On the Resolution Semiring

Soutenance de Thèse

Marc Bagnol

Institut de Mathématiques de Marseille

4 décembre 2014

Introduction

- *Proof theory, sequent calculus*
- GoI *and the resolution algebra*

Sequent Calculus

Proof-theory is the branch of logic concerned with the study of proofs (rather than propositions) as a fundamental object.

In this perspective, the tools for describing proofs are important.

A major milestone is the introduction of *sequent calculus* by Gentzen in his work on consistency of arithmetic.

$$H_1, \dots, H_n \vdash C_1, \dots, C_m$$

"Under the hypothesis H_i, one of the C_j holds."

The *rules* of logic are written as

$$\frac{\mathbf{P}_1 \quad \cdots \quad \mathbf{P}_n}{\mathbf{C}}\ \texttt{R}$$

where $\mathbf{P}_i$ and $\mathbf{C}$ are sequents.

A *prooftree* is a tree with nodes labeled by such rules.

Cut-elimination and GoI

Among rules, the `cut`-rule

$$\frac{A \vdash B \qquad B \vdash C}{A \vdash C}\ \texttt{cut}$$

plays a specific role, enabling deductive reasoning (from $A \Rightarrow B$ and $B \Rightarrow C$, deduce $A \Rightarrow C$), *composition* of proofs.

A key result by Gentzen: `cut`-elimination.

Theorem

A proof π can be rewritten into a `cut`-free proof π' with the same conclusion.

Explicitation procedure, sheds an operational light on logic.

The GoI research program, stemming from the theory of *proofnets*: tools to study this procedure abstractly.

Focus on *interactive* & *dynamic* aspects of logic.

The resolution Algebra

The first step was a model of MLL (the very basic and primitive core of linear logic) in terms of finite permutations.

Not enough to account for the potential infinity at work in the full `cut`-elimination procedure. *(structural rules, contraction. . .)*

An algebra/semiring based on the resolution rule: a finite syntax that can represent infinite sets.

Outline

- Presentation of the resolution semiring
- GoI construction, an interpretation of *λ-calculus*
- Implicit complexity

The Resolution Semiring

- *A semiring with a product based on the resolution rule*
- *An algebraic view of logic programs*
- *Vocabulary and tools from abstract algebra*

Unification

Does the equation $t =_? u$ have a formal solution?

We consider (first-order) terms $t, u, v, \ldots$ built using function symbols $\mathtt{c}, \mathtt{f}(\cdot), \mathtt{g}(\cdot,\cdot), \ldots$ and variables $x, y, z, \ldots$

The equation $t =_? u$ has a *unifier* if there is a substitution θ such that $\theta t = \theta u$.

In that case, there is a most general unifier (MGU) ψ such that any other unifier is an instance of ψ.

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

$\mathtt{f}(x) =_? \mathtt{f}(\mathtt{g}(y))$ $\quad \{\, x \mapsto \mathtt{g}(y) \,\}$

$x \bullet \mathtt{c} =_? y \bullet x$ $\quad \{\, x \mapsto \mathtt{c}\,,\, y \mapsto \mathtt{c} \,\}$

$\mathtt{g}(x) =_? \mathtt{f}(\mathtt{c})$ $\quad$ no solution

The unification problem is PTIME**-complete, with subcases in** LOGSPACE**.**

Flows

Flow: a pair $t \leftharpoonup u$ of terms with $\mathtt{var}(t) \subseteq \mathtt{var}(u)$.

(considered up to renaming of variables)

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

Product:

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

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

Examples:

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

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

Wirings

Wiring: a set of flows. (*i.e.* logic programs)

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

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

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

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

$$(l_1 + \dots + l_n)\,(k_1 + \dots + k_m) := \sum_{l_i k_j \text{ defined}} l_i k_j \qquad \textit{(product)}$$

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

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

Geometry of Interaction

- *Interpretation of λ-calculus in $\mathcal{R}$*
- *Undecidablilty of nilpotency*

GoI Situations

Original GoI models: direct definitions and proofs.

Axiomatization led to the notion of GoI situation. Rather than prove everything from scratch, validate the axioms.

A traced category $\mathfrak{R}$ with a functor $\mathbf{!}$ and retractions (embeddings).

eg. Embed ternary $\mathbf{u}_2$ into binary $\mathbf{u}_1$

$$\mathbf{u}_1(x, y \bullet z) \leftharpoonup \mathbf{u}_2(x, y, z)$$

using $\bullet$. *(fundamental to interpret the digging rule)*

GoI **situations automatically yield an interactive (game-like) interpretation of MELL/** λ-***calculus*.**

An Undecidability Theorem

In the interpretation, we have that a λ-term t is strongly normalizing *iff.* some associated wiring $EX[t]$ is *nilpotent*.

Definition

A wiring F is nilpotent *iff.* $F^n = 0$ for some n.

We derive from this observation an undecidability theorem:

Theorem

The nilpotency problem is undecidable.

We will use nilpotency as an acceptance condition, therefore need to restrict it for specific complexity classes.

Complexity

- *Representation of inputs*
- *Normativity*
- *Characterisations of* Logspace *and* Ptime

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

$$\textit{configuration term:} \quad \mathtt{c} \bullet \mathtt{l}/\mathtt{r} \bullet s \bullet m \bullet \mathtt{HEAD}(\mathtt{p})$$

- $\mathtt{c}$ 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).
- $\mathtt{HEAD}(\mathtt{p})$ 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:

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

Well-suited for LOGSPACE computation: GoI, interactive computation, configurations can be stored within logarithmic space.

Observations and Normativity

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

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

$$\phi W[\mathtt{p}_0, \ldots, \mathtt{p}_n] \textit{ is nilpotent}$$

Theorem (Normativity)

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

We define, for any observation ϕ,

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

The balanced semiring

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.

Examples:

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

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

Nilpotency can be decided by a *simulation* technique: instead of computing F^n, we build a graph $\mathbf{G}(F)$ such that F **is nilpotent *iff*.** $\mathbf{G}(F)$ **is acyclic.**

(cycle search in a graph is a Logspace *problem)*

Balanced Observations and LOGSPACE

We consider **balanced** observations.

Theorem

Languages recognized by balanced observations correspond to coNLOGSPACE languages.

Moreover we can isolate a subclass of balanced observation that recognize DLOGSPACE languages.

Proof. Soundness by the simulation technique evoked above.

Completeness by encoding pointer machines.

The $\mathcal{S}tack$ semiring

Flows built using only unary function symbols:

$$\mathtt{f}(\mathtt{f}(x)) \leftharpoonup \mathtt{g}(x)\ ,\ x \leftharpoonup \mathtt{g}(x) \dots$$

Intuition: manipulating stack of function symbols.

These are the flows that arise when interpreting MLL.

The cut-elimination problem for MLL is Ptime*-complete*

Algebraic properties: we say a flow l is a *cycle* if $l^2 \neq 0$ and a wiring F is *cyclic* if F^n contains a cycle for some n.

Lemma

$F \in \mathcal{S}tack$ **is cyclic *iff.* it is not nilpotent.**

Not valid in general: with $l = \mathtt{c} \bullet x \leftharpoonup x \bullet \mathtt{d}$, we have $l^2 = \mathtt{c} \bullet \mathtt{c} \leftharpoonup \mathtt{d} \bullet \mathtt{d}$ but $l^3 = 0$.

Observations with stack and Ptime

Observation with stack: sum of flows of the form $t \bullet u \leftharpoonup v \bullet w$ with

- $t \leftharpoonup v$ is balanced
- $u \leftharpoonup w \in \mathcal{S}tack$
- no shared variables

Adding a stack to pointer machines, inpired by a theorem by S. Cook "automata with pointers and a stack correspond to polynomial time".

Characterization of Ptime:

Theorem

Languages recognized by observations with stack correspond to Ptime languages.

Proof. Completeness by encoding Cook's automata.

Soundness by a polynomial decision procedure derived from the algebraic properties of $\mathcal{S}tack$. *Again, simulation. The nilpotency index can be exponential.*

Perspectives

- *Light logics*
- *Relating recent proof theory and logic programming*
- *Consider a wider class of flows (J.-Y Girard's* stars, *relax safety)*
- *Extend implicit complexity results (*PSPACE, NC*)*
- *Decision problems* vs. *functions*
- *Complexity and abstract algebra*

...THANK YOU FOR YOUR ATTENTION