Indexed complexity classes

Siddharth Bhaskar

Department of Computer Science, James Madison University

Grounded programming languages

- A *programming language* consists of a set P of programs, D of data, and a ternary relation $[\![p]\!](x) = y$.
  - (a.k.a. $\varphi_p(x) = y$)
- It is *grounded* if $P \subseteq D$.
- Its first-order $[\![\,]\!]$-theory captures some coarse structural information about it.

$$(\forall p, q \in P)(\exists c \in P)(\forall x \in D)\ [\![c]\!](x) = [\![p]\!]([\![q]\!](x))$$

Grounded programming languages

- A *programming language* consists of a set P of programs, D of data, and a ternary relation $[\![p]\!](x) = y$.
  - (a.k.a. $\varphi_p(x) = y$)
- It is *grounded* if $P \subseteq D$.
- Its first-order $[\![]\!]$-theory captures some coarse structural information about it.

$$(\forall p, q \in P)(\exists c \in P)(\forall x \in D)\ [\![c]\!](x) = [\![p]\!]([\![q]\!](x))$$

Classical recursion theory

- The *classical case* is when we consider a non-pathological Turing-complete programming language.
- This yields an *acceptable* indexing of the partial recursive functions.
- Any two such languages have an identical first-order $[\![\,]\!]$-theory.
- Upshot: in the classical case, structure does not meet power.

Complexity classes

- Consider a *non*-Turing complete programming language that captures C a complexity class.
- This yields an indexing of C.
- Here, we have the possibility of different indexings with different structural properties.

Kozen's axioms

- Getting off the ground: any analogue of acceptability?
- Yes (Kozen 1978):
  - Constant functionals:
    $(\forall x \in D)(\exists p \in P)(\forall y \in D)\ [\![[\![p]\!]]\!](y) = x.$
  - Sequential composition:
    $(\forall p, q \in P)(\exists c \in P)(\forall x \in D)\ [\![c]\!](x) = [\![p]\!]([\![q]\!](x)).$
  - Parallel composition:
    $(\forall p, q \in P)(\exists c \in P)(\forall x \in D)\ [\![c]\!](x) = ([\![p]\!](x), [\![q]\!](x)).$
- Nontrivial consequences, e.g.,
  - Kleene's second recursion theorem
  - abstract Rice's theorem

Motivating questions

- A provocative question: *Do complexity classes obey conservation of structure?*
- In other words: to be sufficiently expressive, do you have to be sufficiently structurally complex?
- Kozen: YES, in the special case of clocked indexings of PTIME.

Motivating questions

- Other questions:
- How many indexings of a given complexity class are there, up to computable isomorphism?
- When can one programming language be efficiently compiled into another?

Why now?

- Study of subrecursive indexings: $\leq 1980's$
  - mostly "large" classes
- Implicit computational complexity: $\geq 1990's$
- New examples from the perspective of subrecursive indexings.
- New questions from the perspective of ICC.

Timed indexed classes

- We can further extend the $[\![\,]\!]$-theory of a programming language by the additional relation

$$\langle\!\langle p \rangle\!\rangle(d) < n,$$

  meaning "program p halts within time n on input d."
- This is basically the **Blum relation** $\Phi_p(d) < n$.
- The resulting theory encodes even more intensional information.

Timed indexed classes

- Classical case: extend acceptable indexings φ_e of partial recursive functions by the additional relation Φ_e plus *Blum's axioms*
- Significant consequences: Rabin's theorem, upward and downward gap theorems, **speedup theorem**
- Is there a subrecursive analogue?
  - YES (Alton, 1980), a *self-simulating indexing*.

Open questions

- Which complexity classes admit a self-simulating indexing?
- Can such indexings be used to exhibit speedup phenomena?