Towards Algebraic Methods in Descriptive Complexity

Eugenia Ternovska

Simon Fraser University

FLOC, VardiFest, August 1, 2022

Open Problem: A Logic for P-time

Is there a logic that exactly characterizes deterministic polynomial-time (P-time) computability on (unordered) relational structures?

Posed by [Chandra,Harel'1982], further developed by Gurevich

Formally, a **logic** consists of a decidable set of sentences and an isomorphism-closed relation $\models$ between structures and sentences

A **logic captures P-time** if

(1) it can define all P-time-decidable Boolean queries and

(2) there is an algorithm that, for each sentence, constructs a P-time TM deciding the corresponding query

Current Status

The seminal work [**Vardi:1982**, Immerman:1986] a partial solution:

$$\text{P-time} \;=\; \text{FO(FP) on } \textbf{ordered} \text{ structures}$$

The general problem, for **arbitrary** structures, **remains open**

Choiceless Polynomial Time [Blass,Gurevich,Shelah:1999]
a "logic" based on a machine model, is still perhaps the main candidate for capturing P-time

In the related area of Constraint Satisfaction Problem (CSP),
a much more significant progress has been made

The research area was initiated by two papers **[Feder,Vardi'93,98]**
This work received the Alonzo Church Award in 2018

CSP is identified with the **Homomorphism Problem:**

Given: two relational structures $\mathfrak{A}$ and $\mathfrak{B}$ Question: is there a homomorphism $\mathbf{h} : \mathfrak{A} \to \mathfrak{B}$?

$\mathfrak{B}$ is called a template

Non-Uniform CSP: the template $\mathfrak{B}$ is fixed

[Feder,Vardi'93] conjectured a dichotomy:

Non-Uniform CSP is either in P-time or NP-complete

[Bulatov:2017, Zhuk:2017] closed the conjecture positively

Importantly,

the CSP development relied on the techniques of **Universal Algebra**

None of the current approaches to Descriptive complexity (to our knowledge) allow one to take advantage of such techniques

Goal: develop an **algebraic view on the problem**

Intuition for the algebra:

iterated applications of unary conjunctive queries

- augmented with a Choice operator
- controlled by algebraic operations

a bit similar to CSP's k-consistency algorithm that repeatedly computes Primitive-Positive-definable relations

- Start from FO(LFP), the logic used in the Immerman-Vardi theorem
- Inspired by bounded-variable fragments **[Vardi:1995]**, **partition variables** of atomic symbols into inputs and outputs
- Produce an **algebra of binary relations** on strings of structures over the same relational vocabulary

$$\alpha ::= \mathrm{id} \ \mid \ \underbrace{q(\underline{\bar{X}}, Y)}_{\text{unary CQs}} \ \overbrace{\mid \curvearrowright \alpha \mid \curvearrowleft \alpha \mid \alpha \,;\, \alpha \mid \alpha \sqcup \alpha \mid}^{\text{restrictions of } \neg, \wedge, \vee} X =_\alpha Y \mid \alpha^\uparrow$$

($=_\alpha$ within the scope of $\curvearrowright$, $\curvearrowleft$)

A unary CQ returns a **set**

Add a history-dependent **Choice** operator to **pick one element**

E.g.:

$$Reach'(y) \quad :- \quad \underbrace{Reach(x), \mathbf{E}(x,y)}_{\text{CQ}}$$

use free Choice function variable ε (at most one per expression)

$$\alpha ::= \mathrm{id} \mid \underbrace{q[\varepsilon](\underline{\bar{X}}, \bar{Y})}_{\text{CQ with Choice}} \mid \curvearrowright \alpha \mid \curvearrowleft \alpha \mid \alpha \,;\, \alpha \mid \alpha \sqcup \alpha \mid \alpha^{\uparrow} \mid X =_{\alpha} Y$$

The Algebra is Equivalent to a Linear-Time Dynamic Logic

$$\alpha ::= \mathrm{id} \ \mid q[\varepsilon](\underline{\bar{X}}, Y) \mid \curvearrowright \alpha \mid \curvearrowleft \alpha \mid \alpha \,;\, \alpha \mid \alpha \sqcup \alpha \mid X =_\alpha Y \mid \alpha^\uparrow \mid \varphi?$$
$$\varphi ::= \mathbf{T} \mid \neg\varphi \mid \varphi \wedge \varphi \mid |\alpha\rangle\varphi \mid \langle\alpha|\varphi$$

$$\varphi? := \curvearrowright\curvearrowright \varphi \qquad = \mathrm{Dom}(\varphi) \qquad \text{(test action)}$$
$$|\alpha\rangle\ \varphi := \mathrm{Dom}(\alpha\,;\varphi)$$

Programming constructs are **definable**

$$\textbf{if } \varphi \textbf{ then } \alpha \textbf{ else } \beta \ := \ (\varphi?\,;\alpha) \sqcup \beta$$
$$\textbf{while } \varphi \textbf{ do } \alpha := (\varphi?\,;\alpha)^\uparrow\,;(\curvearrowright\varphi?)$$
$$\textbf{repeat } \alpha \textbf{ until } \varphi := \alpha\,;((\curvearrowright\varphi?)\,;\alpha)^\uparrow\,;\varphi?$$

$$\underbrace{\mathfrak{A} \ \models_T \ |\alpha\rangle \mathbf{T} \ [h/\varepsilon] \text{ for some } h}_{\text{there is a successful execution of } \alpha \text{ at } \mathfrak{A}}$$

A concrete Choice function h **witnesses** a Yes answer to this query

The set of **all** such Choice functions a set of certificates for the membership in the computational problem specified by α

E.g.: Cardinality, Reachability, CFI-types of examples

Such sets of certificates is the main object of our study

Analyze data complexity **[Vardi:1982]**

Theorem: The logic captures NP

There can be exponentially many Choice functions (certificates)

Main Goal: syntactic conditions for **equivariance**
(so that a Yes/No answer for one would give us the answer for all)

would give us a decidable syntax for a logic for P-time

The conditions are derived from the analysis of the **automorphism and homomorphism structure** of the sets of certificates
(category and group theory, universal algebra)

Open problem: automata for language containment $\mathcal{L}(\alpha) \subseteq \mathcal{L}(\beta)$

Thank you, Moshe Vardi, for so many inspirations for my work!

Extra Slide: Semantics of Operations (Informally)

$$\alpha ::= \mathrm{id} \ \mid q[\varepsilon](\underline{\bar{X}}, Y) \mid \curvearrowright \alpha \mid \curvearrowleft \alpha \mid \alpha \,;\, \alpha \mid \alpha \sqcup \alpha \mid X =_\alpha Y \mid \alpha^{\uparrow}$$

Forward Unary Negation (Anti-Domain): $\curvearrowright \alpha$ – there is no outgoing α-transition ($\curvearrowleft \alpha$ is similar) – subset of identity id

Composition: $\alpha \,;\, \beta$ – execute sequentially

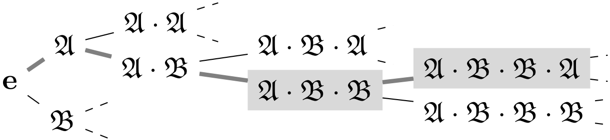
Preferential Union: $\alpha \sqcup \beta$ – perform α if it's defined, o.w. perform β

Comparison for Equality: $X =_\alpha Y$ – compare the "content" of X and Y before and after α (within the scope of $\curvearrowright$, $\curvearrowleft$)

Maximum Iterate: $\alpha^{\uparrow}$ – output the longest transition of α^*

Extra Slide: Choice Functions

A Concrete Choice function h maps, e.g., $\mathfrak{A} \cdot \mathfrak{B} \cdot \mathfrak{B} \mapsto \mathfrak{A} \cdot \mathfrak{B} \cdot \mathfrak{B} \cdot \mathfrak{A}$



At each transition, h resolves atomic non-determinism of a unary CQ