An Implicit Characterization of the Polynomial-Time Decidable Sets by Cons-Free Rewriting

Daniel de Carvalho (Innopolis University, Russia)
Jakob Grue Simonsen (University of Copenhagen, Denmark)

Marseille, 2nd of September, 2016

Implicit computational complexity

We present a class of term rewriting systems that characterizes *implicitly* the complexity class **P** of the Polynomial-Time Decidable Sets.

Classical results in Implicit Computational Complexity:

- Leivant (1991)
- Bellantoni-Cook (1992)
- Goerdt (1992)
- Girard (1998)
- Jones (2001)
- etc...

Cons-free functional programs (Jones)

Definition. (Jones) A simply typed functional programm language is said to be *cons-free* (or *read-only*) if it is a program with no constructor operator.

Cons-free functional programs (Jones)

Definition. (Jones) A simply typed functional programm language is said to be *cons-free* (or *read-only*) if it is a program with no constructor operator.

Example. $< (x, y) =$ if null(y) then **false** else
(if null(x) then **true** else $<$ (tl(x), tl(y)))

Cons-free functional programs (Jones)

Definition. (Jones) A simply typed functional programm language is said to be *cons-free* (or *read-only*) if it is a program with no constructor operator.

Example. $< (x, y) =$ if null(y) then **false** else
(if null(x) then **true** else $<$ (tl(x), tl(y)))

Theorem. (Jones) The class of cons-free simply typed functional programming languages characterizes the complexity class **P**.

Non-deterministic cons-free programs

If we add non-determinism to cons-free programs, the *intensional* expressivity increases.

Non-deterministic cons-free programs

If we add non-determinism to cons-free programs, the *intensional* expressivity increases.

Definition. An input is *accepted* by a non-deterministic Turing machine (resp. a non-deterministic functional program) if there exists at least one computation that stops with an accepting state (resp. that evaluates to **true**).

Non-deterministic cons-free programs

If we add non-determinism to cons-free programs, the *intensional* expressivity increases.

Definition. An input is *accepted* by a non-deterministic Turing machine (resp. a non-deterministic functional program) if there exists at least one computation that stops with an accepting state (resp. that evaluates to **true**).

What is the *extensional* expressivity of non-deterministic cons-free functional programs?

Non-deterministic cons-free programs

If we add non-determinism to cons-free programs, the *intensional* expressivity increases.

Definition. An input is *accepted* by a non-deterministic Turing machine (resp. a non-deterministic functional program) if there exists at least one computation that stops with an accepting state (resp. that evaluates to **true**).

What is the *extensional* expressivity of non-deterministic cons-free functional programs?

Theorem. (Jones-Bonfante) The class of non-deterministic cons-free simply typed functional programming languages characterizes the complexity class **P**.

Functional programs versus TRS

Functional programs have a specific strategy (here: a call-by-value strategy).

Term rewriting systems have no specific reduction strategy.

An example of TRS

7 function symbols: $\mathbf{c}/2$, $\mathbf{n}/0$, $\mathbf{true}/0$, $\mathbf{false}/0$, $k/1$, $h/2$ and $p/1$.
4 rules:

- $p(\mathbf{c}(x, \mathbf{c}(y, z))) \to_1 x$
- $p(\mathbf{c}(x, \mathbf{c}(y, z))) \to_2 y$
- $h(x, \mathbf{false}) \to_3 x$
- $k(x) \to_4 h(x, x)$

Innermost reductions

Two innermost reductions (= call-by-value strategy):

Innermost reductions

Two innermost reductions (= call-by-value strategy):

- $k(p(\mathbf{c}(\mathbf{true}, \mathbf{c}(\mathbf{false}, \mathbf{n}))))$
  $\to_1 k(\mathbf{true})$
  $\to_4 h(\mathbf{true}, \mathbf{true})$
  and $h(\mathbf{true}, \mathbf{true})$ is a normal form

Innermost reductions

Two innermost reductions (= call-by-value strategy):

- $k(p(\mathbf{c}(\mathbf{true}, \mathbf{c}(\mathbf{false}, \mathbf{n}))))$
  $\to_1 k(\mathbf{true})$
  $\to_4 h(\mathbf{true}, \mathbf{true})$
  and $h(\mathbf{true}, \mathbf{true})$ is a normal form

- $k(p(\mathbf{c}(\mathbf{true}, \mathbf{c}(\mathbf{false}, \mathbf{n}))))$
  $\to_2 k(\mathbf{false})$
  $\to_4 h(\mathbf{false}, \mathbf{false})$
  $\to_3 \mathbf{false}$
  and $\mathbf{false}$ is a normal form

Non-innermost reductions

But we have also the following (non-innermost) reduction:

$k(p(\mathbf{c}(\mathbf{true}, \mathbf{c}(\mathbf{false}, \mathbf{n}))))$
$\to_4 h(p(\mathbf{c}(\mathbf{true}, \mathbf{c}(\mathbf{false}, \mathbf{n}))), p(\mathbf{c}(\mathbf{true}, \mathbf{c}(\mathbf{false}, \mathbf{n}))))$
$\to_1 h(\mathbf{true}, p(\mathbf{c}(\mathbf{true}, \mathbf{c}(\mathbf{false}, \mathbf{n}))))$
$\to_2 h(\mathbf{true}, \mathbf{false})$
$\to_3 \mathbf{true}$

Cons-free TRS

Definition. A *constructor TRS* is a TRS in which the set of function symbols is partitioned into

- a set $\mathcal{D}$ of *defined function symbols*
- and a set $\mathcal{C}$ of *constructors*

such that, for every rewrite rule $(l, r) \in \mathcal{R}$, the left-hand side l has the form $f(t_1, ..., t_n)$ with $f \in \mathcal{D}$ and $t_1, \ldots, t_n \in \mathcal{T}(C)$, the set of terms built from variables and constructors.

Definition. A *cons-free TRS* is a finite constructor TRS s.t., for every rule $(f(c_1, ..., c_q), r)$, we have:

- for any $c(u_1, \ldots, u_n) \trianglelefteq r$ with $c \in \mathcal{C}$, we have $c(u_1, \ldots, u_n) \trianglelefteq l$ or $c(u_1, \ldots, u_n) \in \mathcal{T}_0(\mathcal{C})$, the set of ground terms built from constructors;
- and, for any variable $x \in \{c_1, \ldots, c_q\}$ s.t. $x \not\trianglelefteq c_1, \ldots, c_q$, we have $c_i = x = c_j \Rightarrow i = j$.

Constrained cons-free TRS

Definition. A cons-free TRS R is said to be *constrained* if there exists $\mathcal{A} \subseteq \mathcal{D}$ such that, for any rule $(f(c_1, ..., c_q), r)$ and for any variable $x \in \{c_1, \ldots, c_q\}$ such that $x \notin c_1, \ldots, c_q$, we have:

- $(\forall p, p' \in Pos(r))(r(p) = x \Rightarrow (p' < p \Rightarrow r(p') \in \mathcal{A}))$
- and $f \in \mathcal{A} \Rightarrow Oc(x, r) \leq 1$.

Example of constrained cons-free TRS

4 constructors: $\mathbf{c}/2$, $\mathbf{n}/0$, $\mathbf{true}/0$, $\mathbf{false}/0$
We have 3 defined function symbols: $k/1$, $h/2$ and $p/1$.
We set $\mathcal{A} = \{h, p\}$.
We have 4 rules:

- $p(\mathbf{c}(x, \mathbf{c}(y, z))) \to_1 x$
- $p(\mathbf{c}(x, \mathbf{c}(y, z))) \to_2 y$
- $h(x, \mathbf{false}) \to_3 x$
- $k(x) \to_4 h(x, x)$

Characterizing **P**

Theorem. Let $L \subseteq \{0,1\}^{<\infty}$. Then, $L \in P$ if, and only if, there exists a constrained cons-free TRS over some signature $\mathcal{F} = \mathcal{D} \cup \mathcal{C}$ such that

- $\{\mathbf{one}/1, \mathbf{zero}/1, \mathbf{nil}/0, \mathbf{true}/0\} \subseteq \mathcal{C}$,
- and there is $f \in \mathcal{D}$ such that, for any $t \in \mathcal{T}_0(\{\mathbf{one}/1, \mathbf{zero}/1, \mathbf{nil}/0\})$, we have $f(t) \to^* \mathbf{true}$ if, and only if, $\langle t \rangle \in L$.

Characterizing **P**

Theorem. Let $L \subseteq \{0,1\}^{<\infty}$. Then, $L \in P$ if, and only if, there exists a constrained cons-free TRS over some signature $\mathcal{F} = \mathcal{D} \cup \mathcal{C}$ such that

- $\{\mathbf{one}/1, \mathbf{zero}/1, \mathbf{nil}/0, \mathbf{true}/0\} \subseteq \mathcal{C}$,
- and there is $f \in \mathcal{D}$ such that, for any $t \in \mathcal{T}_0(\{\mathbf{one}/1, \mathbf{zero}/1, \mathbf{nil}/0\})$, we have $f(t) \to^* \mathbf{true}$ if, and only if, $\langle t \rangle \in L$.

Proof. The completeness direction is (essentially) like in Jones. For the correctness direction, we proceed in two steps:

- we extend the calculus and show that we can consider this new calculus instead of the original one
- we use memoization in this new calculus

Extending the calculus (1/3)

(Non-innermost) Reductions in the original TRS can be simulated by innermost reductions in some extended calculus:

Extending the calculus (1/3)

(Non-innermost) Reductions in the original TRS can be simulated by innermost reductions in some extended calculus:

- We set $\Delta_0 = \mathcal{T}_0(\mathcal{C})$.
- We set $\Delta_{i+1} = \Delta_i \cup \bigcup_{m\in\mathbb{N}}\{f(U_1,\ldots,U_m); f \in \mathcal{D}$ *of arity* m *and* $U_1,\ldots,U_m \in \mathbf{2}\langle\Delta_i\rangle\}$,

where $\mathbf{2}$ is the semi-ring $\{0,1\}$ with $1+1=1$ and $\mathbf{2}\langle\mathcal{E}\rangle$ is the free $\mathbf{2}$-semi-module on $\mathcal{E}$.
We set $\Delta = \bigcup_{i\in\mathbb{N}} \Delta_i$.
Notice that

- any ground (usual) term is an element of $\mathbf{2}\langle\Delta\rangle$
- $0 \in \mathbf{2}\langle\Delta_i\rangle$ for any i

Extending the calculus (2/3)

Definition. We define the relation $\lhd$ on $\mathbf{2}\langle\Delta\rangle$ as follows:
For any $U, U' \in \mathbf{2}\langle\Delta\rangle$, we have $U \lhd U'$ iff there exist a context $C[]$, $u \in \Delta_1 \setminus \Delta_0$ and $V \in \mathbf{2}\langle\Delta\rangle$ such that $U = C[u]$, $U' = C[V]$ and $(u \lhd_\Delta V$ or $V = 0)$, where:
$f(V_1, \ldots, V_q) \lhd_\Delta V$ iff there exist a rule $(f(c_1, \ldots, c_q), r)$ and a partial function φ from the set of variables to $\mathbf{2}\langle\mathcal{T}_0(\mathcal{C})\rangle$ such that $V = r^\varphi$ and, for any $j \in \{1, \ldots, q\}$,

- $V_j = {c_j}^\varphi$
- and if c_j is not a variable, then, for any variable x occurring in c_j, we have $\varphi(x) \in \mathcal{T}_0(\mathcal{C})$.

Extending the calculus (3/3)

For any $U \in \mathbf{2}\langle\Delta\rangle$, we denote by $\|U\|$ the maximum number of distict summands occurring anywhere in U.
For any integer k, we set $\mathbf{2}\langle\Delta\rangle_k = \{U \in \mathbf{2}\langle\Delta\rangle; \|U\| \leq k\}$ and $\lhd_k = \lhd_{\mathbf{2}\langle\Delta\rangle_k \times \mathbf{2}\langle\Delta\rangle_k}$.

Given a constrained cons-free TRS, we let $K \geq 1$ be an integer such that, for any rule $(f(c_1, ..., c_q), r)$, for any variable $x \in \{c_1, ..., c_q\}$, $Oc(x, r) \leq K$.

Proposition. Let $c_1, \ldots, c_q \in \mathcal{T}_0(\mathcal{C})$, let $c \in \mathcal{T}_0(\mathcal{C})$ and $f \in \mathcal{D}$ of arity q . We have $f(c_1, ..., c_q) \to^* c$ iff $f(c_1, ..., c_q) \lhd_K{}^* c$.

Memoization

Given $f \in \mathcal{D}$ of arity q and $c_1, \ldots, c_{q+1} \in \mathcal{T}_0(\mathcal{C})$. We want to know whether $f(c_1, \ldots, c_q) \to^* c_{q+1}$.
Let $\mathcal{I}$ be the set of ground constructor terms c such that c is a subterm of $c_1, \ldots, c_q$ or of some $c_0 \in \mathcal{T}_0(\mathcal{C})$ occurring in the right-handside of some rule.

- If $c_{q+1} \notin \mathcal{I}$, then the answer is NO.
- Otherwise, we consider the table having as entries the pairs $(g(C_1, \ldots, C_p), c)$ with $g \in \mathcal{D}$, $C_1, \ldots, C_p \in \mathbf{2}\langle\mathcal{I}\rangle$, $\|C_1\|, \ldots, \|C_p\| \leq K$ and $c \in \mathcal{I}$. For any entry (t, c) that is not filled, we use the table to check whether $t \triangleleft_K{}^* c$ and, if it is successful, then we fill the table at this input.

The size of the table is polynomial in the size of $(c_1, \ldots, c_q)$.

What about non-constrained cons-free TRS's?

What about non-constrained cons-free TRS's?

Theorem. (Kop-Grue Simonsen 2016)
Left-linear cons-free TRS's characterize the class **E** (= the class of problems that are solvable in exponential time with linear exponent).

Conclusion

Starting from a characterization of **P** by Jones, we obtained a characterization of **P** by first-order TRS's.

Conclusion

Starting from a characterization of **P** by Jones, we obtained a characterization of **P** by first-order TRS's.

As shown by Kop and Grue Simonsen, one cannot remove the new constraints we added.

Conclusion

Starting from a characterization of **P** by Jones, we obtained a characterization of **P** by first-order TRS's.

As shown by Kop and Grue Simonsen, one cannot remove the new constraints we added.

Further work: Starting from a characterization of **P** by Goerdt, we should obtain a characterization of **NP** by *second-order* functional programming languages.