Tailoring Recursion to Characterize Non-Deterministic Complexity Classes Over Arbitrary Structures
Extended abstract

O. Bournez[1], F. Cucker[⋆2], P. Jacobé de Naurois[1], and J.-Y. Marion[1]

[1] LORIA, 615 rue du Jardin Botanique,BP 101, 54602 Villers-lès-Nancy Cedex, Nancy, FRANCE
e-mail: {Olivier.Bournez,Paulin.De-Naurois,Jean-Yves.Marion}@loria.fr
[2] Department of Mathematics, City University of Hong Kong, 83 Tat Chee Avenue, Kowloon, HONG KONG
e-mail: macucker@math.cityu.edu.hk

1 Focus

1.1 BSS Model of Computation

Definition 1. *A* structure $\mathcal{K} = (\mathbb{K}, \{op_i\}_{i\in I}, rel_1, \ldots, rel_l, \mathbf{0}, \mathbf{1})$ *is given by some underlying set* $\mathbb{K}$*, a family of operators* op_i*, and a finite number of relations* $rel_1, \ldots, rel_l$*.*

A BSS machine over $\mathcal{K}$ is, roughly speaking, a Turing Machine which is able to perform basic operations op_i and test rel_i at unit cost, and whose cells can hold arbitrary elements of $\mathbb{K}$. Notion of complexity classes such as $\mathrm{P}_{\mathcal{K}}$ and completeness of computational problems arise. When $\mathcal{K}$ is a Discrete structure, these coincide with those of the classical complexity theory.

Similarly, a natural notion of computational circuit over $\mathcal{K}$ exists, extending the classical notion of decision circuits with computation nodes for the operations op_i and relation nodes for the rel_i.

1.2 A Previous Characterization of Polynomial Deterministic Time

Similarly to the classical complexity theory, machine independent characterization of complexity classes can be obtained. In the spirit of Bellantoni and Cook's characterization of P in terms of algebras of safe recursive functions, we have defined a notion of recursive function over $\mathcal{K}$. Theses functions take as arguments words (lists of variable length) of elements of $\mathbb{K}$ and output one such word of elements of $\mathbb{K}$. The length of these arguments is used to control recursion.

Safe recursive functions have two different types of arguments, each of them having different properties and purposes. The first type of argument, called *normal*, can be used to make basic computation steps or to control recursion. The

⋆ Partially supported by City University of Hong Kong SRG grant 7001290.

second type of argument, called *safe*, can not be used to control recursion. This distinction between safe and normal arguments ensures that safe recursive functions can be computed in polynomial time.

The set of safe recursive functions over $\mathcal{K}$ is formally defined as the closure of some basic functions -elementary manipulations of the arguments, operations and relations, a selector, and projections- under safe composition and safe recursion schemes. It coincides with the set of functions computable in polynomial time by a BSS machine.

Similarly, the set of safe recursive functions with substitutions over $\mathcal{K}$ is the closure of the same basic functions under safe composition and safe recursion with substitutions. It coincides with the set of functions computable in parallel polynomial time.

2 Capturing the Polynomial Hierarchy

The polynomial hierarchy over $\mathcal{K}$ is defined in a similar way as the classical polynomial hierarchy. The first level is $\Sigma^0_{\mathcal{K}} = \Pi^0_{\mathcal{K}} = \mathrm{P}_{\mathcal{K}}$. Then,

$$\Sigma^1_{\mathcal{K}} = \mathrm{NP}_{\mathcal{K}} \qquad \Pi^1_{\mathcal{K}} = \mathrm{coNP}_{\mathcal{K}}$$
$$\Sigma^{i+1}_{\mathcal{K}} = \mathrm{NP}_{\mathcal{K}}^{\Sigma^i_{\mathcal{K}}} \; \Pi^{i+1}_{\mathcal{K}} = \mathrm{NP}_{\mathcal{K}}^{\Sigma^i_{\mathcal{K}}}$$

We denote by $F\Delta^i_{\mathcal{K}} = \mathrm{FP}_{\mathcal{K}}^{\Sigma^i_{\mathcal{K}}} = \mathrm{FP}_{\mathcal{K}}^{\Pi^i_{\mathcal{K}}}$ the levels of the functional polynomial hierarchy. These are the sets of functions computable with oracle queries to the corresponding levels of the decisional polynomial hierarchy, and is closed by complement.

Steming on the previous characterization of $\mathrm{P}_{\mathcal{K}}$, we introduce a predictive scheme, denoted as *predicative minimization*. When used in combination with the safe recursion scheme and safe composition scheme above, this yields new characterization of every level of the functional polynomial hierarchy.

A similar result is obtained for digital quantification.

3 Capturing Polynomial Alternation

A decision problem S belongs to the class $\mathrm{PAT}_{\mathcal{K}}$ when there exists a polynomial time BSS machine M_S and a polynomial function q such that:

$$x \in S \Leftrightarrow \exists a_1 \in \mathbb{K} \forall b_1 \in \mathbb{K} \ldots \exists a_{q(n)} \forall b_{q(n)}$$
$$M_S \text{ accepts on input } (x, a_1, b_1, \ldots, a_{q(n)}, b_{q(n)})$$

The number of quantifiers here is not fixed for a given problem S, but actually depends on the size n of the input. Note that the parallel polynomial time class over $\mathcal{K}$ is included in $\mathrm{PAT}_{\mathcal{K}}$. When $\mathcal{K}$ is a discrete structure, they both coincide with PSPACE.

A slight modification in the schemes introduced in the previous section allow us to capture $\mathrm{PAT}_m K$. When restricted to discrete structure, this gives us a characterization of PSPACE very different from the previous ones.

A similar result is also obtained for digital quantification.