)\big)\big(y \leftharpoondown \mathtt{g}(y)\big) = \mathtt{g}(x) \leftharpoondown g(\mathtt{f}(x))$

$\big(\mathtt{g}(x) \leftharpoondown x \bullet \mathtt{c}\big)\big(y \bullet y \leftharpoondown \mathtt{f}(y)\big) = \mathtt{g}(\mathtt{c}) \leftharpoondown \mathtt{f}(\mathtt{c})$

Wires

Wires: sets of flows. (*i.e.* logic programs)

The set of wires has a structure of **semiring**:

$$L = \{l_1, \ldots, l_n\} = l_1 + \cdots + l_n = \sum_i l_i$$

$$L + K = L \cup K \qquad \textit{(sum)}$$

$$0 = \varnothing \qquad \textit{(neutral for } +\textit{)}$$

$$L\,K := \sum_{\substack{l \in L,\, k \in K \\ lk \text{ defined}}} lk \qquad \textit{(product)}$$

$$\mathbf{I} := x \leftharpoonup x \qquad \textit{(neutral for product)}$$

We write $\mathcal{R}$ the set of wires, the **resolution semiring**.

Somewhere to start

Unification is **Ptime**-complete.

Theorem (Dwork, Kellenakis, Mitchell – 1984)

*The matching problem (unifying two terms when one of the terms has no variable) is in **DLogspace**.*

And therefore the product FG can be computed in logspace if either F or G contains only closed flows.

Words and Observations

- Representing inputs as wires
- Accepting/rejecting

Words

The encoding of words in $\mathcal{R}$ comes from the Church encoding of words in LL/λ-calculus and their GoI representation.

Another intuition: transitions of an automaton

$$\text{configuration term:} \quad \mathtt{c}\bullet\mathtt{L}/\mathtt{R}\bullet s\bullet m\bullet\mathtt{H}(\mathtt{p})$$

- $\mathtt{c}$ is the symbol under the reading head.
- $\mathtt{L}/\mathtt{R}$ is the direction of the next move of the head.
- s is the internal state of the automaton.
- m is the memory of the automaton (pointers, for instance).
- $\mathtt{H}(\mathtt{p})$ is the position of the head.

The action of the encoding can be understood as *moving the head*.

Words

Formal definition: if $W = \mathtt{c}_1 \dots \mathtt{c}_n$ is a word of length n and $\mathtt{p}_0, \mathtt{p}_1, \dots, \mathtt{p}_n \in \mathbf{P}$ distinct (*position*) constants:

$$
\begin{aligned}
W[\mathtt{p}_0, \mathtt{p}_1, \dots, \mathtt{p}_n] := \quad & \star \bullet \mathtt{R} \bullet x \bullet y \bullet \mathtt{H}(\mathtt{p}_0) \leftrightharpoons \mathtt{c}_1 \bullet \mathtt{L} \bullet x \bullet y \bullet \mathtt{H}(\mathtt{p}_2) \\
+ \quad & \mathtt{c}_1 \bullet \mathtt{R} \bullet x \bullet y \bullet \mathtt{H}(\mathtt{p}_1) \leftrightharpoons \mathtt{c}_2 \bullet \mathtt{L} \bullet x \bullet y \bullet \mathtt{H}(\mathtt{p}_2) \\
+ \quad & \cdots \\
+ \quad & \mathtt{c}_n \bullet \mathtt{R} \bullet x \bullet y \bullet \mathtt{H}(\mathtt{p}_n) \leftrightharpoons \star \bullet \mathtt{L} \bullet x \bullet y \bullet \mathtt{H}(\mathtt{p}_0)
\end{aligned}
$$

Well-suited for log-space computation: interactive, Q/A model. Configurations can be stored within logarithmic space.

Observations

Observations are elements of a fixed semiring $\mathcal{A}$, and cannot use the position constants.

An observation ϕ **accepts** a representation $W[\mathrm{p}_0, \ldots, \mathrm{p}_n]$ if

$$(\phi W[\mathrm{p}_0, \ldots, \mathrm{p}_n])^k = 0 \text{ for some } k \textbf{ (nilpotency)}$$

Corresponds to termination (strong normalization) of computation in GoI, and *boundedness* in LP.

Potential issue: different representations of the same word.

Theorem (Normativity)

Let ϕ be an observation, W a word. If $\phi W[\mathrm{p}_0, \ldots, \mathrm{p}_n]$ is nilpotent for one choice of $\mathrm{p}_0, \ldots, \mathrm{p}_n$, then it is for all choices.

We define, for any observation ϕ,

$$\mathbf{L}(\phi) := \{\ W \text{ word} \mid \phi\, W[\mathrm{p}_i] \text{ nilpotent for any choice of } [\mathrm{p}_i]\ \}$$

The Balanced Semiring

- Logspace and the height of variables

Balanced Flows

We seek a semiring with a nilpotency problem space-efficiently tractable.

Balance: $t \leftharpoondown u$ is balanced if for any variable x, all occurences of x in t and u have the same height (distance from the root).

Intuitively, this forbids to stack symbols on top of a variable to store information.

Examples:

$\mathtt{f}(x) \leftharpoondown x$	not balanced
$\mathtt{g}(x \bullet x) \leftharpoondown \mathtt{f}(x \bullet \mathtt{g}(y))$	balanced

Preserved by sum and product: subsemiring of balanced wires.

Deciding nilpotency

Balanced flows behave well *w.r.t.* height of terms.

Given a balanced F, we can tell its nilpotency by observing its behaviour on closed terms of height $\mathbf{h}(F)$.

Basic idea: build a graph $\mathbf{G}(F)$

- vertices are closed terms of height at most $\mathbf{h}(F)$, using only the symbols in F
- edge from $\mathbf{u}$ to $\mathbf{v}$ when $\mathbf{v} \in F(\mathbf{u})$

Theorem

If F is balanced, $\mathbf{G}(F)$ is acyclic iff. F is nilpotent.

This reduces nilpotency to cycle search in a directed graph.

Logarithmic space

- Soundness: *via* the graph above
- Completeness: encoding of pointer machines

Space soundness

We consider **balanced observations**.

Moreover, we say an observation ϕ is **deterministic** if $\mathtt{card}\big(\phi(\mathbf{t})\big) \leq 1$ for any closed $\mathbf{t}$.

Theorem

- *If ϕ is a balanced observation, $\mathbf{L}(\phi)$ is in **co-NLogspace**.*
- *If moreover ϕ is deterministic, $\mathbf{L}(\phi)$ is in **DLogspace**.*

Proof. First show that $\mathbf{G}\big(\phi W[\mathrm{p}_i]\big)$ can be generated in logarithmic space.Then use the fact that the acyclicity problem for directed graphs is solvable in logarithmic space.

Space completeness

Conversely we have:

Theorem

- *If L is in* ***co-NLogspace*** *then there is a balanced observation ϕ such that $\mathbf{L}(\phi) = L$.*
- *If L is in* ***DLogspace*** *then there is a* deterministic *balanced observation ϕ such that $\mathbf{L}(\phi) = L$.*

Proof. By an encoding of *pointer machines*: automata with a reading head and a fixed number of auxiliairy pointers, a standard (qualitative) characterization of logarithmic space computation.

Elements of encoding

Representing pointer manipulation with balanced terms: the configurations are

$$\mathtt{c}\bullet\mathtt{L}/\mathtt{R}\bullet s\bullet\mathtt{A}(\mathtt{p}_{i_1},\ldots,\mathtt{p}_{i_k})\bullet\mathtt{H}(\mathtt{p})$$

where $\mathtt{A}(\mathtt{p}_{i_1},\ldots,\mathtt{p}_{i_k})$ represents the stored positions of k auxiliairy pointers.

For instance:

$$\cdots\bullet A(x,\ldots,x)\bullet\mathtt{H}(x) \leftharpoondown \cdots\bullet A(y_1,\ldots,y_n)\bullet\mathtt{H}(x)$$

Encodes the operation "*move all the pointers to the position of the reading head*".

Work in progress...

Work in Progress

- **Logspace** predicates *vs.* **Logspace** functions.
- Find other semirings that correspond to other complexity classes.
  Possible method: consider a light logic capturing the complexity class **C**, look at the GoI translation, try to guess the corresponding "**C** semiring".
- Compare/relate with other work on the complexity of logic programming.

Thank you.