A Rewriting Characterization of Higher-Order Feasibility via Tuple Interpretations

Ongoing joint work with Patrick Baillot, Ugo dal Lago,
Cynthia Kop, and **Deivid Vale**
June 28, 2023

iCIS | Software Science
Radboud University

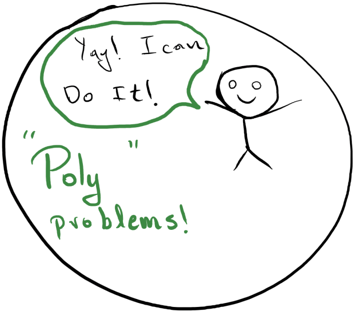
Outline

Poly-time in a nutshell

Higher-order Feasibility

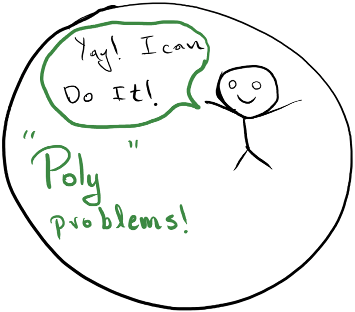
BFFs Characterization

Poly-time in a nutshell



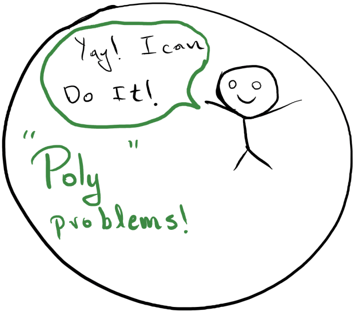
- Ordering a list of size n

Poly-time in a nutshell



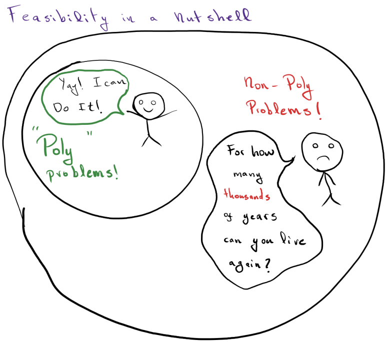
- Ordering a list of size n
- Computing the strongly connected components in a graph

Poly-time in a nutshell



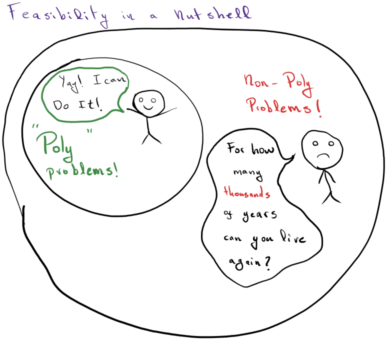
- Ordering a list of size n
- Computing the strongly connected components in a graph
- Adding/multiplying numbers (matrices)

Polytime in a nutshell



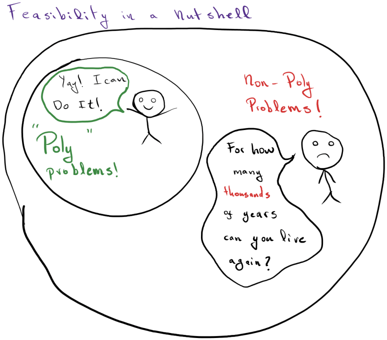
- decomposition of integers
- “a lot” of proof-searching algorithms
- automata learning

Polytime in a nutshell

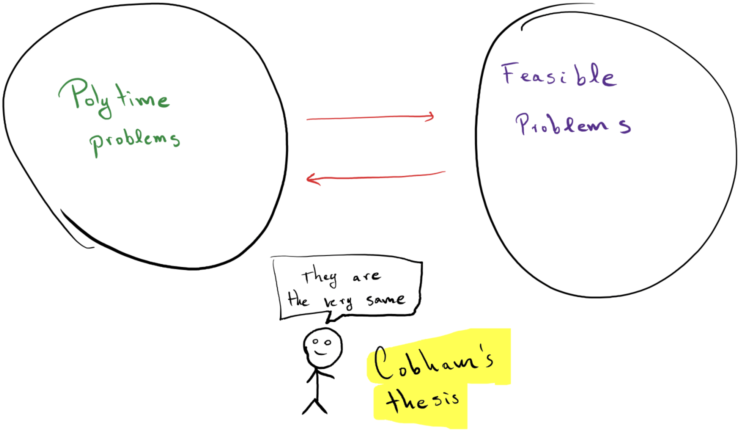


- "a lot" of proof-searching algorithms
- automata learning

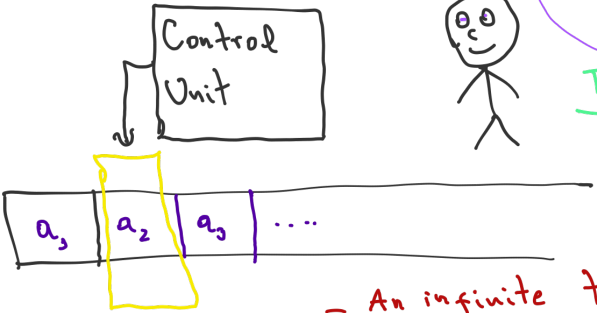
Polytime in a nutshell

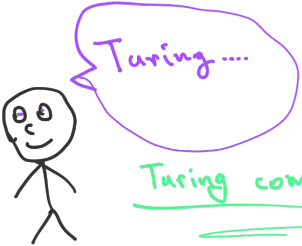


- automata learning


Poly time problems
Feasible Problems
They are the very same
Cobham's thesis

This is rather informal. . .





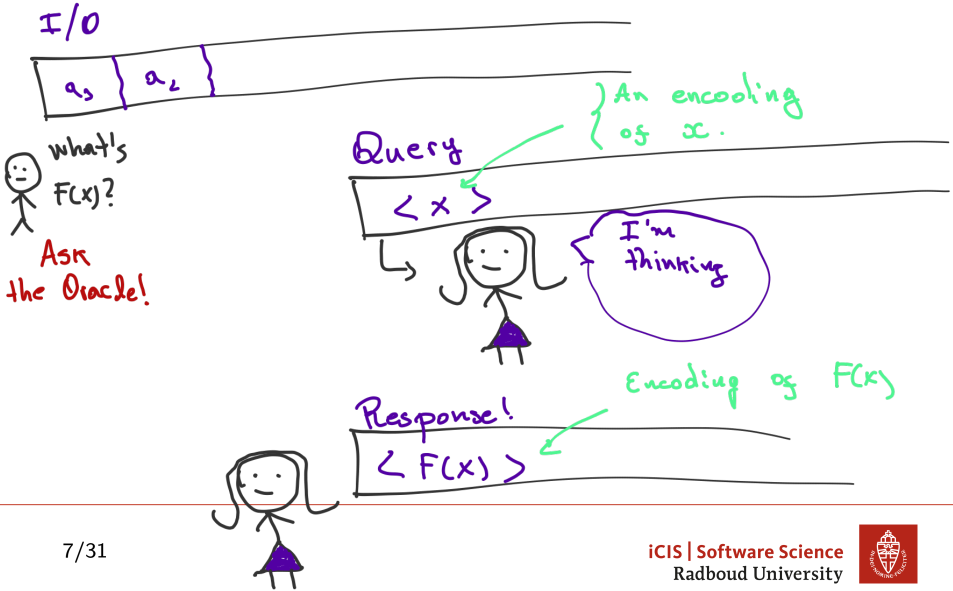
Turing computation

- An infinite tape
- The head moves L or R
- It can change the content of a cell

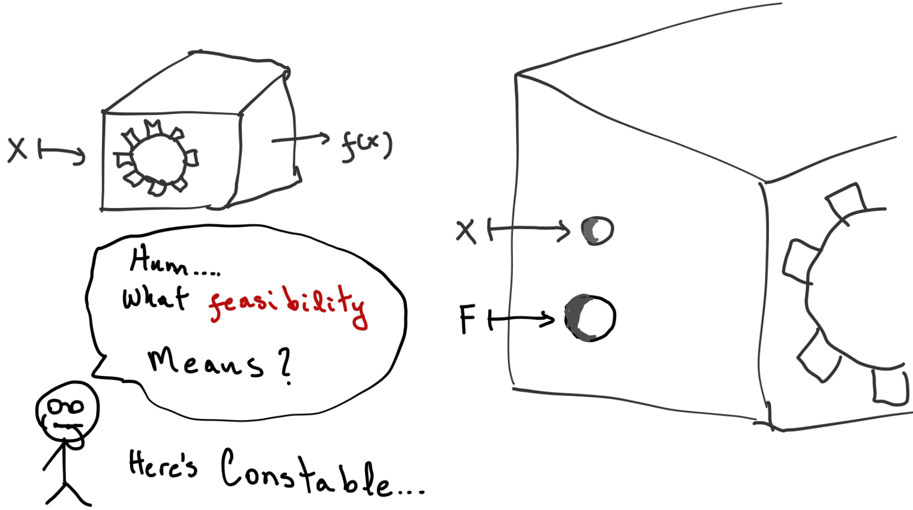
The "head"

Wait, machines have heads!

Hey Oracle! Compute this F at x for me!



Higher-Order what? "Feasibility!!!"



Outline

Poly-time in a nutshell

Higher-order Feasibility

BFFs Characterization

Constable problem

Constable (1973) posed the problem of finding a **natural analogue** of polynomial time (P) for (type-2) functionals:

$$(\mathbb{N} \to \mathbb{N})^k \times \mathbb{N}^\ell \to \mathbb{N}$$

This problem has been studied since the 70's.

Constable problem

Constable (1973) posed the problem of finding a **natural analogue** of polynomial time (P) for (type-2) functionals:

$$(\mathbb{N} \to \mathbb{N})^k \times \mathbb{N}^\ell \to \mathbb{N}$$

This problem has been studied since the 70's.

Why this problem is interesting?

- most tasks considered feasible are in P

Constable problem

Constable (1973) posed the problem of finding a **natural analogue** of polynomial time (P) for (type-2) functionals:

$$(\mathbb{N} \to \mathbb{N})^k \times \mathbb{N}^\ell \to \mathbb{N}$$

This problem has been studied since the 70's.

Why this problem is interesting?

- most tasks considered feasible are in P
- most tasks **outside of** P seems quite infeasible

Constable problem

Constable (1973) posed the problem of finding a **natural analogue** of polynomial time (P) for (type-2) functionals:

$$(\mathbb{N} \to \mathbb{N})^k \times \mathbb{N}^\ell \to \mathbb{N}$$

This problem has been studied since the 70's.

Why this problem is interesting?

- most tasks considered feasible are in P
- most tasks **outside of** P seems quite infeasible
- almost all **reasonable** models of deterministic computation are **polynomially** related

Constable problem

Constable (1973) posed the problem of finding a **natural analogue** of polynomial time (P) for (type-2) functionals:

$$(\mathbb{N} \to \mathbb{N})^k \times \mathbb{N}^\ell \to \mathbb{N}$$

This problem has been studied since the 70's.

Why this problem is interesting?

- most tasks considered feasible are in P
- most tasks **outside of** P seems quite infeasible
- almost all **reasonable** models of deterministic computation are **polynomially** related
- both P and FP have good closure properties

Basic Feasible Functionals (BFFs)

Good candidate? Let's bring. . . **BFFs**

Basic Feasible Functionals (BFFs)

Good candidate? Let's bring. . . **BFFs**

$\alpha : (\mathbb{N} \to \mathbb{N})^k \times \mathbb{N}^\ell \to \mathbb{N}$ is in $\texttt{BFF}$ if

Basic Feasible Functionals (BFFs)

Good candidate? Let's bring. . . **BFFs**

$\alpha : (\mathbb{N} \to \mathbb{N})^k \times \mathbb{N}^\ell \to \mathbb{N}$ is in BFF if

- there is an OTM M

Basic Feasible Functionals (BFFs)

Good candidate? Let's bring. . . **BFFs**

$\alpha : (\mathbb{N} \to \mathbb{N})^k \times \mathbb{N}^\ell \to \mathbb{N}$ is in BFF if

- there is an OTM M
- a second order polynomial P

Basic Feasible Functionals (BFFs)

Good candidate? Let's bring. . . **BFFs**

$\alpha : (\mathbb{N} \to \mathbb{N})^k \times \mathbb{N}^\ell \to \mathbb{N}$ is in $\texttt{BFF}$ if

- there is an OTM M
- a second order polynomial P
- M **computes** F

Basic Feasible Functionals (BFFs)

Good candidate? Let's bring. . . **BFFs**

$\alpha : (\mathbb{N} \to \mathbb{N})^k \times \mathbb{N}^\ell \to \mathbb{N}$ is in $\texttt{BFF}$ if

- there is an OTM M
- a second order polynomial P
- M **computes** F

$$\texttt{TIME}_M(\vec{f}, \vec{x}) \leq P(\vec{f}, \vec{x})$$

Goal

Our goal is to characterize **BFFs** via **higher-order rewriting** and **tuple interpretations**.

Higher-Order Rewriting

- function symbols with arity

Higher-Order Rewriting

- function symbols with arity
- terms are applicative (uncurried)

Higher-Order Rewriting

- function symbols with arity
- terms are applicative (uncurried)
- variables can be of higher-order type

Higher-Order Rewriting

- function symbols with arity
- terms are applicative (uncurried)
- variables can be of higher-order type

Higher-Order Rewriting

- function symbols with arity
- terms are applicative (uncurried)
- variables can be of higher-order type

$$\mathbb{R}_{\mathtt{map}} := \begin{cases} \mathtt{map}\ F\ \mathsf{nil} \to \mathsf{nil} \\ \mathtt{map}\ F\ x :: q \to (F\ x)\ \mathtt{map}\ F\ q \end{cases}$$

Tuple Interpretations 101

$$\llparenthesis \sigma \rrparenthesis = \mathcal{C}_\sigma \times \mathcal{S}_\sigma$$

Tuple Interpretations 101

$$(\!|\sigma|\!) = \mathcal{C}_\sigma \times \mathcal{S}_\sigma$$

$$(\!|\mathsf{nat}|\!) = \langle \text{cost}, \text{number of s's} \rangle$$

$$\mathcal{J}_{\mathsf{0}} = \left\langle \; \boxed{0} \; , 1 \right\rangle \qquad \mathcal{J}_{\mathsf{s}} = \left\langle \; \boxed{\boldsymbol{\lambda} x.0} \; , \boldsymbol{\lambda} x.x + 1 \right\rangle$$

Tuple Interpretations 101

$$(\!|\sigma|\!) = \mathcal{C}_\sigma \times \mathcal{S}_\sigma$$

$$(\!|\mathsf{list}|\!) = \langle \mathsf{cost}, (\mathsf{length},\ \mathsf{maximum\ element}) \rangle$$

$$\mathcal{J}_{\mathsf{nil}} = \left\langle\ 0\ , (0, 0) \right\rangle$$

$$\mathcal{J}_{\mathsf{cons}} = \left\langle\ \boldsymbol{\lambda} x.\boldsymbol{\lambda} q.0\ , \boldsymbol{\lambda} xq.(q_{\mathsf{l}} + 1, \max(x, q_{\mathsf{m}})) \right\rangle$$

Tuple Interpretations 101

$$(\!|\sigma|\!) = \mathcal{C}_\sigma \times \mathcal{S}_\sigma$$

Let's get back to `map`.

$$\mathbb{R}_{\texttt{map}} := \begin{cases} \texttt{map}\ F\ \mathsf{nil} \to \mathsf{nil} \\ \texttt{map}\ F\ x :: q \to (F\ x)\ \texttt{map}\ F\ q \end{cases}$$

$$[\![\texttt{map}(F, q)]\!]_{\mathsf{cost}} = (q_{\mathsf{l}} + 1) \cdot (\ \underbrace{[\![F]\!](q_{\mathsf{m}})_1}_{\text{behavior of } f!}\ )$$

Tuple Interpretations 101

$$(\!|\sigma|\!) = \mathcal{C}_\sigma \times \mathcal{S}_\sigma$$

Let's get back to `map`.

$$\mathbb{R}_{\texttt{map}} := \begin{cases} \texttt{map}\ F\ \mathsf{nil} \to \mathsf{nil} \\ \texttt{map}\ F\ x :: q \to (F\ x)\ \texttt{map}\ F\ q \end{cases}$$

$$[\![\texttt{map}(F, q)]\!]_{\mathsf{cost}} = (q_{\mathsf{l}} + 1) \cdot (\ \underbrace{[\![F]\!](q_{\mathsf{m}})_1}_{\textbf{behavior of } f\textbf{!}}\ )$$

$$[\![\texttt{map}(F, q)]\!]_{\mathsf{length}} = q_{\mathsf{l}}$$

Tuple Interpretations 101

$$(\!|\sigma|\!) = \mathcal{C}_\sigma \times \mathcal{S}_\sigma$$

Let's get back to `map`.

$$\mathbb{R}_{\mathtt{map}} := \begin{cases} \mathtt{map}\ F\ \mathsf{nil} \to \mathsf{nil} \\ \mathtt{map}\ F\ x :: q \to (F\ x)\ \mathtt{map}\ F\ q \end{cases}$$

$$[\![\mathtt{map}(F, q)]\!]_{\mathsf{cost}} = (q_{\mathsf{l}} + 1) \cdot (\underbrace{[\![F]\!](q_{\mathsf{m}})_1}_{\text{behavior of } f!})$$

$$[\![\mathtt{map}(F, q)]\!]_{\mathsf{length}} = q_{\mathsf{l}}$$

$$[\![\mathtt{map}(F, q)]\!]_{\mathsf{max}} = \underbrace{[\![F]\!](q_{\mathsf{c}}, q_{\mathsf{m}})_2}_{\text{behavior of } f}$$

Outline

Poly-time in a nutshell

Higher-order Feasibility

BFFs Characterization

How to characterize BFFs by Rewriting?

In order to capture BFFs we need to:

(**Soundness**) Show that if a TRS $\mathbb{R}$ **satisfying certain conditions** computes a type-2 functional $\alpha : (\mathbb{N} \longrightarrow \mathbb{N})^k \times \mathbb{N}^\ell \longrightarrow \mathbb{N}$

How to characterize BFFs by Rewriting?

In order to capture BFFs we need to:

(**Soundness**) Show that if a TRS $\mathbb{R}$ **satisfying certain conditions** computes a type-2 functional $\alpha : (\mathbb{N} \longrightarrow \mathbb{N})^k \times \mathbb{N}^\ell \longrightarrow \mathbb{N}$ **then** α is in $\mathtt{BFF}$

How to characterize BFFs by Rewriting?

In order to capture BFFs we need to:

(**Soundness**) Show that if a TRS $\mathbb{R}$ **satisfying certain conditions** computes a type-2 functional $\alpha : (\mathbb{N} \longrightarrow \mathbb{N})^k \times \mathbb{N}^\ell \longrightarrow \mathbb{N}$ **then** α is in $\mathtt{BFF}$

(**Completeness**) Show that if a functional $\alpha : (\mathbb{N} \longrightarrow \mathbb{N})^k \times \mathbb{N}^\ell \longrightarrow \mathbb{N}$ is in $\mathtt{BFF}$

How to characterize BFFs by Rewriting?

In order to capture BFFs we need to:

(**Soundness**) Show that if a TRS $\mathbb{R}$ **satisfying certain conditions** computes a type-2 functional $\alpha : (\mathbb{N} \longrightarrow \mathbb{N})^k \times \mathbb{N}^\ell \longrightarrow \mathbb{N}$

then α is in $\mathtt{BFF}$

(**Completeness**) Show that if a functional $\alpha : (\mathbb{N} \longrightarrow \mathbb{N})^k \times \mathbb{N}^\ell \longrightarrow \mathbb{N}$ is in $\mathtt{BFF}$

then there exists a TRS $\mathbb{R}$ **satisfying the same certain conditions** that computes α.

How to characterize BFFs by Rewriting?

In order to capture BFFs we need to:

- show that every TRS **satisfying certain conditions** **represent** a BFF

How to characterize BFFs by Rewriting?

In order to capture BFFs we need to:

- show that every TRS **satisfying certain conditions** **represent** a BFF
  - we limit constructor symbols to **additive interpretations**

How to characterize BFFs by Rewriting?

In order to capture BFFs we need to:

- show that every TRS **satisfying certain conditions** **represent** a BFF
  - we limit constructor symbols to **additive interpretations**
  - all defined symbols have polynomial bounded interpretations

How to characterize BFFs by Rewriting?

In order to capture BFFs we need to:

- show that every TRS **satisfying certain conditions** **represent** a BFF
  - we limit constructor symbols to **additive interpretations**
  - all defined symbols have polynomial bounded interpretations
  - we add an infinite number of extra function symbols `f` to represent the **calls to ORACLES**

How to characterize BFFs by Rewriting?

In order to capture BFFs we need to:

- show that every TRS **satisfying certain conditions** **represent** a BFF
  - we limit constructor symbols to **additive interpretations**
  - all defined symbols have polynomial bounded interpretations
  - we add an infinite number of extra function symbols `f` to represent the **calls to ORACLES**
  - the cost int. of each oracle call is 1 and the size is polynomially bounded

How to characterize BFFs by Rewriting?

In order to capture BFFs we need to:

- show that every TRS **satisfying certain conditions** represent a BFF
- show that every BFF can be embedded as a TRS

Higher-Order Rewriting with Oracles

Higher-Order Rewriting with Oracles

Definition

A set of rules $\mathbb{R}$ over Σ **defines a function** $f : \mathbb{N} \longrightarrow \mathbb{N}$ by way of the symbol $\mathtt{f}$ if the following conditions are satisfied:

Higher-Order Rewriting with Oracles

Definition

A set of rules $\mathbb{R}$ over Σ **defines a function** $f : \mathbb{N} \longrightarrow \mathbb{N}$ by way of the symbol $\mathtt{f}$ if the following conditions are satisfied:

- the only defined symbol used in $\mathbb{R}$ is $\mathtt{f}$;

Higher-Order Rewriting with Oracles

Definition
A set of rules $\mathbb{R}$ over Σ **defines a function** $f : \mathbb{N} \longrightarrow \mathbb{N}$ by way of the symbol $\mathtt{f}$ if the following conditions are satisfied:

- the only defined symbol used in $\mathbb{R}$ is $\mathtt{f}$;
- there is a bijection $\mathbb{N} \longrightarrow \mathcal{N}$, with $\mathcal{N} \subseteq \mathsf{T}(\Sigma^{\mathtt{con}})$, that is, each $n \in \mathbb{N}$ has a unique data representation $\ulcorner \mathtt{n} \urcorner$;

Higher-Order Rewriting with Oracles

Definition

A set of rules $\mathbb{R}$ over Σ **defines a function** $f : \mathbb{N} \longrightarrow \mathbb{N}$ by way of the symbol $\mathtt{f}$ if the following conditions are satisfied:

- the only defined symbol used in $\mathbb{R}$ is $\mathtt{f}$;
- there is a bijection $\mathbb{N} \longrightarrow \mathcal{N}$, with $\mathcal{N} \subseteq \mathsf{T}(\Sigma^{\mathtt{con}})$, that is, each $n \in \mathbb{N}$ has a unique data representation $\ulcorner \mathtt{n} \urcorner$;
- for each $n, m \in \mathbb{N}$ such that $m = f(n)$, there exists exactly one rule $\mathtt{f} \ulcorner \mathtt{n} \urcorner \rightarrow \ulcorner \mathtt{m} \urcorner$ in $\mathbb{R}$.

Higher-Order Rewriting with Oracles

Definition

Let it be given a finite TRS $(\mathbb{F}, \mathbb{R})$,

Higher-Order Rewriting with Oracles

Definition

Let it be given a finite TRS $(\mathbb{F}, \mathbb{R})$,

- a distinguished function symbol $\mathtt{F} \in \Sigma$ of type $(\mathsf{nat} \Rightarrow \mathsf{nat}) \Rightarrow \mathsf{nat} \Rightarrow \mathsf{nat}$,

Higher-Order Rewriting with Oracles

Definition

Let it be given a finite TRS $(\mathbb{F}, \mathbb{R})$,

- a distinguished function symbol $\mathtt{F} \in \Sigma$ of type $(\mathsf{nat} \Rightarrow \mathsf{nat}) \Rightarrow \mathsf{nat} \Rightarrow \mathsf{nat}$,
- a type-1 function $f : \mathbb{N} \longrightarrow \mathbb{N}$,

Higher-Order Rewriting with Oracles

Definition

Let it be given a finite TRS $(\mathbb{F}, \mathbb{R})$,

- a distinguished function symbol $\mathtt{F} \in \Sigma$ of type $(\mathsf{nat} \Rightarrow \mathsf{nat}) \Rightarrow \mathsf{nat} \Rightarrow \mathsf{nat}$,
- a type-1 function $f : \mathbb{N} \longrightarrow \mathbb{N}$,
- fresh symbols $\mathtt{G}, \mathtt{S}_f : \mathsf{nat} \Rightarrow \mathsf{nat}$ not in Σ

Higher-Order Rewriting with Oracles

Definition
Let it be given a finite TRS $(\mathbb{F}, \mathbb{R})$,

- a distinguished function symbol $\mathtt{F} \in \Sigma$ of type $(\mathsf{nat} \Rightarrow \mathsf{nat}) \Rightarrow \mathsf{nat} \Rightarrow \mathsf{nat}$,
- a type-1 function $f : \mathbb{N} \longrightarrow \mathbb{N}$,
- fresh symbols $\mathtt{G}, \mathtt{S}_f : \mathsf{nat} \Rightarrow \mathsf{nat}$ not in Σ

Higher-Order Rewriting with Oracles

Definition

Let it be given a finite TRS $(\mathbb{F}, \mathbb{R})$,

- a distinguished function symbol $\mathtt{F} \in \Sigma$ of type $(\mathsf{nat} \Rightarrow \mathsf{nat}) \Rightarrow \mathsf{nat} \Rightarrow \mathsf{nat}$,
- a type-1 function $f : \mathbb{N} \longrightarrow \mathbb{N}$,
- fresh symbols $\mathtt{G}, \mathtt{S}_f : \mathsf{nat} \Rightarrow \mathsf{nat}$ not in Σ

We write $\mathbb{R}_{\mathtt{F},f,\mathtt{G}}$ for the infinite TRS consisting of the rules in $\mathbb{R}$ together with the rules defining f by way of $\mathtt{S}_f$ and the rule:

$$\mathtt{G}\ x \rightarrow \mathtt{F}\ \mathtt{S}_f\ x$$

First-Order Rewriting Computability

Definition (Type-1 Computability)

Let $(\mathbb{F}, \mathbb{R})$ be a TRS and $\mathtt{f} \in \Sigma$. We say that the symbol $\mathtt{f}$ $\mathbb{R}$-**computes** a type-1 function $f : \mathbb{N} \longrightarrow \mathbb{N}$ **whenever**

$$\mathtt{f} \ulcorner \mathtt{n} \urcorner \twoheadrightarrow \ulcorner \mathtt{m} \urcorner \text{ iff } f(n) = m.$$

Higher-Order Rewriting Computability

Definition (Type-2 Computability)
We say that in a finite TRS $\mathbb{R}$ the function symbol $\texttt{F} : (\mathsf{nat} \Rightarrow \mathsf{nat}) \Rightarrow \mathsf{nat} \Rightarrow \mathsf{nat}$ **computes** the type-2 functional $\alpha : \mathbb{N}^{\mathbb{N}} \longrightarrow \mathbb{N} \longrightarrow \mathbb{N}$ iff

- for every type-1 function f in $\mathbb{N}^{\mathbb{N}}$,

Higher-Order Rewriting Computability

Definition (Type-2 Computability)
We say that in a finite TRS $\mathbb{R}$ the function symbol $\texttt{F} : (\mathsf{nat} \Rightarrow \mathsf{nat}) \Rightarrow \mathsf{nat} \Rightarrow \mathsf{nat}$ **computes** the type-2 functional $\alpha : \mathbb{N}^{\mathbb{N}} \longrightarrow \mathbb{N} \longrightarrow \mathbb{N}$ iff

- for every type-1 function f in $\mathbb{N}^{\mathbb{N}}$,
- the TRS $\mathbb{R}_{\texttt{F},f,\texttt{G}}$ is such that the symbol G computes $\alpha(f)$.

Polynomial tuple interpretations give BFF!

Theorem

Let $(\mathbb{F}, \mathbb{R})$ *be a finite TRS such that the symbol* $\texttt{F} \in \Sigma$ *computes the type-2 functional* $\alpha : \mathbb{N}^{\mathbb{N}} \longrightarrow \mathbb{N} \longrightarrow \mathbb{N}$.

Polynomial tuple interpretations give BFF!

Theorem

Let $(\mathbb{F}, \mathbb{R})$ *be a finite TRS such that the symbol* $\texttt{F} \in \Sigma$ *computes the type-2 functional* $\alpha : \mathbb{N}^{\mathbb{N}} \longrightarrow \mathbb{N} \longrightarrow \mathbb{N}$*.*

If $(\mathbb{F}, \mathbb{R})$ *is compatible with a polynomial interpretation*

Polynomial tuple interpretations give BFF!

Theorem

Let $(\mathbb{F}, \mathbb{R})$ *be a finite TRS such that the symbol* $\texttt{F} \in \Sigma$ *computes the type-2 functional* $\alpha : \mathbb{N}^{\mathbb{N}} \longrightarrow \mathbb{N} \longrightarrow \mathbb{N}$*.*

If $(\mathbb{F}, \mathbb{R})$ *is compatible with a polynomial interpretation*

then α *is in* $\texttt{BFF}$*.*

One tuple for the oracles that know it all!

$$\mathtt{G}\ x \rightarrow \mathtt{F}\ \mathtt{S}_f\ x$$

One tuple for the oracles that know it all!

$$\mathtt{G}\ x \to \mathtt{F}\ \mathtt{S}_f\ x$$

$$\mathcal{J}_{\mathtt{S}_f} = \left\langle \underbrace{(\boldsymbol{\lambda} x.1)}_{\text{cost of oracle}}, \underbrace{\boldsymbol{\lambda} x. \max_{y \leq x} f(y)}_{\text{size of oracle}} \right\rangle$$

One tuple for the oracles that know it all!

$$\mathtt{G}\ x \to \mathtt{F}\ \mathtt{S}_f\ x$$

$$\mathcal{J}_{\mathtt{S}_f} = \left\langle \underbrace{(\boldsymbol{\lambda} x.1)}_{\text{cost of oracle}}, \underbrace{\boldsymbol{\lambda} x. \max_{y \leq x} f(y)}_{\text{size of oracle}} \right\rangle$$

$$\begin{aligned} [\![\mathtt{S}_f\ \ulcorner \mathtt{n} \urcorner]\!] &= \left\langle (\boldsymbol{\lambda} x.1), \mathcal{J}^{\mathsf{s}}_{\mathtt{S}_f} \right\rangle \cdot \langle 0, n \rangle \\ &= \left\langle 1, \mathcal{J}^{\mathsf{s}}_{\mathtt{S}_f}(n) \right\rangle \\ &\succ \langle 0, m \rangle \end{aligned}$$

One tuple for the G that starts it all!

$$\mathtt{G}\ x \rightarrow \mathtt{F}\ \mathtt{S}_f\ x$$

One tuple for the G that starts it all!

$$\mathtt{G}\ x \to \mathtt{F}\ \mathtt{S}_f\ x$$

$$\mathcal{J}_{\mathtt{G}_f} = \langle(1, \mathcal{J}^{\mathsf{c}}_{\mathtt{F}}), \mathcal{J}^{\mathsf{s}}_{\mathtt{F}}\rangle \cdot [\![\mathtt{S}_f]\!]$$

One tuple for the G that starts it all!

$$\mathtt{G}\ x \to \mathtt{F}\ \mathtt{S}_f\ x$$

$$\mathcal{J}_{\mathtt{G}_f} = \langle (1, \mathcal{J}^{\mathsf{c}}_{\mathtt{F}}), \mathcal{J}^{\mathsf{s}}_{\mathtt{F}} \rangle \cdot [\![\mathtt{S}_f]\!]$$

$$\begin{aligned}
[\![\mathtt{G}\ x]\!] &= \left\langle 1 + \mathcal{J}^{\mathsf{c}}_{\mathtt{F}}(\left\langle \mathcal{J}^{\mathsf{c}}_{\mathtt{S}_f}, \mathcal{J}^{\mathsf{s}}_{\mathtt{S}_f} \right\rangle, x), \mathcal{J}^{\mathsf{s}}_{\mathtt{F}}(\mathcal{J}^{\mathsf{s}}_{\mathtt{S}_f}, x) \right\rangle \\
&\succ \left\langle \mathcal{J}^{\mathsf{c}}_{\mathtt{F}}(\left\langle \mathcal{J}^{\mathsf{c}}_{\mathtt{S}_f}, \mathcal{J}^{\mathsf{s}}_{\mathtt{S}_f} \right\rangle, x), \mathcal{J}^{\mathsf{s}}_{\mathtt{F}}(\mathcal{J}^{\mathsf{s}}_{\mathtt{S}_f}, x) \right\rangle \\
&= [\![\mathtt{F}]\!] \cdot [\![\mathtt{S}_f]\!] \cdot [\![x]\!] \\
&= [\![\mathtt{F}\ \mathtt{S}_f\ x]\!]
\end{aligned}$$

First-Order typed based interpretation

$$(\mathbb{F}, \mathbb{R})_f := \begin{cases} \mathtt{f}(0, y) \to y \\ \mathtt{f}(\mathsf{s}(x), y) \to \mathtt{f}(x, \mathsf{c}(y, y)) \end{cases}$$

$$\begin{array}{rclrcl} [\![0]\!] & = & 1 & [\![\mathsf{s}(x)]\!] & = & 4x + 1 \\ [\![\mathsf{c}(x, y)]\!] & = & x + y & [\![\mathtt{f}(x, y)]\!] & = & x + xy^2 + y \end{array}$$

The Size Explosion Problem

How many steps to normalize $t = f(\mathsf{s}^{100}(0), 0)$?

The Size Explosion Problem

How many steps to normalize $t = f(\mathsf{s}^{100}(0), 0)$?

$$\mathtt{f}(\mathsf{s}^{100}(0), 0) \quad \to \quad \mathtt{f}(\mathsf{s}^{99}(0), \underbrace{\mathsf{c}(0, 0)}_{c_0})$$

The Size Explosion Problem

How many steps to normalize $t = f(\mathsf{s}^{100}(0), 0)$?

$$
\begin{aligned}
\mathtt{f}(\mathsf{s}^{100}(0), 0) &\to \mathtt{f}(\mathsf{s}^{99}(0), \underbrace{\mathsf{c}(0, 0)}_{c_0}) \\
&\to \mathtt{f}(\mathsf{s}^{98}(0), \underbrace{\mathsf{c}(\mathsf{c}_0, \mathsf{c}_0)}_{\mathsf{c}_1})
\end{aligned}
$$

The Size Explosion Problem

How many steps to normalize $t = f(\mathsf{s}^{100}(0), 0)$?

$$
\begin{array}{rcl}
\mathtt{f}(\mathsf{s}^{100}(0), 0) & \rightarrow & \mathtt{f}(\mathsf{s}^{99}(0), \underbrace{\mathsf{c}(0, 0)}_{c_0}) \\
& \rightarrow & \mathtt{f}(\mathsf{s}^{98}(0), \underbrace{\mathsf{c}(\mathsf{c}_0, \mathsf{c}_0)}_{\mathsf{c}_1}) \\
& \vdots & \\
& \rightarrow & \mathtt{f}(\mathsf{s}^{100-i}(0), \mathsf{c}_{i-1})
\end{array}
$$

The Size Explosion Problem

How many steps to normalize $t = f(\mathsf{s}^{100}(0), 0)$?

$$\begin{aligned}
\mathtt{f}(\mathsf{s}^{100}(0), 0) &\to \mathtt{f}(\mathsf{s}^{99}(0), \underbrace{\mathsf{c}(0, 0)}_{c_0}) \\
&\to \mathtt{f}(\mathsf{s}^{98}(0), \underbrace{\mathsf{c}(\mathsf{c}_0, \mathsf{c}_0)}_{\mathsf{c}_1}) \\
&\vdots \\
&\to \mathtt{f}(\mathsf{s}^{100-i}(0), \mathsf{c}_{i-1}) \\
&\vdots \\
&\to \mathtt{f}(0, \mathsf{c}_{99})
\end{aligned}$$

The Size Explosion Problem

How many steps to normalize $t = f(\mathsf{s}^{100}(0), 0)$?

$$
\begin{array}{rcl}
\mathtt{f}(\mathsf{s}^{100}(0), 0) & \to & \mathtt{f}(\mathsf{s}^{99}(0), \underbrace{\mathsf{c}(0, 0)}_{c_0}) \\
& \to & \mathtt{f}(\mathsf{s}^{98}(0), \underbrace{\mathsf{c}(\mathsf{c}_0, \mathsf{c}_0)}_{\mathsf{c}_1}) \\
& \vdots & \\
& \to & \mathtt{f}(\mathsf{s}^{100-i}(0), \mathsf{c}_{i-1}) \\
& \vdots & \\
& \to & \mathtt{f}(0, \mathsf{c}_{99}) \\
& \to & \mathsf{c}_{99}
\end{array}
$$

The Size Explosion Problem

How many steps to normalize $t = f(\mathsf{s}^{100}(0), 0)$?

$$
\begin{aligned}
\mathtt{f}(\mathsf{s}^{100}(0), 0) &\rightarrow \mathtt{f}(\mathsf{s}^{99}(0), \underbrace{\mathsf{c}(0, 0)}_{c_0}) \\
&\rightarrow \mathtt{f}(\mathsf{s}^{98}(0), \underbrace{\mathsf{c}(\mathsf{c}_0, \mathsf{c}_0)}_{\mathsf{c}_1}) \\
&\vdots \\
&\rightarrow \mathtt{f}(\mathsf{s}^{100-i}(0), \mathsf{c}_{i-1}) \\
&\vdots \\
&\rightarrow \mathtt{f}(0, \mathsf{c}_{99}) \\
&\rightarrow \mathsf{c}_{99}
\end{aligned}
$$

Is the cost of $f(\mathsf{s}^n(0), 0)$ linear in n?

The Size Explosion Problem

How many steps to normalize $t = f(\mathsf{s}^{100}(0), 0)$?

$$
\begin{aligned}
\mathtt{f}(\mathsf{s}^{100}(0), 0) &\to \mathtt{f}(\mathsf{s}^{99}(0), \underbrace{\mathsf{c}(0,0)}_{c_0}) \\
&\to \mathtt{f}(\mathsf{s}^{98}(0), \underbrace{\mathsf{c}(\mathsf{c}_0, \mathsf{c}_0)}_{\mathsf{c}_1}) \\
&\ \vdots \\
&\to \mathtt{f}(\mathsf{s}^{100-i}(0), \mathsf{c}_{i-1}) \\
&\ \vdots \\
&\to \mathtt{f}(0, \mathsf{c}_{99}) \\
&\to \mathsf{c}_{99}
\end{aligned}
$$

Is the cost of $f(\mathsf{s}^n(0), 0)$ linear in n? c_{n-1} **is exponential in n!**

First-Order type-based interpretation

$$0 :: \mathsf{nat} \quad s :: \mathsf{nat} \Rightarrow \mathsf{nat} \quad c :: \mathsf{nat} \times \mathsf{nat} \Rightarrow \mathsf{nat} \quad f :: \mathsf{nat} \times \mathsf{nat} \Rightarrow \mathsf{nat}$$

First-Order type-based interpretation

$$\mathsf{0}::\mathsf{nat} \quad \mathsf{s}::\mathsf{nat} \Rightarrow \mathsf{nat} \quad \mathsf{c}::\mathsf{nat} \times \mathsf{nat} \Rightarrow \mathsf{nat} \quad \mathsf{f}::\mathsf{nat} \times \mathsf{nat} \Rightarrow \mathsf{nat}$$

$$[\![\mathsf{nat}]\!] = \langle\ \mathsf{cost}\ ,\ \mathsf{size}\ \rangle$$

First-Order type-based interpretation

$0 :: \mathsf{nat} \quad \mathsf{s} :: \mathsf{nat} \Rightarrow \mathsf{nat} \quad \mathsf{c} :: \mathsf{nat} \times \mathsf{nat} \Rightarrow \mathsf{nat} \quad \mathsf{f} :: \mathsf{nat} \times \mathsf{nat} \Rightarrow \mathsf{nat}$

$$[\![\mathsf{nat}]\!] = \langle \text{ cost }, \text{ size } \rangle$$

$$[\![0]\!] = \langle 0, 1 \rangle \qquad [\![\mathsf{s}(x)]\!] = \langle x_{\mathsf{c}}, x_{\mathsf{s}} + 1 \rangle$$

First-Order type-based interpretation

$0 :: \mathsf{nat} \quad \mathsf{s} :: \mathsf{nat} \Rightarrow \mathsf{nat} \quad \mathsf{c} :: \mathsf{nat} \times \mathsf{nat} \Rightarrow \mathsf{nat} \quad \mathsf{f} :: \mathsf{nat} \times \mathsf{nat} \Rightarrow \mathsf{nat}$

$$[\![\mathsf{nat}]\!] = \langle \ \mathsf{cost} \ , \ \mathsf{size} \ \rangle$$

$$\begin{array}{rclrcl} [\![0]\!] & = & \langle 0, 1 \rangle & [\![\mathsf{s}(x)]\!] & = & \langle x_\mathsf{c}, x_\mathsf{s} + 1 \rangle \\ [\![c(x, y)]\!] & = & \langle x_c + y_c, x_s + y_s \rangle & & & \end{array}$$

First-Order type-based interpretation

$0 :: \mathsf{nat} \quad \mathsf{s} :: \mathsf{nat} \Rightarrow \mathsf{nat} \quad \mathsf{c} :: \mathsf{nat} \times \mathsf{nat} \Rightarrow \mathsf{nat} \quad \mathsf{f} :: \mathsf{nat} \times \mathsf{nat} \Rightarrow \mathsf{nat}$

$$[\![\mathsf{nat}]\!] = \langle \ \mathsf{cost}\ ,\ \mathsf{size}\ \rangle$$

$$\begin{array}{rclrcl} [\![0]\!] & = & \langle 0, 1 \rangle & [\![\mathsf{s}(x)]\!] & = & \langle x_\mathsf{c}, x_\mathsf{s} + 1 \rangle \\ [\![c(x,y)]\!] & = & \langle x_c + y_c, x_s + y_s \rangle & & & \\ [\![f(x,y)]\!] & = & \langle x_\mathsf{c} + x_\mathsf{s} + 2^{x_\mathsf{s}} \cdot y_\mathsf{c}, 2^{x_\mathsf{s}} \cdot y_\mathsf{s} \rangle & & & \end{array}$$

Lemma (Subterm Lemma)

Let $(\mathbb{F}, \mathbb{R})$ *be a term rewriting system admitting a CPI. Then there is a second-order polynomial interpretation* P *such that for every type-1 functional* $f : \mathbb{N} \longrightarrow \mathbb{N}$*, data term* $\ulcorner \mathtt{n} \urcorner : \mathsf{nat}$*, and context* C*:*

$$\textit{\textbf{if}}\ \mathtt{F}\ \mathtt{S}_f\ \ulcorner \mathtt{n} \urcorner \twoheadrightarrow C[\mathtt{S}_f\ \ulcorner \mathtt{m} \urcorner]$$

Lemma (Subterm Lemma)

Let $(\mathbb{F}, \mathbb{R})$ *be a term rewriting system admitting a CPI. Then there is a second-order polynomial interpretation* P *such that for every type-1 functional* $f : \mathbb{N} \longrightarrow \mathbb{N}$*, data term* $\ulcorner \mathtt{n} \urcorner : \mathsf{nat}$*, and context* C*:*

$$\textbf{\textit{if}}\ \mathtt{F}\ \mathtt{S}_f\ \ulcorner \mathtt{n} \urcorner \twoheadrightarrow C[\mathtt{S}_f\ \ulcorner \mathtt{m} \urcorner]$$

$$\textbf{\textit{then}}\ |\ulcorner \mathtt{m} \urcorner| \leq P(|f|, |\ulcorner \mathtt{n} \urcorner|).$$

Polynomial tuple interpretations give BFF!

To prove this theorem we needed an interesting strategy:

- show that polynomial interpretations induce polynomial bounds to the runtime complexity of terms $\mathtt{G}\ \ulcorner\mathtt{n}\urcorner$

Polynomial tuple interpretations give BFF!

To prove this theorem we needed an interesting strategy:

- show that polynomial interpretations induce polynomial bounds to the runtime complexity of terms $\mathtt{G}\ \ulcorner \mathtt{n} \urcorner$
- fix the size-explosion problem computing with **graph rewriting**

Polynomial tuple interpretations give BFF!

To prove this theorem we needed an interesting strategy:

- show that polynomial interpretations induce polynomial bounds to the runtime complexity of terms $\mathtt{G}\ \ulcorner \mathtt{n} \urcorner$
- fix the size-explosion problem computing with **graph rewriting**
- show that OTMs can simulate graph rewriting with polynomial time overhead

Overview

One Tuple for the data c

Overview

One Tuple for the data c
additive all

Overview

One Tuple for the data c

additive all

One Tuple for the RULErS of $\mathbb{R}$

Overview

One Tuple for the data c

additive all

One Tuple for the RULErS of $\mathbb{R}$

bound by polynomials, you see

Overview

One Tuple for the data c
additive all
One Tuple for the RULErS of $\mathbb{R}$
bound by polynomials, you see
One Tuple for the Oracles that know it all

Overview

One Tuple for the data c

additive all

One Tuple for the RULErS of $\mathbb{R}$

bound by polynomials, you see

One Tuple for the Oracles that know it all

their knowledge a shield, embracing everything there are

Overview

One Tuple for the data c

additive all

One Tuple for the RULErS of $\mathbb{R}$

bound by polynomials, you see

One Tuple for the Oracles that know it all

their knowledge a shield, embracing everything there are

One Tuple to orient them all

Overview

One Tuple for the data c
additive all
One Tuple for the RULErS of $\mathbb{R}$
bound by polynomials, you see
One Tuple for the Oracles that know it all
their knowledge a shield, embracing everything there are
One Tuple to orient them all
one Tuple to forever bind them

Overview

One Tuple for the data c

additive all

One Tuple for the RULErS of $\mathbb{R}$

bound by polynomials, you see

One Tuple for the Oracles that know it all

their knowledge a shield, embracing everything there are

One Tuple to orient them all

one Tuple to forever bind them

For now in `BFF` they are

Overview

One Tuple for the data c

additive all

One Tuple for the RULErS of $\mathbb{R}$

bound by polynomials, you see

One Tuple for the Oracles that know it all

their knowledge a shield, embracing everything there are

One Tuple to orient them all

one Tuple to forever bind them

For now in `BFF` they are

These Tuples, we still need to find them!

Overview

One Tuple for the data c
additive all
One Tuple for the RULErS of $\mathbb{R}$
bound by polynomials, you see
One Tuple for the Oracles that know it all
their knowledge a shield, embracing everything there are
One Tuple to orient them all
one Tuple to forever bind them
For now in `BFF` they are
These Tuples, we still need to find them!

Overview

One Tuple for the data c

additive all

One Tuple for the RULErS of $\mathbb{R}$

bound by polynomials, you see

One Tuple for the Oracles that know it all

their knowledge a shield, embracing everything there are

One Tuple to orient them all

one Tuple to forever bind them

For now in `BFF` they are

These Tuples, we still need to find them!

Thank you!