On the Decidability of Finding a Positive ILP-Instance in a Regular Set of ILP-Instances

Petra Wolf[✉]

FB 4 - Abteilung Informatikwissenschaften, Universität Trier, Trier, Germany
wolfp@uni-trier.de

Abstract. The regular intersection emptiness problem for a decision problem P ($int_{\text{Reg}}(P)$) is to decide whether a potentially infinite regular set of encoded P-instances contains a positive one. Since $int_{\text{Reg}}(P)$ is decidable for some NP-complete problems and undecidable for others, its investigation provides insights in the nature of NP-complete problems. Moreover, the decidability of the int_{Reg}-problem is usually achieved by exploiting the regularity of the set of instances; thus, it also establishes a connection to formal language and automata theory. We consider the int_{Reg}-problem for the well-known NP-complete problem Integer Linear Programming (ILP). It is shown that any DFA that describes a set of ILP-instances (in a natural encoding) can be reduced to a finite core of instances that contains a positive one if and only if the original set of instances did. This result yields the decidability of int_{Reg}(ILP).

Keywords: Deterministic finite automaton · Regular languages · Regular intersection emptiness problem · Decidability · Integer linear programming

1 Introduction

The problem Integer Linear Programming (ILP for short) asks whether a given set of inequalities with integer coefficients has an integer solution. ILP is among the first problems for which NP-hardness was shown (it is on Karp's original list of 21 NP-complete problems) and it is of great practical relevance in mathematical optimisation. There is a large number of academic prototypes as well as commercial implementations of ILP-solvers that are applied in various contexts; therefore ILP is arguably of similar importance as the well-known Boolean satisfiability problem. For recent theoretical papers on ILP see, e. g., [5,7].

Linear and Integer Linear Programs are often used to model observations of the real world under the assumption that some properties are present. Important fields of applications are for example image segmentation [12] and motion

M. Hospodár et al. (Eds.): DCFS 2019, LNCS 11612, pp. 272–284, 2019.
https://doi.org/10.1007/978-3-030-23247-4_21

segmentation [13]. These models often face uncertainties due to lack of information or measurement errors [10]. One possibility to handle this problem is to take every possible instance into account, in which the uncertainty is replaced by an actual value and ask whether one of them is solvable. In doing so, we get a potentially *infinite* set of instances under which we seek a solvable one. For example, suppose we have a system of two inequalities $a_{11}x_1 + a_{12}x_2 \leq b_1$ and $a_{21}x_1a_{22}x_2 \leq b_2$ with two integer variables x_1 and x_2 and only partial knowledge of the coefficients $a_{11}, a_{12}, b_1, a_{21}, a_{22}, b_2$. Due to measurement inaccuracies all we know is that a_{11} is a power of 2; a_{12} is even and negative; b_1 is positive and less than 100; a_{21} is congruent to 3 modulo 29; a_{22} is 1 less than an odd power of 2; and b_2 is negative. The described inequalities form an infinite family of inequalities and the described system represents an infinite family S of instances of ILP. Since each coefficient fits a regular pattern, a DFA can describe the encodings of exactly the instances in S.

Compact representations of finite sets of instances have already been considered for other problems. In graph modification,[1] the task is to transform a given graph using a given set of edit operations into a graph of a certain graph-family using as few operations as possible [2,14]. The possible edit operations give rise to an edit distance [8] with respect to the set of graphs; thus, the above described task can be seen as checking whether the set of all graphs within a certain distance from the given graph contains a member of the specified graph-family. The same can be done for string-problems where a given string is to be transformed (by using certain operations) into a target string [4].

Searching for a positive instance among infinitely many instances of a problem P seems to be a natural generalization of this setting. If we consider regular sets of instances, this task can be formalised as checking whether a given regular language of P-instances (represented by a deterministic finite automaton) and the fixed language of positive P-instances have a non-empty intersection. This was the original viewpoint of the line of research introduced by Güler et al. [9,23], where this problem is called the $\mathit{int}_{\mathrm{Reg}}$*-problem of* P (or $\mathit{int}_{\mathrm{Reg}}(P)$ for short).[2]

The $\mathit{int}_{\mathrm{Reg}}$-problem has independently been studied under the name *regular realizability problem* $RR(L)$, where the *filter language* L plays the role of problem P as defined above, i. e., $RR(L) = \mathit{int}_{\mathrm{Reg}}(L)$ (see [1,15–17,19–21]). The RR problem appeared when considering *models of generalized nondeterminism* (GNA) where an auxiliary memory is used as a source of nondeterminism [18]. For each GNA class there are complete $RR(L)$ problems where the filter language L consists of prefixes of GNA-certificates (or guess words) [19]. That fact already gives RR-problems which are complete under log space reductions for LOG, NLOG, P, NP, PSPACE, EXP, and Σ_1. This observation motivated the attempt to present with the RR-problem 'a specific class of algorithmic problems that represents complexities of all known complexity classes [...] in a unified way' [20]. It turned out that RR-problems are universal in the sense that for any prob-

[1] A Dagstuhl seminar on 'Graph Modification Problems' was held in 2014 [2].

[2] Note that this problem is only well-defined if it is clear how P is represented as a language, i. e., we have to define how P-instances are encoded as strings.

lem P, there exists an RR-problem $RR(L)$ with the same complexity (note that P and L are different languages). In [21], instead of focusing on which complexity classes can be covered by an RR-problem, the authors concentrate on context-free filter languages and present examples for which $RR(L)$ is either P-complete, NLOG-complete or has an intermediate complexity. In [17] the decidability of the RR-problem with languages of permutations of binary words as filters have been considered. In this line of research, the filter languages are closely related to computations of specific machine models. As a consequence the regularity of the input language is not exploited at all and the hard part of a problem is coded into regular languages consisting of single words only. In [20] the author notes that the presented reductions 'cut off almost all properties of regular languages'.

In [1], $int_{\mathrm{Reg}}(L)$ has been studied for L with low computational complexity, but which describe structural properties of words that have high relevance for combinatorics on words and formal language theory (e.g., set of primitive words, palindromes, etc.). In this regards, (efficient) decision procedures are obtained.

In contrast to these research questions, the line of work initiated in [9,23] focuses on classical (hard) computational problems as filter languages and respective decision procedures heavily take advantage of the regularity of the set of input instances. Investigating the int_{Reg}-problem for NP-complete problems shows that the decidability of their int_{Reg}-problem is not trivial, e. g., int_{Reg}(SAT) is decidable [9], whereas int_{Reg}(Bounded Tiling) is not [23].[3] This is particularly interesting because the original hardness proofs of SAT and Bounded Tiling are both given by directly encoding Turing-machine computations into a problem instance [3,6]. Finding a generic characterization of NP-complete problems with a decidable int_{Reg}-problem is still an open problem. This work continues this line of research and we will focus on the NP-complete integer linear programming problem as the filter language, i. e., we investigate the problem int_{Reg}(ILP).

Our main result is that int_{Reg}(ILP) is decidable. The idea is to transform the given DFA that represents the regular set of instances into a condensed one that accepts a finite set of instances, such that the condensed set contains a positive instance if and only if this is the case for the original set.[4] This is done by first identifying for all pairs of states the set of coefficients that can be read between these two states, and then choosing a finite number of representatives for each such set of coefficients (in a sense, these are the coefficients that are 'most promising' regarding possible solutions). Then, again for all pairs of states, we identify a set of whole inequalities that can be read between these two states and that only have coefficients from the set of 'promising' coefficients constructed before. Finally, we will again choose suitable representatives for those sets of inequalities, from which we will construct the desired condensed automaton. We will also give bounds on the number and length of words accepted by the

[3] LOGSPACE and P also contain problems with undecidable int_{Reg}-problem [23].
[4] Our construction uses similar ideas as given in [11].

condensed automaton and, in the conclusions, discuss the chosen encoding and present an alternative encoding. The presented arguments can easily be adapted to proof the decidability of the int_{Reg}-problem for Linear Programming (with integer coefficients). Due to space restrictions some proofs are omitted.

2 Preliminaries

We assume the reader to be familiar with the basics of formal language theory and the complexity class NP. For a language descriptor A (e. g., regular expressions or automata), $L(A)$ denotes the language described by A. With $[n]$, $n \in \mathbb{N}$ we denote the set $\{1, \dots, n\}$. A deterministic finite automaton (DFA) A is a tuple $(Q, \Sigma, \delta, q_0, F)$ where Q is a finite set of states, Σ a finite alphabet, $\delta\colon Q \times \Sigma \to Q$ the (partial) transition function, q_0 the start state, and F is the set of final states. The transition function δ extends to the function $\delta^*\colon Q \times \Sigma^* \to Q$ in the usual way. We will only consider partial automata where every state is *coaccessible*, i. e., from every state, some final state is reachable.

We first give a formal definition of the problem Integer Linear Programming. While the standard-form of ILP varies in different areas, we refer to the definition in [22] where this problem is called LIQ. We will refer to the described problem as ILP. The problem is NP-complete if we ask for solutions in $\mathbb{Z}$ [22].

Definition 1 (ILP).
Given: Finite set $\mathcal{A}$ of pairs $(\boldsymbol{\alpha}, \beta)$ where $\boldsymbol{\alpha} \in \mathbb{Z}^m$ and $\beta \in \mathbb{Z}$.
Question: Is there an m-tuple $\boldsymbol{x} \in \mathbb{Z}^m$ such that $\boldsymbol{\alpha}^{\mathsf{T}} \cdot \boldsymbol{x} \le \beta$ for all $(\boldsymbol{\alpha}, \beta) \in \mathcal{A}$?

The problem will be encoded in the following way. The whole set $\mathcal{A}$ will be encoded in one word. For each pair $(\boldsymbol{\alpha}, \beta)$ the elements of $\boldsymbol{\alpha}$ and the β-value are encoded in binary over {0,1}. Each positive integer will be preceded with a + while each negative integer will be preceded with a $-$. The integers of α will be separated from β by a $\le$ symbol. The inequalities themselves are terminated by \$-symbols. Since we want to talk about *regular* languages of ILP-instances, we aim to have an encoding which is verifiable by a finite automaton. Therefore, we allow the inequalities of an ILP-instance to have different numbers of variables. The assignment of the coefficients to the variables is implicitly made by the order in which the coefficients occur. So, the i-th encoded coefficient in an inequality refers to variable x_i and is referenced as the coefficient with index i. As the inequalities of an ILP-instance may have different numbers of coefficients, they are interpreted as filled up with coefficients zero until all inequalities have the same number of coefficients and hence the same number of variables. Alternative encodings are discussed in Sect. 5. More formally,

$$L_{\mathrm{enc}} := L\left(\left(([+|-][0|1(0|1)^*])^* \le [+|-][0|1(0|1)^*]\$\right)^*\right)$$

is the *set of all encoded* ILP*-instances* and with $\mathrm{ILP}_{\mathrm{enc}}$ we denote the set of all *solvable* encoded ILP-instances. As an example, consider the following integer linear program and its encoding:

$$\{((5,1,0,-7),15)\,,((0,-8,1,0),-4)\,,((1,0,0,0),-1)\}\ ,$$
$$+\,101+1+0-111\leq+1111\,\$\ -0-1000+1\leq-100\,\$\ +1\leq-1\,\$\,.$$

Note that coefficients zero can either occur with a $+$ or a $-$ sign.

The question we want to investigate is whether the set of solvable ILP-instances, encoded in the above described way, and a regular language, given by an automaton, have a non-empty intersection.

Definition 2 **(*int*$_{\text{Reg}}$*(*ILP*)*).**
Given: Deterministic finite automaton A.
Question: Is $L(A) \cap \text{ILP}_{enc} \neq \emptyset$?

3 Construction of the Condensed Automaton

We will follow the ideas presented in [9] of investigating what kinds of loops can occur in the automaton without violating the encoding format, namely loops inside a coefficient, loops over whole coefficients, and loops over whole inequalities.

Definition 3. *Let $A = (Q, \Sigma, \delta, q_0, F)$ be a DFA. We define for all $q, q' \in Q$ and $s \in \{+, -\}$ the* coefficient transition set *$\Lambda^s_{q,q'}$ and β-*transition sets *$B^s_{q,q'}$ as*

$$\Lambda^s_{q,q'} = \{si \mid i \in \{0,1\}^* \wedge \exists \sigma \in \Sigma\backslash\{0,1,\$\} : \delta^*(q, si) = q' \wedge\ \delta(q', \sigma) \neq \emptyset\}.$$
$$B^s_{q,q'} = \{si \mid i \in \{0,1\}^* \wedge \delta^*(q, si) = q' \wedge \delta(q', \$) \neq \emptyset\}.$$

Intuitively speaking, these transition sets contain all coefficients and β-values which can be *completely* read between q and q'. Note that automata recognizing the transition sets are easily obtained from the original automata. When q, q' and s is clear from the context, then we will simply write Λ and B.

We now want to find a set of representatives reps(Λ) for each coefficient transition set Λ. The set reps(Λ) will contain only the smallest and largest coefficient, which in the following we will denote *extreme* coefficients, from the set Λ. Since all inequalities are of the form $\alpha_1 x_1 + \cdots + \alpha_n x_n \leq \beta$, increasing the absolute value of a positive summand $\alpha_i x_i$ makes the inequality system harder to be solved, while decreasing it may only enlarge the set of solutions (correspondingly for negative summands). So, we only have to consider the largest and smallest coefficient α_i contained in the coefficient transition set. The largest and smallest coefficient will correspond, in combination with a negative and positive x_i value, respectively, to the smallest negative and positive summand, respectively. If a coefficient transition set is infinite, it contains coefficients with an arbitrarily large magnitude, which we will represent by the meta-characters $+\infty$ and $-\infty$ in order to indicate that we can replace them with large enough values. Similarly, if a β-transition set $B^+_{q,q'}$ is infinite, we will use $+\infty$-symbol as a representative (indicating that we can find arbitrary large β-values and therefore such inequalities can be ignored), and for β-transition sets $B^-_{q,q'}$ we choose the element with the smallest magnitude as representative.

Definition 4. *For transition sets $\Lambda^s_{q,q'}$ and $B^s_{q,q'}$, we define:*

$$\operatorname{reps}(\Lambda^+_{q,q'}) := \begin{cases} \{\min(\Lambda^+_{q,q'}), +\infty\}, & \textit{if } |\Lambda^+_{q,q'}| = \infty; \\ \{\min(\Lambda^+_{q,q'}),\, \max(\Lambda^+_{q,q'})\}, & \textit{otherwise,} \end{cases}$$

$$\operatorname{reps}(\Lambda^-_{q,q'}) := \begin{cases} \{-\infty, \max(\Lambda^-_{q,q'})\}, & \textit{if } |\Lambda^-_{q,q'}| = \infty; \\ \{\min(\Lambda^-_{q,q'}),\, \max(\Lambda^-_{q,q'})\}, & \textit{otherwise,} \end{cases}$$

$$\operatorname{reps}(B^+_{q,q'}) := \begin{cases} \{+\infty\}, & \textit{if } |B^+_{q,q'}| = \infty; \\ \{\max(B^+_{q,q'})\}, & \textit{otherwise,} \end{cases}$$

$$\operatorname{reps}(B^-_{q,q'}) := \{\max(B^-_{q,q'})\}.$$

Since the transition sets are given by finite automata, it can be checked whether they are finite or infinite. The next step is to identify all inequalities which can be completely read in between two states and that only contain extreme coefficients, i. e., members from $\operatorname{reps}(\Lambda)$ and $\operatorname{reps}(B)$ as coefficients and β-values.

Definition 5. *Let $A = (Q, \Sigma, \delta, q_0, F)$ be a DFA with $L(A) \subseteq L_{enc}$. For every pair of states $q, q' \in Q$ we define the* inequality transition set *$\Xi_{q,q'}$ as:*

$$\Xi_{q,q'} = \{s_1 i_1 s_2 i_2 \dots s_k i_k \leq s_b j\$ \mid k \in \mathbb{N}, \exists p_0, \dots, p_{k+2} : \bigwedge_{\ell=1}^{k} s_\ell i_\ell \in \operatorname{reps}(\Lambda^{s_\ell}_{p_{\ell-1},p_\ell})$$
$$\wedge\, p_0 = q \,\wedge\, \delta(p_k, \leq) = p_{k+1} \wedge s_b j \in \operatorname{reps}(B^{s_b}_{p_{k+1},p_{k+2}}) \wedge \delta(p_{k+2}, \$) = q'\}.$$

Now we want to pick finitely many representatives for every inequality transition set $\Xi_{q,q'}$. Some sets Ξ contain for every partial solution $\boldsymbol{x}$ an inequality which can be satisfied by an extension of $\boldsymbol{x}$.[5] For those inequality transition sets, we simply choose \$ as the representative to indicate that this transition set does not participate in the problem as we can always find a satisfiable inequality in it. Two types of inequality transition sets have this property. If $\Xi_{q,q'}$ contains an inequality with an $+\infty$-symbol as β-value, then an inequality with an arbitrary high actual β-value can be read in between q and q'. So, for every value of the left side of the inequality we can read an even larger right side. The other type of Ξ sets are those which contain inequalities with an unbounded number of non-zero coefficients. Recall that the Ξ sets only contain coefficients which are representatives of Λ sets and hence the number of different coefficients in all Ξ sets is finite. Hence, the only reason an Ξ set is infinite is because the number of coefficients in the inequalities can be arbitrarily large. Therefore, those inequality transition sets are exactly the sets which are infinite after we removed all inequalities ending with *more* than $|Q| = n$ consecutive coefficients zero. By removing more than n consecutive coefficients zero from the end of the sum, we ensure that there is a non-zero coefficient under the last n coefficient. If the set is still infinite we can find inequalities with non-zero coefficients with an arbitrary

[5] An extension $\boldsymbol{v'} \in \mathbb{Z}^n$ of $\boldsymbol{v} \in \mathbb{Z}^m$ with $n > m$ coincides with $\boldsymbol{v}$ in all positions $i \leq m$.

high index. If the modified inequality transition sets are finite, we simply pick the whole set as the set of representatives. Inequalities with more than n consecutive coefficients zero after the last non-zero coefficient can also be ignored, because there is an equivalent inequality with less than n coefficients zero in the inequality transition set. With this considerations in mind, we define for all states $q, q' \in Q$ a set of representatives $\text{reps}(\Xi_{q,q'})$ for the inequality transition set $\Xi_{q,q'}$.

Definition 6. *Let $L_{Val} := L\left([+|-]\left([0|1(0|1)^*]|\infty\right)\right)$ and let $L_{Trash} := L((L_{Val})^* ([+|-]0)^{>n} \leq L_{Val}\$)$. For every inequality transition set $\Xi_{q,q'}$ we define*

$$\text{reps}(\Xi_{q,q'}) := \begin{cases} \{\$\}, & \text{if } \exists w \in \Xi_{q,q'} \text{ which ends with } +\infty\$; \\ & \text{or } |\Xi_{q,q'} \backslash L_{Trash}| = \infty; \\ \Xi_{q,q'} \backslash L_{Trash}, & \text{otherwise.} \end{cases}$$

Note that there are only finitely many sets $\text{reps}(\Xi_{q,q'})$ which are by construction all of a finite size.

We will now construct a condensed automaton which will have the finitely many inequalities, chosen as a representative, as its alphabet.

Definition 7. *Let $A = (Q, \Sigma, \delta, q_0, F)$ be a deterministic finite automaton with $L(A) \subseteq L_{enc}$. We define $\text{cond}(A) := (Q, \Sigma', \delta', q_0, F)$ with the alphabet $\Sigma' = \bigcup_{q,q' \in Q} \text{reps}(\Xi_{q,q'})$ and $\delta' = \{(q, \xi, q') \mid \xi \in \text{reps}(\Xi_{q,q'})\}$.*

Lemma 12 will show that we only have to consider simple paths in $\text{cond}(A)$.

4 Correctness of the Condensed Automaton

We will now present several lemmas which in the end will prove that $L(A) \cap \text{ILP}_{\text{enc}} \neq \emptyset$ if and only if $L'(\text{cond}(A)) \cap \text{ILP}_{\text{enc}} \neq \emptyset$. With $L'(\text{cond}(A))$ we refer to the language $L(\text{cond}(A))$ where the wild-cards ∞ are replaced by actual coefficients. First, we will show that it is sufficient to consider only the largest and smallest coefficient which can be read in between two states.

Lemma 8. *Let $A = (Q, \Sigma, \delta, q_0, F)$ be a DFA, let $w \in L(A) \cap \text{ILP}_{enc}$ with solution $\boldsymbol{x}$ and let α_{ij} be the j-th coefficient of the i-th inequality of w. Let $w = w'\alpha_{ij}w''$. If $\alpha_{ij} = a_{ij}b_{ij}c_{ij}$, $b_{ij} \neq \varepsilon$, and $\delta^*(q_0, w'a_{ij}) = \delta^*(q_0, w'a_{ij}b_{ij})$, then the following holds:*

1. *Assume $x_j \geq 0$ and α_{ij} has a $+$ sign. Let w' result from w by replacing α_{ij} with $a_{ij}c_{ij}$. Then $w' \in L(A) \cap \text{ILP}_{enc}$ and $\boldsymbol{x}$ is a solution for w'.*
2. *Assume $x_j \geq 0$ and α_{ij} has a $-$ sign. Let w' result from w by replacing α_{ij} with $a_{ij}(b_{ij})^2c_{ij}$. Then $w' \in L(A) \cap \text{ILP}_{enc}$ and $\boldsymbol{x}$ is a solution for w'.*
3. *Assume $x_j \leq 0$ and α_{ij} has a $+$ sign. Let w' result from w by replacing α_{ij} with $a_{ij}(b_{ij})^2c_{ij}$. Then $w' \in L(A) \cap \text{ILP}_{enc}$ and $\boldsymbol{x}$ is a solution for w'.*
4. *Assume $x_j \leq 0$ and α_{ij} has a $-$ sign. Let w' result from w by replacing α_{ij} with $a_{ij}c_{ij}$. Then $w' \in L(A) \cap \text{ILP}_{enc}$ and $\boldsymbol{x}$ is a solution for w'.*

Next, we will focus on whole inequalities and show that restricting the inequalities in words from $L(A)$ to the above defined representatives does not affect the existence of a solvable ILP-instance in $L(A)$. We already explained before Definition 6 that for every solution vector $\boldsymbol{x}$ we can replace the inequalities with an $+\infty$-symbol as β-value by inequalities with actual β-values, which are satisfied by $\boldsymbol{x}$. With respect to inequality transition sets containing inequalities with arbitrarily large non-zero coefficients, we will show next how to simultaneously replace such inequalities in a way that the replacements are satisfied by an extension of $\boldsymbol{x}$. So, if the ILP-instance is solvable without inequalities from sets $\varXi$ which are represented by \$-symbols, then we can enlarge the instance and the solution to include those inequalities.

For the next lemma, we want to distinguish the infinite inequality transition sets without an unbounded β-value from the finite ones.

Definition 9.

$$\begin{aligned} \mathit{Inf}_{\varXi} &:= \{\varXi_{q,q'} \mid \mathrm{reps}(\varXi_{q,q'}) = \{\$\} \wedge \varXi_{q,q'} \cap (L\,({L_{\mathit{Val}}}^{*} \leq +\infty\$) = \emptyset)\} \\ \mathit{Fin}_{\varXi} &:= \{\varXi_{q,q'} \mid \mathrm{reps}(\varXi_{q,q'}) \neq \{\$\}\} \end{aligned}$$

We will now find alternative representatives for the sets in $\mathrm{Inf}_{\varXi}$ such that if an ILP-instance consisting only of inequalities from the sets in $\mathrm{Fin}_{\varXi}$ has a solution $\boldsymbol{x}$, then we can extend the ILP-instance with any combination of alternative representatives of the sets in $\mathrm{Inf}_{\varXi}$, such that $\boldsymbol{x}$ can be extended to a solution of the extended ILP-instance (we shall prove this in Lemma 11). This shows that we can ignore inequalities from the sets in $\mathrm{Inf}_{\varXi}$, i. e., the ones with representative \$.

Definition 10. *Let* $\sigma\colon [|\mathit{Inf}_{\varXi}|] \to \mathit{Inf}_{\varXi}$ *be an arbitrary but fixed ordering of the sets in* $\mathit{Inf}_{\varXi}$. *Let* $n := |Q|$ *and* $\#_{\pm}(w)$ *denote the number of signs in an inequality* w. *The function* $\min\limits_{\mathrm{lex}}$ *returns the lexicographical minimal element of a set*[6]. *For every* $1 \leq i \leq |\mathit{Inf}_{\varXi}|$ *we define for the inequality transition set* $\sigma(i)$ *in* $\mathit{Inf}_{\varXi}$ *a set of* alternative representatives arep *as*

$$\mathit{arep}(\sigma(i)) \leftarrow \min_{\mathrm{lex}}(\{w \in \sigma(i) \mid (i+1)\cdot n < \#_{\pm}(w) \leq (i+2)\cdot n\}).$$

For each fixed i the assignment of $\mathit{arep}(\sigma(i))$ in the above definition can be determined by computing the intersection of two regular sets given by DFAs, yielding a finite language. This finite language can be enumerated in order to find the lexicographical minimal element. The idea is to pick inequalities as alternative representatives which together form a matrix in row echelon form. For every inequality we assign the variable x_k with the highest indexed non-zero coefficient α_k with a value of which magnitude is large enough, such that the summand $\alpha_k x_k$ dominates the inequality. An inequality in the sets of $\mathrm{Fin}_{\varXi}$ can only consist of up to $n = |Q|$ different coefficients. The definition of $\mathit{arep}(\varXi)$

[6] The function $\min\limits_{\mathrm{lex}}$ is used to make the definition clear. Any other element of the set could be used as well.

ensures that the representatives of $\Xi \in \mathrm{Inf}_\Xi$ contain more coefficients than any representative of the finite inequality transition sets. It also ensures that the number of coefficients contained in the representing inequality is strictly monotonously rising with the order σ. Especially, the index of the highest non-zero coefficient of $arep(\sigma(i+1))$ is higher than the index of the highest non-zero coefficient of $arep(\sigma(i))$.

Lemma 11. *Let w be a solvable* ILP*-instance consisting only of inequalities from sets in Fin_Ξ. Let $\boldsymbol{x}$ be a valid solution of w. Then, for every* ILP*-instance w' consisting of w and additional inequalities from $\{arep(\Xi) \mid \Xi \in Inf_\Xi\}$ the vector $\boldsymbol{x}$ can be extended to a solution $\boldsymbol{x}'$ of w'.*

Proof. Let $\boldsymbol{x} = (x_1, x_2, \ldots, x_i)$, let m be the number of variables in w', and let $var\text{-}set(\xi)$ be a function returning the variables appearing in the inequality ξ with a non-zero coefficient. Let $coeff(\xi, y_j)$ denote the coefficient of variable y_j in the inequality ξ, let $value(y_j)$ denote the assigned value x_j of the variable y_j, and let $\beta(\xi)$ refer to the right side β of the inequality ξ. Algorithm 1 assigns values to the new variables $y_{i+1}, y_{i+2}, \ldots, y_m$ appearing in w' such that $\boldsymbol{x}' = (x_1, \ldots, x_i, x_{i+1}, \ldots, x_m)$ is a solution of the instance w' and works as follows. We go through the inequalities appearing in w' which have been chosen

Algorithm 1. Extending solution $\boldsymbol{x}$ of ILP-instance w to solution $\boldsymbol{x}'$ of w'.

```
AssignedVars ← $\{y_1, \ldots, y_i\}$
for $j \leftarrow 1$ to $|\mathrm{Inf}_\Xi|$ do
    CurIneq ← $arep(\sigma(j))$,    ToAssign ← $\emptyset$
    if CurIneq appears in $w'$ then
        ToAssign ← $var\text{-}set$(CurIneq)\AssignedVars
        MaxVar ← $y_k \in$ ToAssign with highest index $k$
        for all $y \in$ ToAssign\{MaxVar} do
            $value(y) \leftarrow 0$
        end for
        SumOthCoeff ← $\sum_{y_l \in \{var\text{-}set(\mathtt{CurIneq}) \setminus \{\mathtt{MaxVar}\}\}} coeff(\mathtt{CurIneq}, y_l) \cdot value(y_l)$
        CoeffMaxVar ← $coeff$(CurIneq, MaxVar),    $b \leftarrow \beta$(CurIneq)
        $value$(MaxVar) ← $|b - \mathtt{SumOthCoeff}| \cdot (-1)^{\frac{\mathtt{CoeffMaxVar}}{|\mathtt{CoeffMaxVar}|}}$
        AssignedVars ← AssignedVars ∪ ToAssign
    end if
end for
```

as alternative representatives for the sets in Inf_Ξ in the same order as when we assigned the representatives. Thus, the number of appearing variables per inequality is rising. In every considered inequality, there is at least one variable which has not appeared in the previously considered inequalities. We assign the new variables with a zero value, except for the variable with the highest index. This variable (MaxVar) gets a value which compensates all the other summands in the inequality. The sign of MaxVar is converse to the sign of its coefficient resulting in a negative summand. We can choose the value of `MaxVar` freely, since the variable has not appeared in any other inequality we considered earlier. If it appears in any later considered inequality, there will always be at least one new variable in the inequality which has not appeared earlier, and which can again compensate every other summand. It is easy to see that the considered inequality `CurIneq` is satisfied by the chosen variable assignment. Hence, $\boldsymbol{x}' = (x_1, \ldots, x_i, value(y_{i+1}), \ldots, value(y_m))$ is a solution of the ILP-instance w'. □

Only simple paths in $\text{cond}(A)$ have to be considered in order to find a solvable ILP-instance in $L'(\text{cond}(A))$.

Lemma 12. *Let $w, w' \in L_{enc}$ and w' be w without an arbitrary inequality ξ from w. (So, w' is w with one inequality less.) If $w \in \text{ILP}_{enc}$ then $w' \in \text{ILP}_{enc}$.*

We will now show that if there is a solvable ILP-instance in $L'(\text{cond}(A))$, then we can replace any \$-symbols in this instance by actual inequalities, resulting in a solvable ILP-instance in $L(A)$. On the other hand, if there is a solvable ILP-instance in $L(A)$ the modifications we made on A while constructing $\text{cond}(A)$ preserve the existence of a solvable ILP-instance in the obtained language $L'(\text{cond}(A))$.

Theorem 13. *Let $A = (Q, \Sigma, \delta, q_0, F)$ be a deterministic finite automaton with $L(A) \subseteq L_{enc}$. Then, $L(A) \cap \text{ILP}_{enc} \neq \emptyset$ if and only if $L'(\text{cond}(A)) \cap \text{ILP}_{enc} \neq \emptyset$.*

Proof Sketch. Let $w \in L(A) \cap \text{ILP}_{\text{enc}}$. We only keep those inequalities in w which are read between some states q and q' on the path labeled with w in A and for which $\Xi_{q,q'} \in \text{Fin}_{\Xi}$. All other inequalities in w are replaced by \$-symbols. Then, wherever possible we pump the coefficients in w up or down, corresponding to the sign of the associated variable in $\boldsymbol{x}$ until we obtain an ILP-instance w' in $L'(\text{cond}(A))$. It holds that w' is also solvable.

Let $w \in L'(\text{cond}(A)) \cap \text{ILP}_{\text{enc}}$. The corresponding ILP-instance in $L(\text{cond}(A))$ only consists of inequalities from Ξ sets in Fin_{Ξ} or \$-symbols. We first replace all \$-symbols which are representatives of Ξ sets in Inf_{Ξ} by alternative representatives from Definition 10. According to Lemma 11 the obtained ILP-instance is still solvable. Then, we replace the leftover \$-symbols which are representatives of Ξ sets which contain inequalities with an unbounded β-value. Since we know a solution for the considered ILP-instance, we can pick inequalities with large enough β-value such that the obtained ILP-instance w' is satisfied by an extension of the considered solution. In w' all \$-symbols are replaced by actual inequalities and hence $w' \in L(A)$. As w' is also solvable $L(A) \cap \text{ILP}_{\text{enc}} \neq \emptyset$ follows. □

Now, we are ready to put the pieces together and present our main result. In the following, we give a decision procedure for the int_{Reg}-problem of ILP.

Theorem 14. *The problem $int_{\text{Reg}}(\text{ILP})$ is decidable.*

Proof. Since L_{enc} is regular, we can restrict $L(A)$ to the regular language $L(A) \cap L_{\text{enc}}$. Let $A' = (Q, \Sigma, \delta, q_0, F)$ be a deterministic finite automaton with $L(A') = L(A) \cap L_{\text{enc}}$. For the automaton A', the Definitions 3 and 4 describe the construction of coefficient transition sets and assigning their representatives. In Definition 5 inequality transition sets are constructed based on those representatives. These inequality transition sets get representatives themselves in Definition 6. In Definition 7 a new automaton $\text{cond}(A')$ is defined, based on the representatives for the inequality transition sets. All those constructions can be computed by an algorithm. Theorem 13 states that $L(A') \cap \text{ILP}_{\text{enc}} \neq \emptyset \Leftrightarrow L'(\text{cond}(A')) \cap \text{ILP}_{\text{enc}} \neq \emptyset$. Finally, Lemma 12 tells us that if there is a solvable ILP-instance in $L'(\text{cond}(A'))$ at all, then there is a solvable ILP-instance w' in $L'(\text{cond}(A'))$ with a corresponding ILP-instance $w \in L(\text{cond}(A'))$ which can be read on a simple path in $\text{cond}(A')$. The instance w is obtained from w' by replacing coefficients with an absolute value above $3|Q| \cdot \left(|Q|2^{|Q|}\right)^{2|Q|+4}(1+2^{|Q|})$ by ∞-symbols. Since there are only finitely many simple paths in an automaton, and testing a given ILP-instance for solvability can be done in finite time, we can test all words in $L'(\text{cond}(A'))$ which correspond to labels of simple paths in $\text{cond}(A')$ for membership in ILP_{enc} in finite time. Hence, $L(A) \cap \text{ILP}_{\text{enc}} \neq \emptyset$ is decidable. □

5 Conclusion

The number of considered words in $L'(\text{cond}(A))$ is bounded by $2^{\mathcal{O}(|Q|^2 \log(|Q|))}$. The length of considered words in $L(\text{cond}(A))$ regarding the alphabet Σ' of $\text{cond}(A)$ is bounded by $\mathcal{O}(|Q|)$. Finally, the length of considered words regarding the alphabet Σ of A, meaning that we replace \$- and ∞-symbols by actual substrings over Σ, is bounded by $\mathcal{O}(|Q|^7)$. Therefore, we can guess some word in $L'(\text{cond}(A))$ and check its membership in ILP_{enc} by solving the represented ILP-instance. Since ILP is NP-complete $\mathit{int}_{\text{Reg}}(\text{ILP}) \in \text{NP}$ follows. For a given ILP-instance, we can construct a DFA accepting only this instance in polynomial time. Hence $\mathit{int}_{\text{Reg}}(\text{ILP})$ is NP-complete.

According to [19], the presented results are stable under applying a length-preserving morphism to the encoding scheme. The results are also stable under changing the binary encoding to any base-k encoding. Recall that in order to talk about regular sets of problem-instances, we want to have a problem encoding which can be verified by a deterministic finite automaton. In particular, we can not verify with a DFA that all variables appear in a certain inequality or that the inequalities have the same length. Therefore, we have implicitly filled the inequalities with coefficients zero to ensure the same number of variables per inequality. Note that this forbids an explicit matrix representation of an ILP-instance. Instead of referencing the variables of an inequality implicitly by the number and order of the coefficients we could also use another encoding, where we explicitly name the variable and the coefficient. In this setting multiple occurrences of the same variable would be possible and would be interpreted as a summation of terms. Here, we would define transition sets for coefficients and for variables. We would not pump the number of variables in an inequality but instead pump the label of a variable to make it independent of other variables and inequalities. We would still treat the coefficients in the same way and we would also consider only simple paths. In terms of the 'int_{Reg}-techniques' of [23] we would switch from the replacing technique to the separating technique and the int_{Reg}-problem of ILP in this variable-explicit encoding would still be decidable.

Although we considered partial DFAs, the construction also works for partial NFAs. It might be worthwhile to investigate further extensions of int_{Reg}(ILP) such as Boolean combinations of inequalities or quadratic programming.

Acknowledgment. The author thanks Markus L. Schmid for proofreading and helpful discussions and is grateful to the anonymous reviewers for their suggestions. The author was partially supported by DFG (FE 560/9-1).

References

1. Anderson, T., Loftus, J., Rampersad, N., Santean, N., Shallit, J.: Detecting palindromes, patterns and borders in regular languages. Inf. Comput. **207**(11), 1096–1118 (2009). https://doi.org/10.1016/j.ic.2008.06.007
2. Bodlaender, H.L., Heggernes, P., Lokshtanov, D.: Graph modification problems (Dagstuhl seminar 14071). Dagstuhl Rep. **4**(2), 38–59 (2014)
3. Cook, S.A.: The complexity of theorem-proving procedures. In: Harrison, M.A., Banerji, R.B., Ullman, J.D. (eds.) Proceedings of 3rd Annual ACM Symposium on Theory of Computing, STOC 1971, pp. 151–158. ACM, New York (1971)
4. Cormode, G., Muthukrishnan, S.: The string edit distance matching problem with moves. ACM Trans. Algorithms **3**(1), 2:1–2:19 (2007)
5. Eiben, E., Ganian, R., Knop, D., Ordyniak, S.: Unary integer linear programming with structural restrictions. In: Lang, J. (ed.) Proceedings of 27th International Joint Conference on Artificial Intelligence, IJCAI 2018, pp. 1284–1290 (2018)
6. Emde Boas van, P.: The Convenience of Tilings. Lecture Notes in Pure and Applied Mathematics, pp. 331–363. Marcel Dekker Inc., New York (1997)
7. Ganian, R., Ordyniak, S.: The complexity landscape of decompositional parameters for ILP. Artif. Intell. **257**, 61–71 (2018)
8. Gao, X., Xiao, B., Tao, D., Li, X.: A survey of graph edit distance. PAA Pattern Anal. Appl. **13**(1), 113–129 (2010). https://doi.org/10.1007/s10044-008-0141-y
9. Güler, D., Krebs, A., Lange, K.-J., Wolf, P.: Deciding regular intersection emptiness of complete problems for PSPACE and the polynomial hierarchy. In: Klein, S.T., Martín-Vide, C., Shapira, D. (eds.) LATA 2018. LNCS, vol. 10792, pp. 156–168. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-77313-1_12
10. Hladík, M.: Interval linear programming: a survey. In: Mann, Z.A. (ed.) Linear Programming – New Frontiers in Theory and Applications, Chap. 2, pp. 85–120. Nova Science Publishers, New York (2012)
11. Lange, K., Reinhardt, K.: Set automata. In: Combinatorics, Complexity and Logic; Proceeding, DMTCS 1996, pp. 321–329 (1996)
12. Lempitsky, V.S., Kohli, P., Rother, C., Sharp, T.: Image segmentation with a bounding box prior. In: ICCV 2009, pp. 277–284. IEEE Computer Society (2009)
13. Li, H.: Two-view motion segmentation from linear programming relaxation. In: Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007), pp. 1–8. IEEE Computer Society (2007)

14. Liu, Y., Wang, J., Guo, J.: An overview of kernelization algorithms for graph modification problems. Tsinghua Sci. Technol. **19**(4), 346–357 (2014)
15. Rubtsov, A.A.: Regular realizability problems and regular languages. CoRR abs/1503.05879 (2015). http://arxiv.org/abs/1503.05879
16. Rubtsov, A.A., Vyalyi, M.N.: Regular realizability problems and models of a generalized nondeterminism. CoRR abs/1105.5894 (2011). http://arxiv.org/abs/1105.5894
17. Tarasov, S., Vyalyi, M.: Orbits of linear maps and regular languages. In: Kulikov, A., Vereshchagin, N. (eds.) CSR 2011. LNCS, vol. 6651, pp. 305–316. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20712-9_24
18. Vyalyi, M.N.: On models of a nondeterministic computation. In: Frid, A., Morozov, A., Rybalchenko, A., Wagner, K.W. (eds.) CSR 2009. LNCS, vol. 5675, pp. 334–345. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03351-3_31
19. Vyalyi, M.N.: On regular realizability problems. Probl. Inf. Transm. **47**(4), 342–352 (2011)
20. Vyalyi, M.N.: On expressive power of regular realizability problems. Probl. Inf. Transm. **49**(3), 276–291 (2013). https://doi.org/10.1134/S0032946013030058
21. Vyalyi, M.N., Rubtsov, A.A.: On regular realizability problems for context-free languages. Probl. Inf. Transm. **51**(4), 349–360 (2015). https://doi.org/10.1134/S0032946015040043
22. Wagner, K., Wechsung, G.: Computational Complexity. Springer, Netherlands (1986)
23. Wolf, P.: Decidability of the regular intersection emptiness problem. Master's thesis, Wilhelm Schickhard Institut für Informatik, Universität Tübingen (2018)