TOPOLOGICAL PROPERTIES OF INFINITE COMPUTATIONS

MICHAŁ SKRZYPCZAK

Abstract. The in-out model of computer programs is not suitable for "reactive" systems (e.g. servers). Appropriate model for them is an idea of infinite computation — a sequence of states, indexed by natural numbers. But how to define (or better automatically check) properties of such objects? It turns out, that this problem naturally connects logic, topology and combinatorics.

This document presents basic notions and results from the area, including ω-regular languages, the MSO logic and the Borel hierarchy. At the end of the article, some current problems are discussed.

1. Motivation

Computer systems are becoming more and more complicated. Moreover, they control many important elements of our environment: traffic lights, flight parameters or even medical equipment. Therefore, sometimes we need to make sure, that a program does not contain any error.

Usually this task is divided into the following steps:

(1) Define a semantics of programs,
(2) express expected properties in some formal language,
(3) proof or better automatically check that a given program fulfils given specification.

There are well known methods of performing this tasks, for example programmers can:

- write programs in declarative languages, where program itself is a kind of a specification,
- define a formal semantics of the given programming language (e.g. small steps, big steps or continuations semantics),
- use induction, Hoare logic, etc. to create formal proofs of correctness,
- ...

But all this methods implicitly treat a program as some kind of a (partial) in$\to$out function: program reads an input, performs some computations step by step and outputs the result. This schema is appropriate for many programs, including `pdflatex` used to create this article. But what about servers that run potentially forever?

Example 1. Mail server has three states: `receiving`, `reordering`, `sending`. We want to ensure that the server will infinitely often receive:

$$\forall_{n\in\mathbb{N}}\exists_{k\geq n}\ S(k) = \text{receiving}.$$

Imagine we obtained a source code of some mail server. How can we check that it satisfies the property defined above? The following problems occur:

- We cannot just simulate the whole computation — its infinite,
- First order logic over $(\mathbb{N}, +, \cdot)$ is undecidable, so even simple properties of moments of time cannot be automatically checked.
- Servers are parallel — they communicate one with another. Therefore, testing a server may not expose possible errors: for example the time when an error may occur during a communication is too short for currently used hardware.

The solution presented in this paper is based on the following ideas (explained in detail later):

(1) model a server as a finite state machine with states denoted by Σ,
(2) treat an infinite computation as a sequence $\alpha \in \Sigma^\omega$,
(3) use *Monadic Second Order logic* to express and check properties of such computations.

2. MSO Logic

In this section we define the MSO logic, show some properties definable in it and present the decidability theorem of Büchi.

Definition 2. MSO logic is second order logic with additional restriction that second order quantification may bind only sets of elements.

In other words, it is a set of formulas containing:

- $x \in X$ for variables x, X,
- $R_i(x_1, \ldots, x_n)$ for every relational symbol R_i,
- $\varphi \vee \psi$,
- $\neg\varphi$,
- $\exists_x\ \varphi(x)$, where x is a variable valued in elements of the structure,
- $\exists_X\ \varphi(X)$, where X is a variable valued in subsets of the structure,

where φ, ψ denote simpler MSO formulas.

The semantics of such formulas is defined in the natural way.

We will restrict ourselves to structures representing infinite words.

Definition 3. Every $\alpha \in \Sigma^\omega$ can be treated as a relational structure of the form $\langle \omega, \leq, \Sigma \rangle$, where

- $\omega = \{0, 1, 2, \ldots\}$ is a carrier of the structure,
- $\leq$ is a binary relation of order on ω,
- for each $A \in \Sigma$ there is an unary predicate A, defined by equation

$$A(n) \Leftrightarrow \alpha_n = A.$$

Definition 4. For a given MSO formula φ, we construct *a language defined by* φ

$$L(\varphi) = \{\alpha \in \Sigma^\omega : \alpha \models \varphi\},$$

that is a set of such $\alpha \in \Sigma^\omega$ that (when treated as a structure) satisfy φ.

To make things easy the signature of the language contains only $\leq$ and elements of Σ. But it is easy to see that using $\leq$ we can define a constant $0 \in \omega$ and a function $s(n) = n + 1$.

Example 5. Take $\Sigma = \{A, B\}$ and consider a MSO formula

$$\varphi = \exists_P \forall_n \ (n \in P \Leftrightarrow s(n) \notin P) \land 0 \in P \land \exists_k \ k \in P \land A(k).$$

There is exactly one set $P \subseteq \omega$ satisfying property $\forall_n \ n \in (P \Leftrightarrow s(n) \notin P) \land 0 \in P$ — the set of even numbers. So $\alpha \models \varphi$ if and only if there exists an even number k such that $\alpha_k = A$.

Definition 6. The family of all sets of the form $L(\varphi) \subseteq \Sigma^\omega$ will be called the family of ω-regular languages (over alphabet Σ).

Sometimes we will consider the following logic.

Definition 7. By WMSO (Weak Monadic Second Order logic) we denote a logic with the same syntax as MSO logic. The only difference is that second order quantifiers in WMSO range over finite sets only. So for example $\exists_X \ \varphi(X)$ means that there exists *finite* set X such that $\varphi(X)$.

Since MSO logic enables us to express that a given set X is finite, every formula of WMSO can be rewritten to the equivalent MSO formula.

Now we proceed to the most important theorem.

Theorem 8 (Büchi 1960)**.** *The emptiness problem for* MSO *over infinite words is decidable. That means, there exists an algorithm that reads a formula φ and outputs* 0 *if $L(\varphi) = \emptyset$ and* 1 *if $L(\varphi) \neq \emptyset$.*

Before we sketch the proof, first observe that solving an emptiness problem can be used to solve the following problems:

- given φ, ψ, answer whether $L(\varphi) \subseteq L(\psi)$ — consider $\gamma = \varphi \land \neg\psi$,
- given φ, ψ, answer whether $L(\varphi) \cap L(\psi) = \emptyset$ — consider $\gamma = \varphi \land \psi$,
- given φ, answer whether $L(\varphi) = \Sigma^\omega$ — consider $\gamma = \neg\varphi$.

Idea of the proof of Theorem 8. The sketch of the proof is explained in the following steps.

(1) Define a model of finite automata that read an infinite word and accept or reject it.
(2) Show that each MSO formula φ can be automatically transformed into an equivalent automaton $\mathcal{A}$.
(3) Write a program that reads $\mathcal{A}$ and answers whether $\mathcal{A}$ is able to accept any infinite word.

The construction of the automata and an insight into second step of the proof are presented in the next section. □

3. Automata

In the original paper of Büchi so called "nondeterministic Büchi automata" were used. To simplify, we will work with other (equivalent) model.

Definition 9. Deterministic parity automaton is a tuple $\mathcal{A} = \langle q_0, Q, \delta, \Omega \rangle$ where

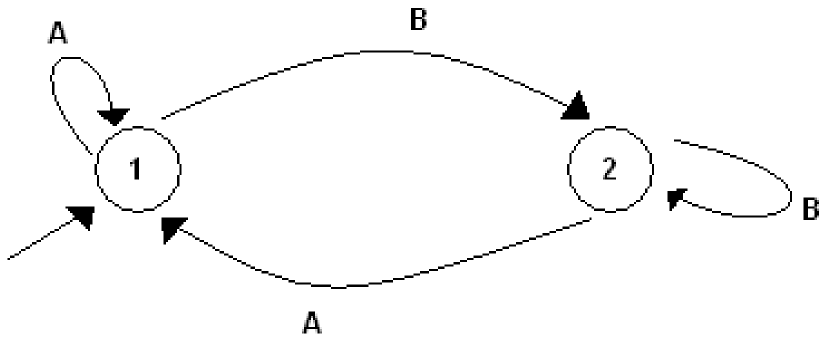
- q_0 is an element of Q called *initial state*,
- Q is a finite set of *states*,
- $\delta \colon Q \times \Sigma \to Q$ is a *transition function*,
- $\Omega \colon Q \to \mathbb{N}$ maps a state $q \in Q$ into its *rank* $\Omega(q) \in \mathbb{N}$.

For a given word $\alpha \in \Sigma^\omega$, we define the *run* $\tau \in Q^\omega$ inductively $\tau_0 = q_0$ and $\tau_{n+1} = \delta(\tau_n, \alpha_n)$. We say that $\mathcal{A}$ accepts a word α iff

$$\liminf_{n\to\infty} \Omega(\tau_n) \equiv 1 \mod 2.$$

By $L(\mathcal{A})$ (language recognised by automaton $\mathcal{A}$) we denote the set of all words $\alpha \in \Sigma^\omega$ accepted by $\mathcal{A}$.

Example 10. Consider automaton $\mathcal{A}$: $Q = \{1, 2\}$, $q_0 = 1$, $\Omega(q) = q$, $\delta(q, A) = 1$ and $\delta(q, B) = 2$. Then $\mathcal{A}$ accepts $\alpha \in \{A, B\}^\omega$ iff α contains infinitely many A's.



Now, after defining the automata model, we can start transforming MSO formulas into automata.

Proof of the second step of the Büchi's theorem. Firstly, using a little of combinatorics, one can show that given automata $\mathcal{A}_1, \mathcal{A}_2$, we can construct automata $\mathcal{A}_\cup, \mathcal{A}_\cap, \mathcal{A}_\neg$ with properties

- $L(\mathcal{A}_\cup) = L(\mathcal{A}_1) \cup L(\mathcal{A}_2)$,
- $L(\mathcal{A}_\cap) = L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$,
- $L(\mathcal{A}_\neg) = \Sigma^\omega \setminus L(\mathcal{A}_1)$.

Additionally, there are simple automata: one reading a word encoding $x \in \mathbb{N}, X \subseteq \mathbb{N}$ and checking whether $x \in X$, the second reading a word encoding $x, y \in \mathbb{N}$ and checking whether $x \leq y$ and the third one reading a word encoding $x \in \mathbb{N}$ and checking whether $A(x)$ for fixed $A \in \Sigma$.

So, using the above remarks, we can inductively transform the formula into the automaton, changing logic operators (like $\wedge$) into set theory operators (like $\cap$) and building more and more complicated automata. The only construction of MSO logic that remains is quantification.

Lemma 11. *For a given automaton $\mathcal{A}$ over alphabet $\Sigma \times \Gamma$, there exists an automaton $\mathcal{B}$ over Σ with a property*

$$L(\mathcal{B}) = \{\alpha \in \Sigma^\omega : \exists_{\beta \in \Gamma^\omega}\ (\alpha, \beta) \in L(\mathcal{A})\},$$

so the automaton recognising a projection of $L(\mathcal{A})$ to Σ^ω.

The proof of this lemma is very technical and involves a lot of tricky combinatorics. But after showing it, we can proceed as follows.

- Given a formula of the form $\exists_X\ \varphi(X)$ create an automaton $\mathcal{A}$ over alphabet $\Sigma \times 2$ such that $(\alpha, X) \in L(\mathcal{A})$ iff $\alpha \models \varphi(X)$. The construction of $\mathcal{A}$ is inductive over the structure of φ.
- Use the construction from Lemma 11 to construct the automaton $\mathcal{B}$ recognising $\pi(L(\mathcal{A}))$.
- Easily show that $L(\mathcal{B}) = L\left(\exists_X\ \varphi(X)\right)$.

Formulas starting with $\exists_x, \forall_x, \forall_X$ can be transformed into formulas of the form $\exists_X$ using syntactical tricks. This ends the proof. $\square$

The other transformation is also possible.

Theorem 12. *Given an automaton $\mathcal{A}$ one can construct a* WMSO *formula φ such that*

$$L(\varphi) = L(\mathcal{A}).$$

Moreover, one can construct a MSO *formula φ with the same property and additionally $\varphi = \exists_{\overline{X}}\ \psi(\overline{X})$ for some first order formula ψ.*

The proof is rather simple and we will skip it.

Remark 13. The family of ω-regular languages can be equivalently defined as a family of languages that:

- are definable by MSO formulas,
- are definable by WMSO formulas,
- are recognisable by deterministic parity automata,
- are recognisable by nondeterministic Büchi's automata.

More about the MSO logic and automata can be found in a paper [3].

4. Topology

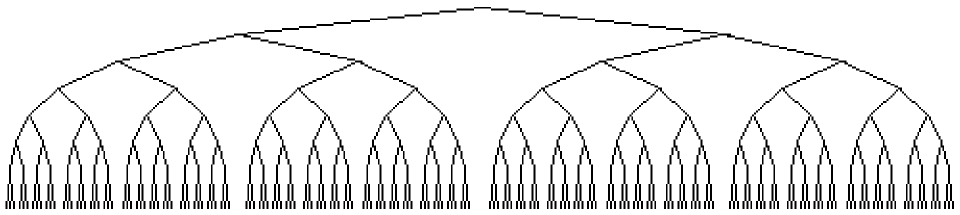
Since ω-regular languages are subsets of Σ^ω, there is a natural idea of using topological methods to investigate their complexity.

Definition 14. Consider a topology on Σ^ω generated by sets

$$[s] := \left\{\alpha \in \Sigma^\omega : \alpha|_{|s|} = s\right\},$$

for $s \in \Sigma^*$.

The topology of Σ^ω can be illustrated by the following picture, where elements of Σ^ω are infinite branches of the tree.



Fact 15. *For every Σ such that $2 \leq |\Sigma| < \infty$ the space Σ^ω is homeomorphic to the Cantor's discontinuum.*

In the following two subsections we will construct tools of measuring the complexity of subsets of a given topological space.

4.1. **The Borel hierarchy.** It is easy to see that the countable intersection of open sets may be no longer open. The same holds for sums of closed sets. Using this observation, we define a hierarchy of more and more complicated subsets of given topological space.

Definition 16. Inductively, for $\eta < \omega_1$, define

- $\mathbf{\Sigma}^0_1$ as a family of all open sets,
- $\mathbf{\Pi}^0_1$ as a family of all closed sets,
- $\mathbf{\Sigma}^0_\eta$ as a family of countable unions of sets from $\bigcup_{\tau<\eta} \mathbf{\Pi}^0_\tau$,
- $\mathbf{\Pi}^0_\eta$ as a family of complements of sets from $\mathbf{\Sigma}^0_\eta$,
- $\mathcal{BC}\left(\mathbf{\Sigma}^0_\eta\right)$ as the smallest boolean algebra containing $\mathbf{\Sigma}^0_\eta$,
- $\mathit{\Delta}^0_\eta = \mathbf{\Sigma}^0_\eta \cap \mathbf{\Pi}^0_\eta$.

The key property of this hierarchy is the fact that it is strict — every inclusion at this diagram is not an equality.

$$\begin{tikzcd}
\mathbf{\Sigma}^0_1 \arrow[dr, dash] & & \mathbf{\Sigma}^0_2 \arrow[dr, dash] & & \mathbf{\Sigma}^0_3 & \\
& \mathcal{BC}\left(\mathbf{\Sigma}^0_1\right) \text{ - } \mathbf{\Delta}^0_2 \arrow[ur, dash] \arrow[dr, dash] & & \mathcal{BC}\left(\mathbf{\Sigma}^0_2\right) \text{ - } \mathbf{\Delta}^0_3 \arrow[ur, dash] \arrow[dr, dash] & & \dots \\
\mathbf{\Pi}^0_1 \arrow[ur, dash] & & \mathbf{\Pi}^0_2 \arrow[ur, dash] & & \mathbf{\Pi}^0_3 &
\end{tikzcd}$$

Definition 17. The family $\mathcal{B} = \bigcup_{\eta<\omega_1} \boldsymbol{\Sigma}^0_\eta$ is called a family of Borel sets.

Fact 18. *The family $\mathcal{B}$ is an σ-algebra.*

4.2. **The projective hierarchy.** To go beyond the Borel hierarchy we have to consider more powerful constructions than countable unions and intersections. Since a countable union can be seen as a projection $\pi\colon X \times \mathbb{N} \to X$, the natural candidates are general projections $\pi\colon X \times Y \to X$.

Definition 19. A set $A \subseteq X$ is called *analytic* (den. $\boldsymbol{\Sigma}^1_1$) iff it is a projection of some Borel set in $X \times Y$. By an induction on $n < \omega$ we define

- $\boldsymbol{\Pi}^1_n$ as complements of sets from $\boldsymbol{\Sigma}^1_n$,
- $\boldsymbol{\Sigma}^1_{n+1}$ as projections of sets from $\boldsymbol{\Pi}^1_n$,
- $\mathit{\Delta}^1_n = \boldsymbol{\Sigma}^1_n \cap \boldsymbol{\Pi}^1_n$.

The same as before the hierarchy is strict. This time its length is ω.

$$\begin{tikzcd}
& \boldsymbol{\Sigma}^1_1 \arrow[dr, dash] & & \boldsymbol{\Sigma}^1_2 \arrow[dr, dash] & & \boldsymbol{\Sigma}^1_3 \arrow[dr, dash] & \\
\boldsymbol{\Delta}^1_1 \arrow[ur, dash] \arrow[dr, dash] & & \boldsymbol{\Delta}^1_2 \arrow[ur, dash] \arrow[dr, dash] & & \boldsymbol{\Delta}^1_3 \arrow[ur, dash] \arrow[dr, dash] & & \cdots \\
& \boldsymbol{\Pi}^1_1 \arrow[ur, dash] & & \boldsymbol{\Pi}^1_2 \arrow[ur, dash] & & \boldsymbol{\Pi}^1_3 \arrow[ur, dash] &
\end{tikzcd}$$

Additionally, the following strong result holds.

Theorem 20 (Souslin)**.** *Borel sets are exactly $\mathit{\Delta}^1_1$ sets, that is*

$$\mathcal{B} = \mathit{\Delta}^1_1 = \boldsymbol{\Sigma}^1_1 \cap \boldsymbol{\Pi}^1_1.$$

4.3. **Continuous reductions.** The families defined in the previous two sections will be called *topological complexity classes* and denoted by $\mathcal{C}$. We will say that a set $A \subseteq X$ has a complexity $\mathcal{C}$ iff $A \in \mathcal{C}$ and for each complexity class $D \subsetneq C$ we have $A \notin \mathcal{D}$.

The main tool for finding a complexity of a given set are so called continuous reductions.

Definition 21. A function $f\colon X \to Y$ is called a *continuous reduction* of $A \subseteq X$ to $B \subseteq Y$ iff f is continuous and $f^{-1}(B) = A$.

In other words, f is a reduction of A to B if to check whether given $x \in X$ belongs to A it is enough to check if $f(x) \in B$.

Definition 22. A set $B \subseteq X$ is called *hard* for a class $\mathcal{C}$ (or $\mathcal{C}$-hard) iff for every $A \in \mathcal{C}$ there exists continuous reduction of A to B.

If additionally $B \in \mathcal{C}$ then B is called *complete* in class $\mathcal{C}$ (or $\mathcal{C}$-complete).

Theorem 23. *There exist complete sets for every class of the form $\boldsymbol{\Sigma}^0_\eta, \boldsymbol{\Pi}^0_\eta, \boldsymbol{\Sigma}^1_n, \boldsymbol{\Pi}^1_n$.*

Example 24. The set $\{\alpha \in \{0,1\}^\omega : \exists_{i\in\mathbb{N}}\ \alpha_i = 1\}$ is $\mathbf{\Sigma}^0_1$-complete.

Moreover, for $i \in \mathbb{N}$ the set

$$\left\{\alpha \in 2^{\mathbb{N}^i} : \exists_{m_i}\forall_{m_{i-1}}\exists_{m_{i-2}}\dots\forall_{m_1}\ \alpha(m_i, m_{i-1}, \dots, m_1) = 1\right\}$$

is $\mathbf{\Sigma}^0_i$-complete.

The following fact is easy but powerful.

Fact 25. *Assume that $\mathcal{C} \subsetneq \mathcal{D}$ are topological complexity classes. Let $C \subseteq X$ satisfy $C \in \mathcal{C}$ and let $D \subseteq X$ be $\mathcal{D}$-complete.*

- *If D reduces to $E \subseteq X$ then E is $\mathcal{D}$-hard.*
- *If $B \subseteq X$ reduces to C then $B \in \mathcal{C}$.*
- *$D \notin \mathcal{C}$.*

5. Complexity of ω-regular languages

In this section we will investigate topological complexity of ω-languages and some extensions of this family.

The fundamental fact follows.

Fact 26. *Therefore, the topological complexity of ω-regular language is $\mathcal{BC}\left(\mathbf{\Sigma}^0_2\right)$.*

Proof. Deterministic automaton $\mathcal{A}$ induces a continuous reduction of $L(\mathcal{A})$ to the set

$$\mathcal{S} = \left\{\tau \in Q^\omega : \liminf_{n\to\infty} \Omega(\tau_n) \equiv 1 \mod 2\right\}.$$

It is easy to show that $\mathcal{S} \in \mathcal{BC}\left(\mathbf{\Sigma}^0_2\right)$. Therefore every ω-regular language is in $\mathcal{BC}\left(\mathbf{\Sigma}^0_2\right)$. Language defined in Example 10 is $\mathbf{\Sigma}^0_2$-complete, so a simple combination of this language and its complement gives us a set that is ω-regular but does not belong to $\mathbf{\Sigma}^0_2 \cup \mathbf{\Pi}^0_2$. □

Using this observation we can easily show that certain sets are not ω-regular languages.

Example 27. Language $\{a^{n_0}ba^{n_1}b\dots : \lim_{i\to\infty}\ n_i = \infty\}$ is $\mathbf{\Pi}^0_3$-complete, therefore it is not ω-regular.

It is possible to show the above fact using combinatorical methods, but this way seems to be more straightforward.

One should not directly connect topological complexity and decidability. The following two examples explain the difference.

Example 28. There are only countably many ω-regular languages. Therefore, there exist $\mathbf{\Sigma}^0_1$ sets that are not ω-regular.

So very simple (from topological point of view) sets may not be ω-regular.

Example 29. Fix some very complex set, e.g. $P \subseteq \Sigma^\omega$ that is $\mathbf{\Sigma}^1_{100}$-complete.

Consider a language $\mathcal{L}$ build up from one atom Φ and boolean operators $\vee, \wedge, \neg$. Let the semantics of Φ be as follows

$$\alpha \models \Phi \quad \Leftrightarrow \quad \alpha \in P.$$

Of course emptiness problem is decidable for $\mathcal{L}$ — complexity of P does not play any role here. But the formula $\Phi \in \mathcal{L}$ defines P, so a $\mathbf{\Sigma}^1_{100}$-complete set.

This example shows that there exist languages defining very complicated sets and still decidable.

More about connections of topology and infinite computations can be found in [4].

5.1. **Extensions of the** MSO **logic.**

Currently, various extensions of ω-regular languages are studied. We will concentrate on one of them.

There are natural properties not expressible in MSO logic, for example "the time between receiving and sending forward a message is bounded". Based on this observation one can add to standard MSO logic additional predicates to express such properties.

Definition 30. Formula $\mathrm{U}X.\varphi(X)$ holds iff

$$\forall_{n\in\mathbb{N}}\exists_{X\subset\omega}\ n \leq |X| < \infty \wedge \varphi(X).$$

Let $\mathrm{MSO}+\mathrm{U}$ denote logic MSO extended with U and analogously for $\mathrm{WMSO}+\mathrm{U}$.

The concept of $\mathrm{WMSO}+\mathrm{U}$ logic was studied by Mikołaj Bojańczyk. His paper [1] contains the following observations.

Fact 31. $\mathrm{WMSO}+\mathrm{U}$ *is a strict extension of* MSO*.*

Theorem 32. *Logic* $\mathrm{WMSO}+\mathrm{U}$ *is decidable over infinite words.*

The idea of the proof of this theorem also goes through the concept of automata. In this case these are deterministic max-automata. Such automaton is similar to the construction of deterministic parity automaton but additionally it is equipped with counters. Such counters are not read during a run. The acceptance condition is a boolean combination of statements "counter c_i is bounded during the run".

Fact 33. *Topological complexity of languages definable in* $\mathrm{WMSO}+\mathrm{U}$ *is* $\mathcal{BC}\left(\mathbf{\Sigma}^0_2\right)$ *— the same as of ω-regular languages.*

The natural question about analogous results for full $\mathrm{MSO}+\mathrm{U}$ still remains open. The most important of them is the following conjecture.

Conjecture 34. *Logic* $\mathrm{MSO}+\mathrm{U}$ *is decidable over infinite words.*

Currently there is no adequate automata model for $\mathrm{MSO}+\mathrm{U}$ known. One of the ideas how to treat this problem is to estimate the topological complexity of languages definable in $\mathrm{MSO}+\mathrm{U}$ and compare it with known automata models. The following theorem from [2] provides a lower estimation.

Theorem 35. *There exists* $\mathbf{\Sigma}^1_1$*-complete (e.g. non Borel) set definable in* $\mathrm{MSO}+\mathrm{U}$*.*

An easy corollary follows.

Corollary 36. *There is no nondeterministic Borel automata model catching full* $\mathrm{MSO}+\mathrm{U}$*.*

The following questions still remains open:

- is there any decidable automata model catching MSO + U,
- does such automata model has decidable emptiness problem,
- what is an exact topological complexity of MSO + U.

References

[1] M. Bojaczyk, *Weak MSO with the unbounding quantifier*, in STACS, 2009, pp. 159–170.

[2] S. Hummel, M. Skrzypczak, and S. Torunczyk, *On the topological complexity of MSO+U and related automata models*, in MFCS, 2010, pp. 429–440. Online: `http://www.mimuw.edu.pl/~mskrzypczak/dokumenty/`.

[3] W. Thomas, *Languages, automata and logics*, Technical Report 9607, Institut für Informatik und Praktische Mathematik, Christian-Albsechts-Universität, Kiel, Germany, 1996.

[4] W. Thomas and H. Lescow, *Logical specifications of infinite computations*, in REX School/Symposium, 1993, pp. 583–621.

Michał Skrzypczak, Insitute of Informatics, University of Warsaw, Banacha 2, 02-097 Warsaw (Poland)

E-mail address: `m.skrzypczak@mimuw.edu.pl`