Separability problem in automata theory

Wojciech Czerwiński

May 31, 2016

In automata theory a fundamental notion are finite automata and languages recognizable by them: family of regular languages. From the logic perspective regular languages have a different characterization: they are exactly these languages, which can be defined by a formula of monadic second order logic (MSO). An example formula is:

$$\exists_X \big(\forall_{x\in X} a(x) \wedge x+1 \notin X\big) \wedge \big(\forall_{x\notin X} b(x) \wedge x-1 \in X\big),$$

which roughly says that there is a set X of positions, such that on positions from X there are letters a, on positions outside X there are letters b and positions alternatingly belong to the set X and do not belong to it. This formula defines a language $\{(ab)^n \mid n \in \mathbb{N}\}$. If we restrict the set of constructions allowed in the logic then not all the regular languages will be still definable by formulas. Example restrictions are: using only first order formulas, bounding depth of quantifier nesting, prohibiting a predicate $+1$, allowing only for existential quantifiers, etc. In that way we get many natural subclasses of regular languages.

For understanding the expressive power of these subclasses the natural question is the *characterization problem*: given a regular language L, does it belong to a given family of languages $\mathcal{F}$? We say that language S *separates* languages K and L if $K \subseteq S$ and $L \cap S = \emptyset$. Another important question is a generalization of characterization problem, *separability problem*: given two regular languages K, L, does there exists a language $S \in \mathcal{F}$, which separates K and L? Both problems can be also considered in a broader generality, when the investigated languages are not necessarily regular, but come from some family $\mathcal{G}$. Then separability problem of languages from $\mathcal{G}$ by languages from $\mathcal{F}$ is of the form: given two languages $K, L \in \mathcal{G}$, does there exists a language $S \in \mathcal{F}$, which separates K and L.

Separability problem is natural and important theoretical problem, which we can be witnessed by the fact that we find it in many seemingly not connected questions. Imagine that we investigate XML data files, some of them fit to our DTD schema and some do not. We would like to distinguish whether a file fits to a schema using an algorithm, which reads file from the beginning to the end and uses a finite memory. It turns out that this is actually a problem of separating correct XML files fitting to the schema from correct XML files not fitting to the schema by some regular language. It is easy to see that we can generalize the above story to any file format and any its property we want to investigate.

Another situation in which separability problem arises naturally is simplification and approximating, for example in verification. Assume that some behaviors of a program, which we want to verify, are desirable, another are prohibited, but some other are actually neutral, there is no gain from them, but also nothing bad will happen if they occur. However it may happen that description of the desirable and prohibited behaviors is very complicated, it is therefore hard to check whether exactly them occur. We can avoid the problem and compute some language of behaviors which separates desirable and prohibited behaviors. It will be some upper approximation of good behaviors, which will contain all the good ones, but also possibly some neutral ones. Such an approximation could be much easier to analyze and also, importantly, also much simpler to understand for a human, not necessarily a programmer.

In a project we plan to work on theoretical basics of separability problem. The first goal is to investigate the separability problem of a broad class of Petri net languages by regular languages and maybe show that this problem is decidable, so one can solve it by an algorithm. The second task is to understand which separability problems for regular tree languages are decidable, trees naturally appear in many cases, for example in XML analysis. The third goal is to investigate computational complexity of separability problems, so in other words designing possibly fast algorithms or proving that such algorithms cannot be found at all.