Subsumption of concepts in $\mathcal{FL}_0$ for (cyclic) terminologies with respect to descriptive semantics is PSPACE-complete.

Yevgeny Kazakov and Hans de Nivelle
MPI für Informatik, Saarbrücken, Germany
E-mail: {ykazakov|nivelle}@mpi-sb.mpg.de

Abstract

We close the gap in the complexity classification of subsumption in the simple description logic $\mathcal{FL}_0$, which allows for conjunctions and universal value restriction only. We prove that the subsumption problem in $\mathcal{FL}_0$ is PSPACE-complete for descriptive semantics when cyclic definitions are allowed. Our proof uses automata theory and as a by-product we establish the PSPACE-completeness of a certain decision problem for regular languages.

1 Introduction

Cyclic terminologies in description logic (DL) appear in natural way when definitions are seen as constraints between concepts rather than abbreviations for compound concepts: $Even \doteq \forall successor.Odd$; $Odd \doteq \forall successor.Even$. Nebel in [10] has argued that reasoning in the presence of cyclic T-Boxes does not always agree with intuition when all interpretations satisfying the definitions are allowed. In order to capture the right meaning of cyclic definitions, he proposed three types of semantics: **(1)** the least fixed-point semantics (lfp), where the concepts are interpreted in the smallest possible way, **(2)** the greatest fixed-point semantics (gfp), where the concepts are interpreted as large as possible, and **(3)** the descriptive semantics, in which all models are allowed.

The other issue of cyclic terminologies is the impact on the complexity of reasoning tasks for DL: satisfiability of cyclic $\mathcal{ALC}$ T-Boxes is EXPTIME-complete, whereas it is merely in PSPACE for acyclic terminologies. In order to identify the possible sources of complexity, several sub-boolean (e.g. without full negation) description logics have been studied with respect to all three kinds of semantics introduced by Nebel [1, 2, 8, 3]. Baader in [1, 2] has considered a simple description logic $\mathcal{FL}_0$, which only allows for conjunctions and universal value restrictions. Despite the severe syntactical restrictions, $\mathcal{FL}_0$ is capable of expressing some nice definitions involving recursion: for example, the definition of a finite acyclic graph can be expressed in $\mathcal{FL}_0$ by means of the cyclic definition $Dag \doteq Node \sqcap \forall arc.Dag$ using the least fixed-point semantics.

Baader has also observed that subsumption of concepts in $\mathcal{FL}_0$ can be characterized by automata theory. For the two types of fixed-point semantics, he showed that subsumption is equivalent to inclusion of some regular languages associated with the terminologies. Therefore, the PSPACE-completeness of concept subsumption was a

Subsumption in $\mathcal{FL}_0$	Cyclic T-Boxes	Acyclic T-Boxes
descriptive semantics	in PSPACE [1, 2], **PSPACE-hard**	co-NP-complete [9]
lfp-semantics	PSPACE-complete [1, 2]	
gfp-semantics	PSPACE-complete [1, 2]	

Figure 1: Summary of known and new (in bold) results

consequence of the PSPACE-completeness of the inclusion problem for regular languages [4]. The subsumption problem with respect to descriptive semantics, however, was not characterized by means of language inclusions, but only reduced to the inclusion problem of ω-languages accepted by Büchi automata, from which a PSPACE upper bound for the problem has been obtained. Therefore, the question about the exact complexity of subsumption with respect to descriptive semantics remained open and the best known lower bound was co-NP, which is the complexity of subsumption for acyclic terminologies [9].

In this paper, we close this remaining gap in the complexity classification of subsumption in $\mathcal{FL}_0$ (Fig. 1). We introduce ω-automata with the *prefix acceptance condition* and reduce the PSPACE-complete problem of checking whether such an automaton accepts all words to the concept subsumption problem with respect to descriptive semantics in $\mathcal{FL}_0$. An ω-automaton with prefix acceptance condition accepts an infinite word iff somewhere in the run, it passes through an accepting state: what happens after passing the accepting state is not important, the automaton may even get stuck after it. The question whether such an automaton accepts all words can be elegantly reformulated in terms of regular languages as follows[1]: $L \cdot \Sigma^\omega \stackrel{?}{=} \Sigma^\omega$, in which L is the regular language accepted by the automaton. To the best of our knowledge, the complexity of this problem was not known. We prove PSPACE-completeness of this problem. Interestingly, for the two types of fixed-point semantics, PSPACE-hardness can be proven by the reduction from the problem $L \stackrel{?}{=} \Sigma^*$ by using similar techniques.

The paper is organized as follows. In Section 2, we give some formal definitions and introduce notation. In Section 3, we characterize the subsumption in $\mathcal{FL}_0$ for descriptive semantics and show that it is not easier than a certain automata-theoretic problem. In Section 4, this problem is recognized as a form of the universality problem for ω-automata with prefix acceptance condition and the proof of its PSPACE-completeness is presented. In Section 5, we draw some conclusions and point the directions for future work.

2 Formal Preliminaries

$\mathcal{FL}_0$ is a simple description logic, which allows for conjunctions and universal value restrictions only, therefore it is a sub-boolean logic. Formally, given a *signature* $\Sigma = (\mathcal{A}, \mathcal{R})$ consisting of *concept names* $\mathcal{A}$ and *role names* $\mathcal{R}$, the set of (*generalized*) *concepts* $\mathcal{C}_\Sigma$ of the description logic $\mathcal{FL}_0$ is defined by the grammar:

$$\mathcal{C}_\Sigma \ ::= \ A \mid C_1 \sqcap C_2 \mid \forall R.C$$

where the $A \in \mathcal{A}$ are usually called *atomic concepts*; C_1, C_2, C are arbitrary generalized concepts of $\mathcal{FL}_0$, and $R \in \mathcal{R}$.

[1]This formulation was suggested by Franz Baader (private communications)

A *terminology* (or $TBox$ for short) is a finite set of *concept definitions* of the form $A \doteq C$, where A is an atomic concept called *defined concept* and C is a generalized concept. Every atomic concept can be defined no more than once in $TBox$.

Since we are interested in proving a hardness result for the concept subsumption problem, it is possible to consider any *restricted form of terminologies*. In the rest of the paper we will do this and assume that each $TBox$ $\mathcal{T}$ contains only definitions of the form:

$$A_i \doteq \forall R_{i,1}.B_{i,1} \sqcap \cdots \sqcap \forall R_{i,k_i}.B_{i,k_i}, \tag{1}$$

where the A_i, $B_{i,j}$ are atomic concepts $(1 \leq i \leq |\mathcal{T}|)$, $(1 \leq j \leq k_i)$, and each $k_i \geq 1$. We also assume that *every atomic concept has a definition* in $TBox$.

With every terminology $\mathcal{T}$ of the form (1), we associate a *non-deterministic semi-automaton* $\mathcal{A}_\mathcal{T} = (\Sigma, Q, \delta)$, called *the description graph*. Here Σ is a *finite alphabet of letters*, Q is a finite *set of states*, and $\delta \subseteq Q \times \Sigma \times Q$ is the *transition relation*. We proceed similarly as in [2, 10]:

- ⋆ the alphabet Σ of $\mathcal{A}_\mathcal{T}$ is the set of role names of $\mathcal{T}$;
- ⋆ the set of states Q is the set of concept names in $\mathcal{T}$ and
- ⋆ the transition relation $\delta = \{(A, R, B) \mid A \doteq \cdots \sqcap \forall R.B \sqcap \cdots \in \mathcal{T}\}$.

Note that this construction gives a one-to-one correspondence between terminologies of the form (1) and semi-automata without *blocking states*: for every state $q \in Q$ there exist some $a \in \Sigma$ and $q' \in Q$ such that $(q, a, q') \in \delta$.

A *run* of a semi-automaton $\mathcal{A} = (\Sigma, Q, \delta)$ over a finite (or infinite) *word* $w = a_1 \cdot a_2 \cdots a_i(\cdots) \in \Sigma^*(\Sigma^\omega)$ is a sequence of states $r: q_0, q_1, \ldots, q_i, (\ldots) \in Q^*(Q^\omega)$ such that $(q_{i-1}, a_i, q_i) \in \delta$ for every $i \geq 1$. With every pair of states $q_1, q_2 \in Q$ of a semi-automaton $\mathcal{A} = (\Sigma, Q, \delta)$ one can associate the *regular language* $L_\mathcal{A}(q_1, q_2) := \{w \in \Sigma^* \mid$ there exists a run $q_1, \ldots, q_2$ over $w\}$.

3 Subsumption for Descriptive Semantics

In this section we characterize the subsumption problem for descriptive semantics in $\mathcal{FL}_0$ using automata theory. Such a characterization was already given in [2], however it can be simplified for the restricted form of terminologies (1):

Theorem 1 (Characterization of concept subsumption) *Let $\mathcal{T}$ be a terminology of form (1) and $\mathcal{A}_\mathcal{T} = (\Sigma, Q, \delta)$ be the corresponding description graph. Then*
$A_0 \sqsubseteq_\mathcal{T} B_0$ *iff* *for every word $w \in \Sigma^\omega$ and for every run*
$r_B: B_0, B_1, \ldots, B_i, \ldots$ *in $\mathcal{A}_\mathcal{T}$ over w, there exist a run*
$r_A: A_0, A_1, \ldots, A_i, \ldots$ *in $\mathcal{A}_\mathcal{T}$ over w and an integer $k \geq 0$ such that $A_k = B_k$.*

Proof. The theorem follows from the more general characterization given in [2] (Theorem 29). The proof of the theorem can be also found in our technical report [7], where we present the extended version of the paper. □

Now we show how to translate a certain intermediate problem for semi-automata into the concept subsumption problem with descriptive semantics for $\mathcal{FL}_0$. We give an instance of the concept subsumption problem which suffices to prove PSPACE-hardness. Take a semi-automaton $\mathcal{A} = (\Sigma, Q, \delta)$ and two states $q_1, q_2 \in Q$. We construct a new semi-automaton $\mathcal{A}'$ from $\mathcal{A}$ by adding a new state q' and making it

reachable from q_2 and itself through every symbol of Σ : So we have $\mathcal{A}' = (\Sigma, Q', \delta')$, where $Q' = Q \cup q'$ and $\delta' = \delta \cup \{(q_2, a, q'),\ (q', a, q') \mid a \in \Sigma\}$.

If $\mathcal{A}'$ does not have blocking states then we can consider the terminology $\mathcal{T}'$ corresponding to $\mathcal{A}'$, so q_1 corresponds to some concept A of $\mathcal{T}'$ and q' corresponds to some concept B of $\mathcal{T}'$. By Theorem 1, B subsumes A *iff* for every run from q' over some word $w \in \Sigma^\omega$ there exists a run from q_1 over w such that both runs share at least one state. Since every run from q' can contain the state q' only, and for every $w \in \Sigma^\omega$ such a run always exists, we obtain: *B subsumes A iff for every $w \in \Sigma^\omega$ there exists a run over w from q_1 containing q'.* Note that in the last sentence we can replace q' by q_2. Thus our construction proves the following lemma:

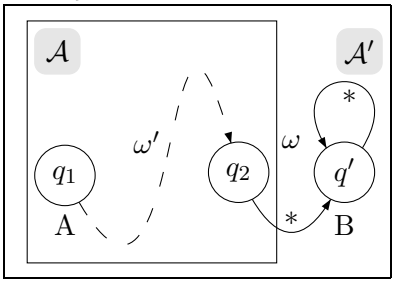

Figure 2: The reduction

Lemma 2 (The reduction lemma) *The concept subsumption problem is not easier than the problem: Given a semi-automaton $\mathcal{A} = (\Sigma, Q, \delta)$ and two states $q_1, q_2 \in Q$ such that $Q \setminus \{q_2\}$ has no blocking states, check whether every word $w \in \Sigma^\omega$ has a finite prefix $w' \in L_{\mathcal{A}}(q_1, q_2)$.*

In the next section we reformulate this problem in terms of automata on infinite words as the *universality problem* and prove that it is PSPACE-complete.

4 Automata and the Universality Problem

The subsumption problem for $\mathcal{FL}_0$ using the two types of fixed-point semantics was characterized in [1, 2] by the inclusion problem of some regular languages associated with the description graph. It is well-known that this problem is PSPACE-complete [4]. However, we have found no way of characterizing the descriptive semantics in a similar way. Therefore, we study the problem formulated in Lemma 2 directly.

A *non-deterministic finite automaton* (**NFA**) is a tuple $\mathcal{A} = (\Sigma, Q, \delta, Q_0, F)$, which is a semi-automaton (Σ, Q, δ) extended with a set of *initial states* $Q_0 \subseteq Q$ and a set of *accepting states* $F \subseteq Q$. The *size* of the automaton $\mathcal{A} = (\Sigma, Q, \delta, Q_0, F)$ is $|\mathcal{A}| = |Q| + |\delta|$. We distinguish several kinds of non-deterministic finite automata according to acceptance conditions:

1. An *automaton on finite words* $\mathbf{NFA}^*$ is an **NFA** $(\Sigma, Q, \delta, Q_0, F)$ which accepts a finite word $w \in \Sigma^*$ *iff* there exists a run $r:\ q_1, \ldots, q_n$ over w with $q_1 \in Q_0$, $q_n \in F$.

2. A *Büchi automaton* $\mathbf{NFA}^\omega_b$ is an **NFA** $(\Sigma, Q, \delta, Q_0, F)$ on *infinite* words. It accepts $w \in \Sigma^\omega$ *iff* there exists a run $r:\ q_1, \ldots, q_i, \ldots$ over w which repeats some state from F *infinitely often*.

3. We introduce the ω-automaton with the *prefix acceptance condition* $\mathbf{NFA}^\omega_p$ as an **NFA** $(\Sigma, Q, \delta, Q_0, F)$, which accepts $w \in \Sigma^\omega$ *iff* there exist a *finite prefix* w' of w and a run $r:\ q_1, \ldots, q_n$ over w' with $q_1 \in Q_0$ and $q_n \in F$. In other words, $\mathbf{NFA}^\omega_p$ accepts an infinite word if it accepts a finite prefix of this word as $\mathbf{NFA}^*$.

According to Lemma 2 the concept subsumption problem is not easier than a certain problem for semi-automata $\mathcal{A} = (\Sigma, Q, \delta)$. Observe that this problem can be formulated in terms of automata with prefix acceptance condition as: *given* $\mathbf{NFA}^\omega_p =$

$(\Sigma, Q, \delta, \{q_1\}, \{q_2\})$ *without blocking states in* $Q \setminus \{q_2\}$*, check whether all words* $w \in \Sigma^\omega$ *are accepted.* This problem appears in the literature as the (non)universality problem for finite automata [12]. The $\mathbf{NFA}^*$ ($\mathbf{NFA}^\omega_b$, $\mathbf{NFA}^\omega_p$) is *universal* iff it accepts every word $w \in \Sigma^*$ ($w \in \Sigma^\omega$). The associated decision problem is called the *universality problem.* This problem is known to be PSPACE-complete for $\mathbf{NFA}^*$ and $\mathbf{NFA}^\omega_b$ (cf. [12]). It is not surprising that we can obtain a similar result for the $\mathbf{NFA}^\omega_p$.

Theorem 3 *The universality problem for* $\mathbf{NFA}^\omega_p$ *is in PSPACE.*

Proof. The proof is by a reduction to the universality problem for Büchi automata. Given $\mathbf{NFA}^\omega_p$ $\mathcal{A} = (\Sigma, Q, \delta, Q_0, F)$ we proceed similarly as in Section 3: Consider the Büchi automaton $\mathcal{A}' = (\Sigma, Q', \delta', Q_0, \{q'\})$, where q' is a new state, $Q' = Q \cup \{q'\}$ and $\delta' = \delta \cup \{(q, a, q') \ , (q', a, q') \mid q \in F, \ a \in \Sigma\}$. $\mathcal{A}$ accepts $w \in \Sigma^\omega$ *iff* $\mathcal{A}'$ does, so $\mathcal{A}$ is universal *iff* $\mathcal{A}'$ is universal. □

Theorem 4 *The universality problem for* $\mathbf{NFA}^\omega_p$ *is PSPACE-hard.*

Proof. The proof is given by a reduction from polynomial-space Turing machines. The idea is quite standard for proving such results [6]. For every Turing machine and input we construct an automaton which accepts every word except the legal computation of the Turing machine: given some candidate word it "guesses" the position of the possible error and accepts the word if there is indeed an error. So the constructed automaton is universal *iff* the Turing machine does not accept the input. The details of the proof can be found in Appendix A. □

Corollary 5 *The universality problem for* $\mathbf{NFA}^\omega_p$ *is PSPACE-complete.*

As a consequence, we have now shown the complexity of the problem for regular languages that is mentioned in the introduction:

Corollary 6 *The following problem is PSPACE-complete: Given an* $\mathbf{NFA}^*$ $\mathcal{A} = (\Sigma, Q, \delta, Q_0, F)$ *check whether for the set* $L \subseteq \Sigma^*$ *of words accepted by the automaton, the property* $L \cdot \Sigma^\omega = \Sigma^\omega$ *holds.*

We have proven the PSPACE-hardness of the universality problem for $\mathbf{NFA}^\omega_p$ $\mathcal{A} = (\Sigma, Q, \delta, Q_0, F)$; however, according to the reduction Lemma 2 we need to prove the hardness for the more restricted case where we have only one initial state, one accepting state and do not have blocking states among the non-accepting states. The next proposition shows that we can assume these restrictions without loss of generality.

Proposition 7 *For any* $\mathbf{NFA}^\omega_p$ $\mathcal{A} = (\Sigma, Q, \delta, Q_0, F)$ *one can construct an* $\mathbf{NFA}^\omega_p$ $\mathcal{A}' = (\Sigma, Q', \delta', \{q'_0\}, \{f'\})$ *without blocking states in* $Q' \setminus \{f'\}$ *in linear size of* $|\mathcal{A}|$ *which accepts exactly the same words as* $\mathcal{A}$*.*

Proof. We consider two cases:

1. $Q_0 \cap F \neq \emptyset$. Then $\mathcal{A}$ trivially accepts all words and we can take for example $A' := \{\Sigma, \{q\}, \emptyset, \{q\}, \{q\}\}$ for some state q.

2. $Q_0 \cap F = \emptyset$. We simply take $\mathcal{A}' = (\Sigma, Q', \delta', \{q'_0\}, \{f'\})$ for two new states q'_0 and f' with $Q' = Q \cup \{q'_0, \ f'\}$, and define

$$\delta' = \delta \cup \{(q'_0, a, q) \mid \exists q_0 \in Q_0 : (q_0, a, q) \in \delta\} \cup \{(q, a, f') \mid \exists f \in F : (q, a, f) \in \delta\}$$
$$\cup \{(q'_0, a, f') \mid \exists q_0 \in Q_0, \exists f \in F : (q, a, f) \in \delta\}.$$

If some state $q' \in Q' \setminus \{f'\}$ is blocking then we can remove it together with the transitions involved since no run from q'_0 to f' can contain q'. □

Corollary 8 *The concept subsumption problem is not easier than the universality problem for* $\mathbf{NFA}_p^\omega$.

Corollary 9 *The concept subsumption problem for DL* $\mathcal{FL}_0$ *with cyclic terminologies w.r.t. descriptive semantics is PSPACE-complete.*

5 Conclusions and Related Work

The results obtained in this paper confirm the relationship between description logics and automata theory. Certainly there is a correspondence between semantics for terminological cycles and acceptance conditions for automata which requires further investigation. We hope that automata theory can provide a useful tool for complexity analysis in description logics and for the development of practical algorithms.

In last years sub-boolean description logics and modal formalisms have drawn more attention. Recently Baader in [3] has shown that concept subsumption in the description logic $\mathcal{EL}$ allowing for only conjunctions and existential value restriction is *polynomial* for fixed-point and descriptive semantics. He gave a classification of subsumption in $\mathcal{EL}$ for all three kinds of semantics in terms of graph simulations, similar as has been given for $\mathcal{FL}_0$. Understanding the sources of complexity in DL-fragments for particular reasoning tasks would be of great value for the DL community.

It would be particularly interesting to study cyclic terminologies with mixed (fixed-point and descriptive) semantics considering the syntactical variants of modal mu-calculus. In the recent paper [5] Henzinger et. al. have studied the fragments of mu-calculus allowing only one type of quantifiers: so-called universal $\forall MC$ and existential fragments $\exists MC$. Among other results, they proved that the satisfiability problems of these fragments are easier than for the full mu-calculus: $\forall MC$ is PSPACE-complete and $\exists MC$ is NP-complete. However, the complexity of the implication problem (which is related to the subsumption problem in DL) is still EXPTIME, as for the full mu-calculus. To obtain fragments with an easy implication problem, one probably needs to restrict the use of boolean connectives in similar ways as in $\mathcal{FL}_0$ and $\mathcal{EL}$.

Acknowledgements

The authors would like to thank Franz Baader, Viorica Sofronie-Stokkermans and anonymous referees for reading the draft of the paper and giving valuable suggestions and remarks.

References

[1] F. Baader. Terminological cycles in KL-ONE-based knowledge representation languages. In *Proc. of AAAI-90*, pages 621–626, Boston, MA, 1990.

[2] F. Baader. Using automata theory for characterizing the semantics of terminological cycles. *Annals of Mathematics and Artificial Intelligence*, 18(2–4):175–219, 1996.

[3] F. Baader. Terminological cycles in a description logic with existential restrictions. LTCS-Report LTCS-02-02, Germany, 2002.

[4] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco, 1979.

[5] T. A. Henzinger, O. Kupferman, and R. Majumdar. On the universal and existential fragments of the mu-calculus. In *Proceedings of the Ninth International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, Lecture Notes in Computer Science. Springer-Verlag, 2003.

[6] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, Reading, Mass., 1979.

[7] Y. Kazakov and H. Nivelle. Subsumption of concepts in $DL\ \mathcal{FL}_0$ for (cyclic) terminologies with respect to descriptive semantics is PSPACE-complete. Research Report MPI-I-2003-2-003, Max-Planck-Institut für Informatik, April 2003. See http://domino.mpi-sb.mpg.de/internet/reports.nsf.

[8] R. Küsters. Characterizing the Semantics of Terminological Cycles in $\mathcal{ALN}$ using Finite Automata. In *Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR'98)*, pages 499–510. Morgan Kaufmann, 1998.

[9] B. Nebel. Terminological reasoning is inherently intractable. *Artificial Intelligence*, 43:235–249, 1990.

[10] B. Nebel. Terminological cycles: Semantics and computational properties. In J. F. Sowa, editor, *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, pages 331–361. Morgan Kaufmann Publishers, San Mateo (CA), USA, 1991.

[11] M. Sipser. *Introduction to the Theory of Computation*. International Thomson Publishing, 1996.

[12] M. Y. Vardi. An automata-theoretic approach to linear temporal logic. In *Banff Higher Order Workshop*, pages 238–266, 1995.

Appendix A.

In this appendix we give a proof of Theorem 4.

Theorem 4 *The universality problem for* $\mathbf{NFA}_p^\omega$ *is PSPACE-hard.*

Proof. We prove the theorem by a reduction from polynomial-space Turing machines. The details of definitions involved can be found, for instance, in [11], however in order to be self-contained, we give the ones that are needed.

A *Turing machine* is a tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$, where Q is the finite *set of states*, Σ is the finite *input alphabet*, Γ is the finite *tape alphabet* containing the special *blank symbol* $\sqcup$ (it must be that $\Sigma \subseteq \Gamma \setminus \{\sqcup\}$), $\delta : Q \times \Gamma \to Q \times \Gamma \times \{-1, +1\}$ is the *transition function*, $q_0 \in Q$ is the *initial state*, $q_{accept} \in Q$ is the accepting state and $q_{reject} \in Q$ ($q_{reject} \neq q_{accept}$) is the rejecting state.

A *configuration* of the Turing machine $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ is a string of the form: $c = a_1\ a_2\ \ldots\ a_{i-1}\ q\ a_i\ \ldots\ a_k$, where each $a_j \in \Gamma$, $q \in Q$, $k \geq i \geq 1$. One could think of the configuration c as the description of the Turing machine in the state q with the head at the i-th cell of the tape, and the tape having content $a_1 \cdots a_k$.

The transition function δ can be extended to configurations in the following way: Let $a, b \in \Gamma$, $u, v \in \Gamma^*$. Let $[c]$ be obtained from the configuration c by removing the rightmost blank symbols $\sqcup$ from c. Then define
$\hat{\delta}(uaq_ibv) := uq_jacv, \quad \hat{\delta}(q_ibv) := q_jcv \quad \text{if} \quad \delta(q_i, b) = (q_j, c, -1);$
$\hat{\delta}(uaq_ibv) := uacq_jv, \quad \hat{\delta}(q_ibv) := cq_jv \quad \text{if} \quad \delta(q_i, b) = (q_j, c, +1); \quad \hat{\delta}(uq_i) := [\hat{\delta}(uq_i\sqcup)].$

A *computation* of the Turing machine $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ *from* $x \in \Sigma^*$ is a sequence of configurations $c_0, c_1, \ldots, c_i, \ldots$ such that $c_0 = q_0x$ and $c_{i+1} = \hat{\delta}(c_i)$. If the computation ends with a configuration c_n then if $q_{accept} \in c_n$, we say that M *accepts* x; if $q_{reject} \in c_n$, we say that M *rejects* x.

The Turing machine M *decides* the language $L \subseteq \Sigma^*$ if for every $x \in \Sigma^*$, $x \in L$ implies M accepts x, and $x \notin L$ implies M rejects x. We say that M is a *polynomial-space* Turing machine if there exists a polynomial $p(n)$ such that for every input $x \in \Sigma^*$ with computation $q_0x = c_0, c_1, \ldots, c_i, \ldots$, for each c_i we have $|c_i| \leq p(|x|)$. PSPACE is defined as a class of all languages decided by polynomial-space Turing machines.

Now we give a polynomial-time reduction from the decision problem for an arbitrary language $L \in$ PSPACE to the universality problem for some set of $\mathbf{NFA}_p^\omega$. Assume $M = (Q, \Sigma, \Gamma, q_0, q_{accept}, q_{reject})$ is a polynomial-space Turing machine that decides L. We give an algorithm which for every $x \in \Sigma^*$ constructs an $\mathbf{NFA}_p^\omega$ $\mathcal{A}_x$ in polynomial size of $|x|$ such that $\mathcal{A}_x$ accepts all words, except the word:

$$w_{rej} = \#\cdot\ \ \#\cdot c_0\cdot(\sqcup)^{l_0}\cdot\quad \#\cdot c_1\cdot(\sqcup)^{l_1}\cdot\ \ldots\ \ \#\cdot c_{k-1}\cdot(\sqcup)^{l_{k-1}}\cdot\ \ (\#\cdot c_k\cdot(\sqcup)^{l_k})^\omega$$

where $c_0, c_1, \ldots, c_k$ is the accepting computation for x (if it exists); $l_i = p(|x|) - |c_i|$. All configurations of M are padded with $\sqcup$'s so that the resulting strings have equal length $P = p(|x|)$. This simplifies the upcoming construction. Symbol $\#$ is a new symbol ($\# \notin \Gamma$). Thus, $x \in \overline{L}$ *iff* M rejects x *iff* M does not accept x *iff* A_x is universal, and we can obtain the reduction since PSPACE = co-PSPACE.

Consider the word w_{rej}. Every three subsequent symbols $\sigma_{i-1}, \sigma_i, \sigma_{i+1}$ at positions $i-1, i, i+1$ of w_{rej} uniquely determine the symbol σ_{i+P+1} at position $i+P+1$ of w_{rej}. To be precise, $\sigma_{i+P+1} = N(\sigma_{i-1}, \sigma_i, \sigma_{i+1})$, where:

$N(q_i, a, \sigma) = c, \ \ N(b, q_i, a) = b, \ \ N(\#, q_i, a) = N(\sigma, b, q_i) = q_j \ \ \text{if } \delta(q_i, a) = (q_j, c, -1);$

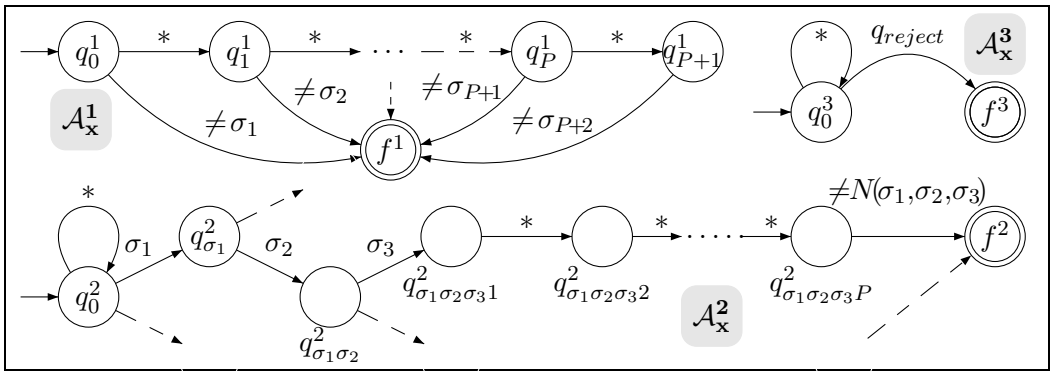

Figure 3: The automata, whose union accepts all non-computations.

$N(q_i,a,\sigma)=q_j,\ \ N(\sigma,q_i,a)=c,\ \ N(\sigma,b,q_i)=b$ if $\delta(q_i,a)=(q_j,c,+1)$;
$N(\sigma_1,\sigma_2,\sigma_3)=\sigma_2$ in all other cases; $(a,b,c\in\Gamma,\ \sigma,\sigma_i\in\Gamma\cup\{\#\})$.

The informal description of $\mathcal{A}_x$ is as follows: given an infinite string $w\in(\Gamma\cup\{\#\})^\omega$, $\mathcal{A}_x$ accepts w if it can find that $w\neq w_{rej}$, which can be done by detecting one of the following:

1. *The first $P+2$ symbols of w differ from those of $\#\cdot\#\cdot x\cdot(\sqcup)^{l_0}$, $(l_0=P-|x|)$;*
2. *For some $i\geq 2$ the symbol $\sigma_{i+P+1}\neq N(\sigma_{i-1},\sigma_i,\sigma_{i+1})$;*
3. *The string w contains the symbol q_{reject}.*

Note that since the Turing machine M decides the language $L\subseteq\Sigma^*$, every computation $c_0,c_1,\ldots,c_i,\ldots$ from $x\in\Sigma^*$ should end either with the accepting state q_{accept} or with the rejecting state q_{reject}. So, $w\neq w_{rej}$ *iff* w satisfies one of the 1-3 above.

Formally, $\mathcal{A}_x=\mathcal{A}_x^1\cup\mathcal{A}_x^2\cup\mathcal{A}_x^3$ where: $\mathcal{A}_x^i$ is the $\mathbf{NFA}_p^\omega$ over $(\Gamma\cup\{\#\})^w$ which accepts a word w iff the corresponding condition i above is fulfilled $(i=1,2,3)$. The *union* of two automata $\mathcal{A}^1=(\Sigma,Q^1,\delta^1,Q_0^1,F^1)$ and $\mathcal{A}^2=(\Sigma,Q^2,\delta^2,Q_0^2,F^2)$ is the automaton $\mathcal{A}=(\Sigma,Q^1\cup Q^2,\delta^1\cup\delta^2,Q_0^1\cup Q_0^2,F^1\cup F^2)$. $\mathcal{A}$ accepts a word *iff* it is accepted by $\mathcal{A}^1$ or $\mathcal{A}^2$. The automata $\mathcal{A}_x^1$, $\mathcal{A}_x^2$ and $\mathcal{A}_x^3$ are constructed as follows (see Fig. 3):

1. $\mathcal{A}_x^1=(\Gamma\cup\{\#\},Q^1,\delta^1,\{q_0^1\},\{f^1\})$, where $Q^1=\{q_0^1,q_1^1,\ldots,q_{P+1}^1,f^1\}$;
   $\delta^1=\{(q_i^1,\sigma,q_{i+1}^1)\mid\sigma\in\Gamma,\ 0\leq i\leq P\}\cup$
   $\{(q_i^1,\sigma,f^1)\mid 0\leq i\leq P{+}1,\ \sigma\neq\sigma_{i+1}:=(i{+}1)\text{-th element of }\#\cdot\#\cdot x\cdot(\sqcup)^{l_0}\}$

2. $\mathcal{A}_x^2=(\Gamma\cup\{\#\},Q^2,\delta^2,\{q_0^2\},\{f^2\})$, where
   $Q^2=\{q_0^2,\ q_{\sigma_1}^2,\ q_{\sigma_1\sigma_2}^2,\ q_{\sigma_1\sigma_2\sigma_3 i}^2,\ f^2\mid\sigma_1,\sigma_2,\sigma_3\in\Gamma\cup\{\#\};\ 1\leq i\leq P\}$;
   $\delta^2=\{(q_0^2,\sigma_1,q_0^2),\ (q_0^2,\sigma_1,q_{\sigma_1}^2),\ (q_{\sigma_1}^2,\sigma_2,q_{\sigma_1\sigma_2}^2),\ (q_{\sigma_1\sigma_2}^2,\sigma_3,q_{\sigma_1\sigma_2\sigma_3 1}^2),$
   $(q_{\sigma_1\sigma_2\sigma_3 i}^2,\sigma,q_{\sigma_1\sigma_2\sigma_3(i+1)}^2)\mid\sigma,\sigma_1,\sigma_2,\sigma_3\in\Gamma\cup\{\#\};\ 1\leq i<P\}\cup$
   $\{(\sigma_{\sigma_1\sigma_2\sigma_3 P}^2,\sigma,f^2)\mid\sigma,\sigma_1,\sigma_2,\sigma_3\in\Gamma\cup\{\#\};\ \sigma\neq N(\sigma_1,\sigma_2,\sigma_3)\}$

3. $\mathcal{A}_x^3=(\Gamma\cup\{\#\},Q^3,\delta^3,\{q_0^3\},\{f^3\})$, where $Q^3=\{q_0^3,f^3\}$;
   $\delta^3=\{(q_0^3,\sigma,q_0^3),\ (q_0^3,q_{reject},f^3)\mid\sigma\in\Gamma\cup\{\#\}\}$

The sizes of automata $\mathcal{A}_x^1$, $\mathcal{A}_x^2$ and $\mathcal{A}_x^3$ are linear in $P=p(|x|)$ (Γ is fixed). Therefore, the construction of A_x can be performed in polynomial time relative to $|x|$.

To summarize, we have constructed a polynomial time reduction from any language $L\in$ PSPACE to the universality problem for $\mathbf{NFA}_p^\omega$ and thus, have proven its PSPACE-hardness. $\square$