Computing affine hulls over Q and Z from sets represented by Number Decision Diagrams

Louis Latour

This work was partially supported by a FRFC grant: 2.4530.02

Computing affine hulls over $\mathbb{Q}$ and $\mathbb{Z}$ from sets represented by Number Decision Diagrams

Louis Latour

Université de Liège
Institut Montefiore, B28
4000 Liège, Belgium
`latour@montefiore.ulg.ac.be`

Abstract. Number Decision Diagrams (NDD) are finite automata representing sets of integer vectors and have recently been proposed as an efficient data structure for representing sets definable in Presburger arithmetic. In this context, some work has been done in order to generate formulas or sets of generators from the NDDs. Taking another step in this direction, this paper present algorithms that takes as input an NDD and computes the affine hull over $\mathbb{Q}$ or over $\mathbb{Z}$ of the set represented by the NDD, i.e., the smallest set defined by a conjunction of equations or by a conjunction of equations and congruence relations that includes the set represented by the NDD. Our algorithms run in time $\mathcal{O}(|Q| \cdot |\Sigma_r^n| \cdot n)$ and $\mathcal{O}(|Q|^3 \cdot |\Sigma_r^n| \cdot n^3)$ respectively, where n is the number of components of the vectors represented by the NDD, and $|Q|$ and Σ_r^n are the number of states and the alphabet of the NDD. On a prototype implementation, the computations of affine hulls of NDDs with more than 100000 states are done in seconds.

1 Introduction

It has been known for a long time that finite automata can be used for representing sets of integer vectors (see [BHMV94]). In particular, sets definable in Presburger arithmetic [Pre29], i.e., first-order logic over the integers with addition and the order relation, can be represented by finite automata. Many applications rely on Presburger arithmetic, including integer programming problems, compiler optimization techniques, program analysis tools and model-checking.

There exist different equivalent representations of Presburger definable sets (see [BHMV94]), including formulas, semi-linear sets and finite automata, and different approaches have been developed for handling Presburger definable sets. Finite automata have recently been investigated as an efficient data structure for representing Presburger definable sets in practical applications [WB95,BC96]. Finite automata present two main advantages, there is a canonical representation and efficient procedures exist for set operations and inclusion tests. However, simple arithmetic operations, such as affine transformation, can be costly if performed on automata. Therefore, it may appear efficient to handle both automaton and formula representations of the set and perform the operations on the most appropriate representation. Also, having access to a simple formula representation of the sets can shed light on the sometimes hidden relationships between variables, or give a useful broad view of the set. It also provides a link to theorem provers.

Working with both automata and formulas implies being able to move from one representation to the other. While generating automata from formulas is now well understood [Kla04], the issue of generating formulas from automata has only been dealt with more recently. Algorithm for restricted classes of sets appeared in [Ler03], [Lat04] and [Lug04], and a solution for the general case has been presented in [Ler04b]. In this paper, we approach the problem of extracting information from automata differently, and instead of generating a formula matching exactly the set represented, we compute the affine hull over $\mathbb{Q}$ and over $\mathbb{Z}$, i.e., the smallest affine space over $\mathbb{Q}$ or affine module over $\mathbb{Z}$ that includes the set represented by the automaton. The main interests are that the computations are fast (linear if arithmetic operations are performed in constant time), and affine spaces and affine modules are easily dealt with since they can be represented by n equations and congruence relations or by n generators together with an element of the set, where n is the number of vector components. Furthermore, for a number of applications, affine hulls already provide useful information. For example, in the context of verification, one could simplify a model by removing some variables via the equations and congruence relations. Finally, the algorithms presented in this paper could be integrated in a more general algorithm computating exact formulas for sets represented by automata, as it is done in [Lat04] where affine hulls are computed in order to identify the left-hand sides (i.e. the vector of coefficients) of the inequations occurring in the formula.

An algorithm computing the affine hull over $\mathbb{Q}$ of sets of positive vectors represented by finite automata (with a least significant digit first encoding) has been already presented in [Ler04a]. The time complexity of this algorithm is $\mathcal{O}(|Q| \cdot |\Sigma_r^n| \cdot n^3)$, where $|Q|$ is the number of states, n is the number of components in the vectors of the represented set and Σ_r^n is the alphabet of the automaton. Also, a finite automaton representing a set of integer vectors can be viewed as a program, the states being the control locations and the transitions being affine assignments since adding a digit d to an encoding of a number z is equivalent to multiply z by the encoding basis and adding d (when using a most significant digit first encoding scheme), some results in the field of static analysis of program can be applied, and in particular, the method proposed in [MH04] for computing affine relations among variables in a program can be used with minor adaptations for computing the affine hull over $\mathbb{Q}$ of sets represented by finite automata. The time complexity is then $\mathcal{O}(|Q| \cdot |\Sigma_r^n| \cdot n^3)$. In this paper, we present a more efficient algorithm whose time complexity is $\mathcal{O}(|Q| \cdot |\Sigma_r^n| \cdot n)$, and $\mathcal{O}(|Q| \cdot |\Sigma_r^n| \cdot n^2)$ if a minimal set of generators is required.

Regarding the affine hull over $\mathbb{Z}$, nothing has been done directly on sets represented by finite automata. In the context of static analysis, an algorithm for computing the affine and congruence relations satisfied in a control point of an affine programs has been presented in [Gra91]. Although the computation is proved to be finite, there is no bound on the number of operations required. More recently, [MH05] describes a polynomial time algorithm for computing affine relations over $\mathbb{Z}_m$, i.e. integer arithmetic modulo m, satisfied by the variables at a given control location. In this paper, we give the first polynomial time algorithm for computing the affine hull over $\mathbb{Z}$ of sets represented by finite automata. The exact time complexity of our algorithm is $\mathcal{O}(|Q|^3 \cdot |\Sigma_r^n| \cdot n^3)$. Note that our algorithm for computing affine hulls over $\mathbb{Q}$ is part of our algorithm for computing the affine hulls over $\mathbb{Z}$.

This paper is organized as follows. In Section 2, we recall basic properties regarding automata theory as well as linear algebra. In Section 3, we show how finite automata can represent sets of integer vectors. In Section 4, we present an efficient representation for generators of vector spaces, $\mathbb{Z}$-modules and $\mathbb{Z}_m$-modules. In Sections 5 and 6, we present our algorithms for computing affine hulls over $\mathbb{Q}$ and over $\mathbb{Z}$ respectively. Some experimental results are provided in Section 7, and we conclude in Section 8.

2 Preliminaries

We start with some preliminaries from linear algebra and automata theory. In what follows, for any finite set S, the number of elements in S will be denoted by $|S|$.

2.1 Finite automata

An *alphabet* is a finite nonempty set of symbols. A *word* over an alphabet Σ is a finite sequence of symbols taken from Σ. The symbol ε denotes the empty word, i.e., the word containing no symbol. The *length* of a word w, denoted by $|w|$, is the number of symbols in w. A language over Σ is a set of words over Σ, and Σ^* denotes the set of all words over Σ.

A *deterministic finite automaton (DFA)* $\mathcal{A}$ is a quintuple $(Q, \Sigma, \delta, s_{init}, Q_F)$, where Q is a finite set of states, Σ is the input alphabet, $\delta : Q \times \Sigma \to Q$ is the transition function, $s_{init} \in Q$ is the initial state and $Q_F \subseteq Q$ is the set of final states.

The function δ is extended to words : $\hat{\delta}(s, \varepsilon) = \{s\}$ and $\hat{\delta}(s, uw) = \bigcup_{s' \in \delta(s,u)} \hat{\delta}(s', w)$. If $s' = \delta(s, u)$ for $s, s' \in Q$ and $u \in \Sigma$, then we say that there is a *transition* from s to s' *labeled* by u. By extension, there is a *path* from s to s' *labeled* by w if $s' = \hat{\delta}(s, w)$. The *language* of $\mathcal{A}$, denoted by $L(\mathcal{A})$ is the set of words labeling paths from the initial state to a final state. The set of words labeling paths from a state s_1 to a state s_2 in $\mathcal{A}$ is denoted as $L_{\mathcal{A}}(s_1 \to s_2)$.

The DFA $\mathcal{A} = (Q, \Sigma, \delta, s_{init}, Q_F)$ is *reduced* if for all words $w \neq \varepsilon$ labeling a path rooted at s_{init}, there exists a word $v \in \Sigma^*$ such that $wv \in L(\mathcal{A})$.

2.2 Basics on linear algebra

The following definitions and results can be found in elementary linear algebra textbooks, such as [Jac89].

As usual, $\mathbb{Q}$, $\mathbb{Z}$ and $\mathbb{N}$ denote the sets of rational numbers, integers and naturals, and $\mathbb{Z}_m = \mathbb{Z}/(m\mathbb{Z})$, i.e., the equivalence classes of $\mathbb{Z}$ modulo m. In the following, $\mathbb{D}$ will denote any set among $\mathbb{Q}$, $\mathbb{N}$, $\mathbb{Z}$ and $\mathbb{Z}_m$. In the case of $\mathbb{Z}_m$, any addition or multiplication of elements in $\mathbb{Z}_m$ correspond to addition or multiplication in $\mathbb{Z}$ modulo m so that the result is in $\{0, \ldots, m-1\}$. The set of vectors with n components in $\mathbb{D}$ is denoted $\mathbb{D}^n$. The i-component of a vector x is written $x[i]$. The superscript $\cdot^T$ denotes transposition.

For any set $S \subseteq \mathbb{D}^n$, vector $a \in \mathbb{D}^n$ and scalar $\gamma \in \mathbb{D}$, we denote by $a + S$ and γS the sets $\{a + x \mid x \in S\}$ and $\{\gamma x \mid x \in S\}$ respectively.

For $m, n \in \mathbb{N}$, $m, n \geq 1$, $\mathbb{D}^{m \times n}$ is the set of $m \times n$-matrices with entry in $\mathbb{D}$. For a matrix $A \in \mathbb{D}^{m \times n}$, the row index set of A is $\{1, \ldots, m\}$ and the column index set is $\{1, \ldots, n\}$, and the entry located in the ith row and jth column is written $A[i, j]$. The ith row of A is denoted $A[i, *]$ and similarly, the jth column is denoted $A[*, j]$. Let $S \subseteq \mathbb{D}^n$. The $\mathbb{D}$-*linear hull* of S, denoted $\mathrm{lin}_{\mathbb{D}}(S)$, and the *D-affine hull* of S, denoted $\mathrm{aff}_{\mathbb{D}}(S)$, are defined as follows.

$$\mathrm{lin}_{\mathbb{D}}(S) = \{\sum_{i=1}^{n} \lambda_i x_i \mid \lambda_i \in \mathbb{D}, x_i \in S\}, \tag{1}$$

$$\mathrm{aff}_{\mathbb{D}}(S) = \{\sum_{i=1}^{n} \lambda_i x_i \mid \lambda_i \in \mathbb{D}, x_i \in S, \sum_{i=1}^{n} \lambda_i = 1\}. \tag{2}$$

Example 1. Let $S = \{(1, 0), (1, 2), (1, 4)\}$. $\mathrm{aff}_{\mathbb{Q}}(S) = \{(1, k) \mid k \in \mathbb{Q}\}$ and $\mathrm{aff}_{\mathbb{Z}}(S) = \{(1, 2 * k) \mid k \in \mathbb{Z}\}$.

The vectors $x_1, \ldots, x_n \in \mathbb{D}^n$ are *linearly independent over* $\mathbb{D}$ iff $\sum_{j=1}^{n} \alpha_j x_j = 0$ with $\alpha_j \in \mathbb{D}$ implies that $\alpha_j = 0$ for $j = 1, \ldots, n$. If the vectors are not linearly independent, they are *linearly dependent over* $\mathbb{D}$. A set of vectors G is *free over* $\mathbb{D}$ iff the vectors in G are linearly independent over $\mathbb{D}$. A set $G \subseteq \mathbb{D}^n$ $\mathbb{D}$-*generates* a set $S \subseteq \mathbb{D}^n$ iff $\mathrm{lin}_{\mathbb{D}}(G) = S$. If in addition, G is free over $\mathbb{D}$, then G is a $\mathbb{D}$-basis of S.

A subset $M \subseteq \mathbb{D}^n$ of vectors with entries in $\mathbb{D}$ is a $\mathbb{D}$-*module* iff $M \neq \emptyset$ and $M = \mathrm{lin}_{\mathbb{D}}(M)$. A subset $S \subseteq \mathbb{D}^n$ is a $\mathbb{D}$-*affine module* iff $S = a + M$, where $a \in \mathbb{D}^n$ and M is a $\mathbb{D}$-module.

Proposition 1 *Let $S \subseteq \mathbb{D}^n$. The set $\mathrm{lin}_{\mathbb{D}}(S)$ (resp. $\mathrm{aff}_{\mathbb{D}}(S)$) is the smallest $\mathbb{D}$-module (resp. $\mathbb{D}$-affine module) containing S. The $\mathbb{D}$-module M such that $\mathrm{aff}_{\mathbb{D}}(S) = a + M$ for some $a \in \mathbb{D}^n$ is unique.*

Proposition 2 *Any $\mathbb{D}$-module $S \subseteq \mathbb{D}^n$ has a $\mathbb{D}$-basis, and all $\mathbb{D}$-basis of S have the same number of elements $d \leq n$ called the* dimension *of S.*

Since $\mathbb{Q}$ is a field, $\mathbb{Q}$-modules and $\mathbb{Q}$-affine modules have more properties than their counterparts over the rings $\mathbb{Z}$ and $\mathbb{Z}_m$ (except if m is prime, in which case $\mathbb{Z}_m$ is also a field). Consequently, $\mathbb{Q}$-modules and $\mathbb{Q}$-affine modules are called *vector space* and *affine space* respectively. The most relevant property of vector spaces is expressed in the following proposition.

Proposition 3 *Let $S \subseteq \mathbb{Q}^n$ be a vector space. If $x_1, \ldots, x_k \in S$ are linearly independent over $\mathbb{Q}$, then there exists $y_1, \ldots, y_t \in S$ such that $\{x_1, \ldots, x_k, y_1, \ldots, y_t\}$ is a $\mathbb{Q}$-basis of S.*

2.3 Size and complexity

We define the size of numbers as follows. The size of an integer number $a \in \mathbb{Z}$ is 1 if $a = 0$, and $1 + \lfloor \log |a| \rfloor$ otherwise. The size of a rational a/b where $a \in \mathbb{Z}$, $b \in \mathbb{N} \setminus \{0\}$ and $\gcd(a, b) = 1$ is 1 if $a = 0$ and $\lfloor 1 + \log |a| + \log |b| \rfloor$ otherwise.

In order to reason about the complexity of the algorithms presented in this paper, we assume that direct memory accesses are performed in constant time and that arithmetic operations are perform in unit time.

3 Automata-based representation of integer vector sets

In this section, we explain how automata can represent sets of integer vectors. The main idea consists in establishing a mapping between vectors and words. Our encoding scheme of vectors is based on the positional expression of numbers (most significant digit first) with a signed-complement system for negative integers.

Given an *encoding basis* $r \in \mathbb{N}$, with $r > 1$, an *r-encoding of an integer* $a \in \mathbb{Z}$ is a word w over Σ_r, such that if $w = u_p u_{p-1} \dots u_0$ where each $u_i \in \Sigma_r = \{0, \dots, r-1\}$, $u_p = 0$ if $a \geq 0$ and $u_p = r - 1$ if $a < 0$, and $a = -r^p \cdot \frac{u_p}{r-1} + \sum_{i=0}^{p-1} u_i r^i$.

In order to encode a vector $z \in \mathbb{Z}^n$, one simply reads synchronously one digit from the encodings of all its components, provided that these encodings share the same length. This requirement can always be met by prefixing the encoding by a sequence of copies of the leading digit of the initial encoding. So, an *r-encoding of an integer vector* $z \in \mathbb{Z}^n$ is a word w over Σ_r^n, such that if $w = u_p u_{p-1}, \dots u_0$ where each $u_i \in (\Sigma_r^n, u_p \in (0, r-1)^n$, and for each $j \in \{1, \dots, n\}$, we have $z[j] = -r^p \cdot \frac{u_p[j]}{r-1} + \sum_{i=0}^{p-1} u_i[j] r^i$.

The fact that w is an r-encoding of z is denoted by $\langle w \rangle_r = z$. Also, we simply write 0 for the symbol $(0, \dots, 0) \in \Sigma_r^n$.

Based on the definition of the encoding scheme, for all encodings $u \in (\Sigma_r^n)^+$ and words $v \in (\Sigma_r^n)^*$, we have $\langle uv \rangle_r = r^{|v|} \langle u \rangle_r + \langle 0v \rangle_r$.

Let $S \subseteq \mathbb{Z}^n$. If the language $L(S)$ containing all the encodings of all the vectors in S is regular, then any DFA $\mathcal{A}$ accepting $L(S)$, i.e. such that $L(\mathcal{A}) = L(S)$, is a *Number Decision Diagram (NDD)*, and we say that $\mathcal{A}$ *represents* S. In this paper, we use the following notations. We denote by $S_{\mathcal{A}(s_{init} \to s)}$ the set of vectors whose encoding labels paths from s_{init} to s in the NDD $\mathcal{A}$, and by $S_{\mathcal{A}}$ the set represented by the NDD $\mathcal{A}$. The encoding scheme that we use here is the same as the one proposed in [BHMV94] and extended to $\mathbb{Z}$ in [WB95].

It is known (see [Boi99]) that the sets definable in the first order theory $\langle \mathbb{Z}, +, <, V_r \rangle$ correspond exactly to the sets that can be represented by finite-state automata using the r-encoding scheme that has just been discussed. Note that $\langle \mathbb{Z}, +, <, V_r \rangle$ is the first-order logic over the integers with addition and the ordering relation, with an additional predicate $V_r(x, y)$ returning true if y is the greatest power of r dividing x and false otherwise.

In the remaining of this paper, r-encodings are simply called encodings since we will always use the same encoding basis r.

4 Triangular sets

The algorithms presented in this paper manipulate intensively vector spaces, $\mathbb{Z}$-modules and $\mathbb{Z}_m$-modules. In order to have more efficient procedures, we maintain sets of generators in a particular form : the *triangular form* [MH05]. For a non-zero vector x, we call i the *leading index* of x and $x[i]$ the *leading entry* of x if $x[i] \neq 0$ and $x[j] = 0$ for $j \in \{1, \ldots, i-1\}$. A set of non-zero vectors T is *triangular* iff the leading entries of all vectors in T are positive and for all distinct vectors $x, x' \in T$, the leading indices of x and x' are distinct. Intuitively, a set is triangular if the vectors are the rows of a echelon matrix A with no zero-row, i.e. each row of A has a non-zero element and if $A[i, k]$ and $A[j, k']$ are the first non-zero element of the ith and jth rows respectively with $j > i$, then $k' > k$. Note that a triangular set of integer vectors is a set of linearly independent vectors over $\mathbb{Q}$ and $\mathbb{Z}$.

Efficient procedures for generating an integer basis in triangular form of a vector space or of a $\mathbb{Z}$-module given a set of integer generators are readily available from the current literature (see [Sto00]). In particular, we have the following results.

Proposition 4 *There exists an algorithm* `GetTriangQBasis` *which, given a finite set $G \subseteq \mathbb{Z}^n$ as input, generates a triangular set $\overline{G} \subseteq \mathbb{Z}^n$ such that the sizes of the components of vectors in $\overline{G}$ are bounded by $n \cdot (k + \log n)$ where k is the bound on the component size of vectors in G, and $\mathrm{lin}_{\mathbb{Q}}(G) = \mathrm{lin}_{\mathbb{Q}}(\overline{G})$. In addition, the time complexity of* `GetTriangQBasis` *is $\mathcal{O}(|G| \cdot n^2)$.*

Proof. It suffices to apply the general Gaussian elimination to the matrix A whose rows are the vectors in G. □

Proposition 5 *There exists an algorithm* `GetTriangZBasis` *which, given a finite set $G \subseteq \mathbb{Z}^n$ as input, generates a triangular set $\overline{G} \subseteq \mathbb{Z}^n$ such that the sizes of the components of vectors in $\overline{G}$ are bounded by $k \cdot n \cdot \log(\sqrt{n})$, where k is the bound on the component size of vectors in G, and $\mathrm{lin}_{\mathbb{Z}}(G) = \mathrm{lin}_{\mathbb{Z}}(\overline{G})$. In addition, the time complexity of* `GetTriangZBasis` *is $\mathcal{O}(|G| \cdot k \cdot n^3 \cdot \log(\sqrt{n}))$.*

Proof. It suffices to generate the matrix A whose rows are the vectors in the set S, and then to compute the Hermite form H of A [Sto00], i.e., the matrix H such that H is in Hermite form[1] and $H = UA$ for some square integer matrix U whose determinant is 1. □

Note that computing a basis is more difficult over $\mathbb{Z}$ than over $\mathbb{Q}$ since a set of linearly independent vectors over $\mathbb{Z}$ cannot be extended to form a basis as it is the case over $\mathbb{Q}$.

Proposition 6 *Given a triangular set $T \subseteq \mathbb{Z}^n$ and a vector $x_0 \in \mathbb{Z}^n$, there exists an algorithm that generates a set of congruences and a set of equations whose modulo and coefficient sizes are bounded by $\mathcal{O}(n \log n + nk)$, k being a bound on the size of the numbers in the vectors in T and x_0, such that the solutions of the system of equations (resp. equations and congruences) are exactly the elements in $x_0 + \mathrm{lin}_{\mathbb{Q}}(T)$ (resp. $x_0 + \mathrm{lin}_{\mathbb{Z}}(T)$).*

Proof. See Appendix, Section 9.1. □

For a triangular set $T = \{y_1, \ldots, y_k\}$ in $\mathbb{Z}^n_{p^q}$, we define $\mathrm{rank}(T)$ as $\sum_i q_i + q^{n-k}$ where $d_i p^{q_i}$ is the leading entry of y_i, with $\gcd(d_i, p)$. Clearly, $0 \leq \mathrm{rank}(T) \leq n \cdot q$.

[1] A matrix H is in Hermite form over $\mathbb{Z}$ if H is in echelon form and if $H[i, j_i]$ denotes the first non-zero elements of the ith row, one has $0 \leq H[i', j_i] < H[i, j_i]$ for all $i' < i$.

Proposition 7 *There exists an algorithm* `UpdateTriangZm`*, which, given a prime number p, a positive integer q, a triangular set $T \subseteq \mathbb{Z}^n_{p^q}$ and a vector $x \in \mathbb{Z}^n_{p^q}$, such that if $T' =$* `UpdateTriangZm`*(p, q, T, x), then the following assertions are valid.*

- *$T' \subseteq \mathbb{Z}^n_{p^q}$ and T' is triangular.*
- *$\mathrm{lin}_{\mathbb{Z}_{p^q}}(T') = \mathrm{lin}_{Z_{p^q}}(T \cup \{x\})$.*
- *If $T' \neq T$, then $\mathrm{rank}(T') < \mathrm{rank}(T)$.*
- *The time complexity of* `UpdateTriangZm` *is $\mathcal{O}(n^2 \cdot q)$.*

Proof. See [MH05]. □

5 Affine hulls over $\mathbb{Q}$

In this section, we present an algorithm which takes as input a reduced NDD $\mathcal{A} = (Q, \Sigma^n_r, \delta, s_{init}, Q_F)$ and generates the affi ne hull over $\mathbb{Q}$ of the set represented by $\mathcal{A}$.

We briefly present the algorithm based on [MH04], and then present a more effi cient algorithm which takes advantage of the special affi ne transformation corresponding to transitions in NDDs. In addition, this more effi cient version is also part of the more sophisticated algorithm for computing the affi ne hull over $\mathbb{Z}$.

The idea of the algorithm based on [MH04] is to explore the paths of $\mathcal{A}$ rooted at the initial state s_{init} and to compute for each state s a vector x_s and a triangular set of vectors G_s such that $x_s \in S_{\mathcal{A}(s_{init} \to s)}$ and $x_s + \mathrm{lin}_{\mathbb{Q}}(G_s) \subseteq \mathrm{aff}_{\mathbb{Q}}(S_{\mathcal{A}(s_{init} \to s)})$. When handling a path labeled by w from s_{init} to s, the algorithm applies the following recursive procedure.

- If x_s has not yet been set, one sets x_s equal to $\langle w \rangle_r$ and we propagate w from s, that is, we apply the procedure to all paths from s_{init} to s' labeled by wu with $u \in \Sigma^n_r$ such that $\delta(s, u) = s'$.
- Otherwise, if $\langle w \rangle_r \in x_s + \mathrm{lin}_{\mathbb{Q}}(G_s)$, then we do not propagate w. If on the other hand, $\langle w \rangle_r \notin x_s + \mathrm{lin}_{\mathbb{Q}}(G_s)$, one has to add $\langle w \rangle_r - x_s$ to G_s and to propagate w from s.

Since for each s, one sets at most once x_s and one adds at most n vectors to G_s, the number of iterations is bounded, and at some point, no more path needs to be explored. It can be proved that at this point, $x_s + \mathrm{lin}_{\mathbb{Q}}(G_s) = \mathrm{aff}_{\mathbb{Q}}(S_{\mathcal{A}(s_{init} \to s)})$ for all states s. Finally, one has to take the union of the affi ne hulls corresponding to fi nal states and again, take the affi ne hull over $\mathbb{Q}$ of this set.

We can improve the algorithm presented above. The main property is expressed in the following lemma.

Lemma 8 *Let $s, s' \in Q$ with $\hat{\delta}(s, v) = s'$ for some v, and let $V, V_{s'} \subseteq \mathbb{Q}^n$ be vector spaces such that $\mathrm{aff}_{\mathbb{Q}}(S_{\mathcal{A}}) = x_F + V$ and $\mathrm{aff}_{\mathbb{Q}}(S_{\mathcal{A}(s_{init} \to s')}) = x' + V_{s'}$ for some $x_F, x' \in \mathbb{Z}^n$. For all $x_1, x_2 \in S_{\mathcal{A}(s_{init} \to s)}$, we have $x_1 - x_2 \in V_{s'} \subseteq V$.*

Proof. By defi nition, there exist encodings u_1, u_2 such that $u_1, u_2 \in L_{\mathcal{A}}(s_{init} \to s)$ and $\langle u_1 \rangle_r = x_1$ and $\langle u_2 \rangle_r = x_2$. By defi nition, $u_1 v, u_2 v \in L_{\mathcal{A}}(s_{init} \to s')$ and $r^{|v|} x_1 + \langle 0v \rangle_r, r^{|v|} x_2 + \langle 0v \rangle_r \in S_{\mathcal{A}(s_{init} \to s')}$. Therefore, $(r^{|v|} x_1 + \langle 0v \rangle_r) - (r^{|v|} x_2 +$

$\langle 0v\rangle_r) \in \mathrm{lin}_{\mathbb{Q}}(S_{\mathcal{A}(s_{init}\to s')})$, and so, by definition, $x_1 - x_2 \in V_{s'}$. Since $\mathcal{A}$ is reduced, there exists a word w with $u_1vw, u_2vw \in L(\mathcal{A})$. So, applying the same reasoning, we find that $V_{s'} \subseteq V$. □

Thanks to the previous property, we note that in the algorithm sketched above, if $\langle w\rangle_r - x_s$ is added to G_s, then $\langle w\rangle_r - x_s$ can be added to all $G_{s'}$ where s' is reachable from s. We deduce that it is not necessary to compute at each individual state s one basis G_s and one element x_s such that $x_s + \mathrm{lin}_{\mathbb{Q}}(G_s) = \mathrm{lin}_{\mathbb{Q}}(S_{\mathcal{A}(s_{init}\to s)})$. One only needs to consider one element x_s per state and one basis G for the whole NDD. Also, from each state, one has to propagate only path. Indeed, if $w_1, w_2 \in L_{\mathcal{A}}(s_{init} \to s)$ and $\langle w_1\rangle_r - \langle w_2\rangle_r$ is added to G, then for $v \in \Sigma_r^n$, $\langle w_1v\rangle_r - \langle w_2v\rangle_r \in \mathrm{lin}_{\mathbb{Q}}(G)$. Finally, in the above description, we did not specify the order with which one consider the propagated paths. Adopting a breadth first search approach allows us to manipulate smaller numbers.

Our algorithm `QAffineHull` takes a reduced NDD as input and it works as follows.

1. Initially, the set G is empty. Also, for each state, one stores a vector $x_s \in \mathbb{Z}^n$ which is initially set to $\perp$.
2. It considers paths of increasing length originating from s_{init}, starting with all paths of length 1, and at the kth iteration, it handles paths of length k that have been propagated so far. When handling a path labeled by w from s_{init} to s, there are two possibilities.
   - If $x_s = \perp$, x_s is set to $\langle w\rangle_r$, and one will consider at the next iteration the paths labeled by wu for all $u \in \Sigma_r^n$ with $\delta(s,u) = s'$ for some s'.
   - If $x_s \neq \perp$, then we add $\langle w\rangle_r - x_s$ to G.
3. When all states have been visited once, we pick one final state $s_F \in Q_F$ and we add to G all vectors $x_s - x_{s_F}$ where $s \in Q_F$. Then, the algorithm terminates and it returns G as well as the vector x_{s_F}.

The formal algorithm is described in Appendix, Section 9.2.

Theorem 9 *Let $l_{\min} \leq |Q|$ be the smallest positive integer such that for all states $s \in Q$, there exists an encoding w_s such that $\hat{\delta}(s_{init}, w_s) = s$ with $|w_s| \leq l_{\min}$. Let $x_F \in \mathbb{Z}^n$ and $G \subseteq \mathbb{Z}^n$ such that $(G, x_F) =$ `QAffineHull`$(\mathcal{A})$. We have*

- $x_F + \mathrm{lin}_{\mathbb{Q}}(G) = \mathrm{aff}_{\mathbb{Q}}(S_{\mathcal{A}})$,
- $|G| \leq |Q| \cdot \Sigma_r^n$,
- *The time complexity of* `QAffineHull` *is* $\mathcal{O}(|Q| \cdot |\Sigma_r^n| \cdot n)$,
- *The size of the numbers in G are bounded by* $\mathcal{O}(l_{\min})$.

Proof. See Appendix, Section 9.3. □

Finally, according to Proposition 4, we can compute a triangular set $\overline{G}$ of at most n generators from the set G computed via the algorithm `QAffineHull`. The sizes of the numbers in $\overline{G}$ are then bounded by $\mathcal{O}(n \cdot (|Q| + \log n))$ and the time complexity for the call `GetTriangQBasis`(G) is $\mathcal{O}(|Q| \cdot |\Sigma_r^n| \cdot n^2)$. In addition, thanks to Proposition 6, we can compute a system of linear equations $a_i x = 0$, $i = |\overline{G}| + 1, \ldots, n$ such that $x \in x_F + \mathrm{lin}_{\mathbb{Q}}(G) \Leftrightarrow \bigwedge_{i=|\overline{G}|+1,\ldots,n} a_i(x - x_F) = 0$.

6 Affine hulls over $\mathbb{Z}$

In this section, we give an algorithm for computing the affine hull in $\mathbb{Z}^n$ of the set represented by a reduced NDD $\mathcal{A} = (Q, \Sigma_r^n, \delta, s_{init}, Q_F)$.

Note first that in general, if $(G, x_F) =$ `QAffineHull`$(\mathcal{A})$, the set $x_F + \mathrm{lin}_{\mathbb{Z}}(G)$ is not equal to $\mathrm{aff}_{\mathbb{Z}}(S_{\mathcal{A}})$. This stems from the fact that Lemma 8 does not hold in the integer case because it does not consider the factor $r^{|v|}$ of the affine transformations corresponding to the path from s_1 to s_2. Taking this factor into consideration leads to the following lemma.

Lemma 10 *Let $s \in Q$ with $\hat{\delta}(s, v) \in Q_F$ for some v, and let $M \subseteq \mathbb{Z}^n$ be the $\mathbb{Z}$-module such that $\mathrm{aff}_{\mathbb{Z}}(S_{\mathcal{A}}) = x_F + M$ for some $x_F \in \mathbb{Z}^n$. For all $x_1, x_2 \in S_{\mathcal{A}(s_{init} \to s)}$, we have $r^{|v|} \cdot (x_1 - x_2) \in M$.*

Proof. The proof is similar to the proof of Lemma 8, except that we do not get rid of the factor $r^{|v|}$. □

Based on the above lemma, we can extend Theorem 9 and prove the following property regarding the output of algorithm `QAffineHull`.

Lemma 11 *Let $d_{\min}$ be the smallest positive integer such that for all states $s \in Q$, there exists an encoding w_s such that $\hat{\delta}(s, w_s) \in Q_F$ with $|w_s| \leq d_{\min}$. Let $M, G \subseteq \mathbb{Z}^n$ and $x_F \in \mathbb{Z}^n$ such that $\mathrm{aff}_{\mathbb{Z}}(S_{\mathcal{A}}) = x_F + M$ and $(G, x_F) =$ `QAffineHull`$(\mathcal{A})$.*

- *for all $s \in Q$, for all $x_1, x_2 \in S_{\mathcal{A}(s_{init} \to s)}$, $x_1 - x_2 \in \mathrm{lin}_{\mathbb{Z}}(G)$, and,*
- *for all $g \in G$, $r^{d_{\min}} g \in M$.*

Proof. See Appendix, Section 9.5. □

We now turn on the actual computation of $\mathrm{aff}_{\mathbb{Z}}(S_{\mathcal{A}})$. A first approach, similar to what is done in [MH04] for the affine hull over $\mathbb{Q}$, is to compute a finite G_s for each state s such that if $x_s \in S_{\mathcal{A}(s_{init} \to s)}$, then $x_s + \mathrm{lin}_{\mathbb{Z}}(G_s) \subseteq \mathrm{aff}_{\mathbb{Z}}(S_{\mathcal{A}(s_{init} \to s)})$. This can be done by keeping a basis of G_s and considering paths of increasing length until reaching a fixpoint at which for all states s, for all $w \in L_{\mathcal{A}}(s_{init} \to s)$, $\langle w \rangle_r \in x_s + \mathrm{lin}_{\mathbb{Z}}(G_s)$. The problems with this approach are that numbers in the basis of G_s can grow exponentially, and secondly, there is no bound on the length of the paths before reaching the fixpoint. Based on Lemma 11, those two problems can be solved in the following way. Let $G_{pre}, M \subseteq \mathbb{Z}^n$ and $x_F \in \mathbb{Z}^n$ such that $(G_{pre}, x_F) =$ `QAffineHull`$(\mathcal{A})$, and $\mathrm{aff}_{\mathbb{Q}}(S_{\mathcal{A}}) = x_F + M$. Since for all states s and $x_s \in S_{\mathcal{A}(s_{init} \to s)}$, $\langle w \rangle_r - x_s \in \mathrm{lin}_{\mathbb{Z}}(G_{pre})$, $\langle w \rangle_r - x_s$ is a linear combination over $\mathbb{Z}$ of vectors in G_{pre}, for any $\mathbb{Z}$-basis $\overline{G_{pre}}$ of $\mathrm{lin}_{\mathbb{Z}}(G_{pre})$, the decomposition of $\langle w \rangle_r - x_s$ with respect to $\overline{G_{pre}}$ is unique. Also, since for all $g \in G_{pre}$, $r^{d_{\min}} g \in M$, this also holds for vectors $\overline{g} \in \overline{G_{pre}}$, and adding any combination of $r^{d_{\min}} \overline{g}$ to any $\langle w \rangle_r - x_s$ does not change the affine hull over $\mathbb{Z}$. So, once the decomposition of $\langle w \rangle_r - x_s$ with respect to $\overline{G_{pre}}$ has been performed, we can work in $\mathbb{Z}_{r^{d_{\min}}}$, i.e. work in integer arithmetic modulo $r^{d_{\min}}$.

Based on the above considerations, our algorithm `ZAffineHull` takes a reduced NDD as input and works as follows.

1. Via the algorithm QAffineHull, one computes a set G_{pre} and a vector x_F. Then, one computes a basis $\overline{G_{pre}}$ of $\mathrm{lin}_{\mathbb{Z}}(G_{pre})$ and set $p = |\overline{G_{pre}}|$. Then for each state s, one associates a triangular set $\Gamma_s \subseteq \mathbb{Z}^p_{r^{d_{\min}}}$ initially empty.
2. One considers paths of increasing length originating from s_{init}, starting with all paths of length 1. Given the label w of a path from s_{init} to s, the procedure works as follows.
   – If $x_s = \bot$, then x_s is set to $\langle w \rangle_r$ and one propagates w from s, that is, for all $u \in \Sigma^n_r$ with $\delta(s, u) = s'$ for some $s' \in Q$, one handles the path labeled by wu at the next iteration.
   – If $x_s \neq \bot$, then one decomposes $\langle w \rangle_r - x_s$ into a linear combination $\sum_{g_i \in \overline{G_{pre}}} \gamma_i g_i$, which is always possible with $\gamma_i \in \mathbb{Z}$. Let $c \in \mathbb{Z}^p$ with $c[i] = \gamma_i \mod r^{d_{\min}}$ and $c[i] \in \{0, \ldots, r^{d_{\min}} - 1\}$, and let Γ'_s = UpdateTriangZm(r, $d_{\min}$, Γ_s, c). There are 2 possibilities.
      • If $\Gamma'_s \neq \Gamma_s$, then Γ_s is set to Γ'_s and one propagates w from s.
      • Otherwise, one does nothing and w is not propagated.
3. One updates a triangular set $\Gamma \subseteq \mathbb{Z}^p_{r^{d_{\min}}}$, initially empty, via UpdateTriangZm with all vectors $c \in \Gamma_s$ for all $s \in Q_F$.
4. Finally one generates the set $G \subseteq \mathbb{Z}^n$ by adding the vectors $g \in \mathbb{Z}^n$ such that $g = \sum_{g_i \in \overline{G_{pre}}} c[i] \cdot g_i$ for some $c \in \Gamma$, $g = r^{d_{\min}} g_i$ for some $g_i \in \overline{G_{pre}}$, or $g = x_s - x_F$ for some final state $s \in Q_F$. Then, one returns G together with x_F.

The formal algorithm is described in Appendix, Section 9.4.

Theorem 12 *Let $l_{\min}, d_{\min} \leq |Q|$ be the smallest positive integers such that for all states $s \in Q$, there exist encodings w_l, w_d such that $\hat{\delta}(s_{init}, w_l) = s$ with $|w_l| \leq l_{\min}$ and $\hat{\delta}(s, w_d) = s_F$ for some $s_F \in Q_F$ with $|w_d| \leq d_{\min}$. Let $x_F \in \mathbb{Z}^n$, $G \subseteq \mathbb{Z}^n$ such that $(G, x_F) =$ ZAffineHull$(\mathcal{A})$. We have*

– $x_F + \mathrm{lin}_{\mathbb{Z}}(G) = \mathrm{aff}_{\mathbb{Z}}(S_{\mathcal{A}})$,
– $|G| \leq |Q| + 2n$,
– *Size of numbers in G are bounded by* $\mathcal{O}(n \cdot \log(\sqrt{n}) \cdot l_{\min} + d_{\min})$,
– *The time complexity of* ZAffineHull *is* $\mathcal{O}(|Q| \cdot |\Sigma^n_r| \cdot (\log(\sqrt{n}) \cdot l_{\min} + d^2_{\min}) \cdot n^3)$.

Proof. See Appendix, Section 9.6.

Note that if $(G, x_F) =$ ZAffineHull$(\mathcal{A})$, then by applying the function GetTriangZBasis to G, we can generate a basis $\overline{G}$ of $\mathrm{lin}_{\mathbb{Z}}(G)$ in time $\mathcal{O}(|Q| \cdot (l_{\min} + d_{\min}) \cdot n^5)$ and the size of the numbers in $\overline{G}$ are bounded by $\mathcal{O}((l_{\min} + d_{\min}) \cdot n^3)$. Also, thanks to Proposition 6, from $\overline{G}$ and x_F, we can generate a set of equations and congruence relations describing $\mathrm{aff}_{\mathbb{Z}}(S_{\mathcal{A}})$.

7 Experimental results

The algorithms presented in this paper have been implemented within the LASH library [LAS]. Note that the algorithms have been slightly modified in order to use the serial encoding as presented in [BL04], which significantly decreases the running time. By using the serial encoding, we simplify the transition relation at the expense of additional states. As a rule of thumb, the number of states is multiplied by the number of components of the represented vectors, and the number of transition can be exponentially decreased. As encoding basis, we have taken $r = 2$.

The time and memory used for the computation of the algorithms `QAffineHull` and `ZAffineHull` in a prototype implementation running on a pentium-M at 1,5 GHz are given in the table below. The computations include the generation of a triangular set $\overline{G}$ such that $x_F + \mathrm{lin}_{\mathbb{Q}}(\overline{G}) = \mathrm{aff}_{\mathbb{Q}}(S_{\mathcal{A}})$ or $x_F + \mathrm{lin}_{\mathbb{Z}}(\overline{G}) = \mathrm{aff}_{\mathbb{Z}}(S_{\mathcal{A}})$. The columns indicate successively the set on which the computation is performed, the number of components of the vectors in the set, the number of states in the corresponding NDD (with alphabet Σ_2), the values of $l_{\min}$ and $d_{\min}$ (see Theorems 9 and 12), and finally, the time and memory requirement for the computation of `QAffineHull` and `ZAffineHull` successively. Note that all sets $S_1, \ldots, S_{12}$ are defined by a boolean combination of several equations, inequations and congruence relations but $S_1, \ldots, S_6$ are $\mathbb{Z}$-affine modules which is not the case of $S_7, \ldots, S_{12}$.

		$\mathcal{A}$			`QAffineHull`		`ZAffineHull`	
Set	n	Nb. States	$l_{\min}$	$d_{\min}$	Time (sec.)	Mem (Mb)	Time (sec.)	Mem (Mb)
S_1	7	64874	3	12	1.0	6.1	3.5	46.7
S_2	6	115727	2	15	1.6	10.4	4.6	64.5
S_3	6	287713	6	27	3.3	27.4	22.5	162.1
S_4	6	215685	4	4	3.3	22.5	10.8	123.4
S_5	10	281135	4	5	3.1	31.4	119.9	379.3
S_6	11	112754	2	5	2.3	13.1	10.9	183.4
S_7	7	279598	4	7	4.3	29.2	63.2	203.8
S_8	7	42067	5	10	0.8	4.3	6.4	30.6
S_9	6	54186	5	5	1.2	5.4	6.6	30.8
S_{10}	7	50580	5	6	0.7	5.1	7.2	36.7
S_{11}	6	52177	4	8	0.9	4.9	4.2	29.3
S_{12}	6	44920	6	7	1.0	4.4	4.5	25.4

In the above table, we note that in the sets considered, the values of $l_{\min}$ and $d_{\min}$ are small compared to $|Q|$, even more so if one uses the serialized encoding. There exist sets for which the values of $l_{\min}$ and $d_{\min}$ have the same magnitude as $|Q|$. For example, the NDDs representing the sets $x = 0 \mod 2^k$ in base 2 have k states and $l_{\min} \simeq d_{\min} \simeq k$. Our intuition is that whenever the characteristics numbers of a set (i.e., the maximal value for finite set, the coefficient of the inequation in a quantifier-free Presburger formula, ...) are small then, $l_{\min}$ and $d_{\min}$ are also small and our algorithms perform very well.

8 Conclusion

In this paper, we have presented two algorithms, `QAffineHull` and `ZAffineHull`, that take a reduced NDD $\mathcal{A}$ as input and compute the affine hull over $\mathbb{Q}$ and over $\mathbb{Z}$ respectively of the set represented by $\mathcal{A}$. More precisely, they generate a pair (G, x_F) with a finite set $G \subseteq \mathbb{Z}^n$ and $x_F \in \mathbb{Z}^n$ such that $x_F + \mathrm{lin}_{\mathbb{Q}}(G)$ (resp. $x_F + \mathrm{lin}_{\mathbb{Z}}(G)$) is the affine hull over $\mathbb{Q}$ (resp. $\mathbb{Z}$) of the set represented by $\mathcal{A}$. The size of the numbers manipulated in `QAfineHull` (resp. `ZAffineHull`) are bounded by $\mathcal{O}(|Q|)$ (resp. $\mathcal{O}(n\log(\sqrt{n})\cdot|Q|)$) and the time complexity is $\mathcal{O}(|Q|\cdot|\Sigma_r^n|\cdot n)$ (resp. $\mathcal{O}(|Q|^3\cdot|\Sigma_r^n|\cdot n^3)$). The algorithms perform very well for NDDs such that the distances from the initial state to each state and the distances from each state to an accepting state are small, as we have shown in a prototype implementation.

References

[BC96] A. Boudet and H. Comon. Diophantine equations, Presburger arithmetic and finite automata. In *Proceedings of CAAP'96*, number 1059 in Lecture Notes in Computer Science, pages 30–43. Springer-Verlag, 1996.

[BHMV94] V. Bruyère, G. Hansel, C. Michaux, and R. Villemaire. Logic and p-recognizable sets of integers. *Bulletin of the Belgian Mathematical Society*, 1(2):191–238, March 1994.

[BL04] B. Boigelot and L. Latour. Counting the solutions of presburger equations without enumerating them. *Theoretical Computer Science*, 313(1):17–29, February 2004.

[Boi99] B. Boigelot. *Symbolic methods for exploring infi nite state spaces*. PhD Thesis, Université de Liège, Belgium, 1999.

[Gra91] P. Granger. Static analysis of linear congruence equalitites among variables of a program. In S. Abramsky and T. S. E. Maibaum, editors, *TAPSOFT'91: Proc. of the International Joint Conference on Theory and Practice of Software Development*, pages 169–192. Springer, Berlin, Heidelberg, 1991.

[Jac89] Nathan Jacobson. *Basic algebra, I.* W. H. Freeman and Company, New York, second edition, 1989.

[Kla04] Felix Klaedtke. On the automata size for Presburger arithmetic. In *Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science (LICS 2004)*, pages 110–119. IEEE Computer Society Press, 2004.

[LAS] The Liège Automata-based Symbolic Handler (LASH). Available at `http://www.montefiore.ulg.ac.be/˜boigelot/research/lash/`.

[Lat04] L. Latour. From automata to formulas: Convex integer polyhedra. In *Proceedings of 19th IEEE Symposium on Logic in Computer Science (LICS 2004)*, pages 120–129. IEEE Computer Society Press, 2004.

[Ler03] J. Leroux. *Algorithmique de la vérifi cation des systèmes à compteurs. Approximation et accélération. Implémentation de l'outil FAST*. PhD Thesis, Ecole Normale Supérieure de Cachan, Cachan, France, 2003.

[Ler04a] J. Leroux. The affine hull of a binary automaton is computable in polynomial time. *Electr. Notes Theor. Comput. Sci.*, 98:89–104, 2004.

[Ler04b] J. Leroux. A polynomial time Presburger criterion and synthesis for number decision diagram. Technical report, Université de Montréal, 2004.

[Lug04] D. Lugiez. From automata to semi-linear sets: a solution for polyhedra and even more general sets. Technical Report 21-2004, Laboratoire d'informatique de Marseilles, 2004.

[MH04] M. Müller-Olm and H. Seidl. A note on Karr's algorithm. In Josep Diaz, Juhani Karhumk̈i, and Arto LepistÖ, editors, *Proceedings of the 31st International Colloquium on Automata, Languages and Programming (ICALP 2004)*, volume 3142 of *Lecture Notes in Computer Science*. Springer-Verlag Heidelberg, 2004.

[MH05] M. Müller-Olm and H. Seidl. Analysis of modular arithmetic. In *To appear in the European Symposium on Programming (ESOP 2005)*, Lecture Notes in Computer Science. Springer-Verlag Heidelberg, 2005.

[Pre29] M. Presburger. Über die Volständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt. In *Comptes Rendus du Premier Congrès des Mathématiciens des Pays Slaves*, pages 92–101, 395, Warsaw, Poland, 1929.

[Sto00] A. Storjohann. *Algorithms for matrix canonical forms*. PhD Thesis, Swiss federal institute, Zurich, 2000.

[WB95] P. Wolper and B. Boigelot. An automata-theoretic approach to Presburger arithmetic constraints. In *Proceedings of Static Analysis Symposium*, volume 983 of *Lecture Notes in Computer Science*, pages 21–32, Glasgow, September 1995. Springer-Verlag.

9 Appendix

9.1 Proof of Proposition 6

Proposition 13 *Given a triangular set $T \subseteq \mathbb{Z}^n$ with $|T| = q$, and a vector $x_0 \in \mathbb{Z}^n$, there exists an algorithm that generates a set of congruences $a_i.x \equiv \beta_i \mod m_i$, $i = 1, \ldots, q$, and a set of equations $a_i.x = \beta_i$, $i = q+1, \ldots, n$, where numbers are bounded by $\mathcal{O}(n \log n + nk)$, k being a bound on the size of the numbers in the vectors in T and x_0, such that*

$$x \in x_0 + \mathrm{lin}_{\mathbb{Z}}(T) \Leftrightarrow \bigwedge_{i=1,\ldots,q} a_i.x \equiv \beta_i \mod m_i \wedge \bigwedge_{i=q+1,\ldots,n} a_i.x = \beta_i,$$

and

$$x \in x_0 + \mathrm{lin}_{\mathbb{Q}}(T) \Leftrightarrow \bigwedge_{i=q+1,\ldots,n} a_i.x = \beta_i.$$

Proof. Without loss of generality, we may assume that T can be represented in matrix form as $\begin{bmatrix} B \\ C \end{bmatrix}$ such that $B \subseteq \mathbb{Z}^{q \times q}$ is not singular, i.e., there exists a matrix B^{-1} such that $B^{-1}B = I_q$ where I_q is the identity matrix with q rows and columns.

So, we have for all $x \in \mathbb{Z}^n$,

$$x \in x_0 + \mathrm{lin}_{\mathbb{Z}}(T) \Leftrightarrow (x - x_0) = \begin{bmatrix} B \\ C \end{bmatrix} \lambda \text{ for some } \lambda \in \mathbb{Z}^q. \tag{3}$$

Since B is not singular, $\begin{bmatrix} B^{-1} & 0 \\ CB^{-1} & I_{n-q} \end{bmatrix}$ is not singular, and we have

$$\begin{bmatrix} B \\ C \end{bmatrix} \lambda \Leftrightarrow \begin{bmatrix} B^{-1} & 0 \\ CB^{-1} & I_{n-q} \end{bmatrix} (x - x_0) = \begin{bmatrix} I_q \\ 0 \end{bmatrix} \lambda. \tag{4}$$

If the ith row of $\begin{bmatrix} B^{-1} & 0 \end{bmatrix}$ is $\frac{a_i}{m_i}$, with $a_i \in \mathbb{Z}^n$ and $m_i \in \mathbb{N} \setminus \{0\}$, the set of congruences is then $a_i(x - x_0) \equiv 0 \mod m_i$, $i = 1, \ldots, q$, and if the ith row of $\begin{bmatrix} CB^{-1} & I_{n-q} \end{bmatrix}$ is $\frac{a_{q+i}}{m_{q+i}}$, with $a_{q+i} \in \mathbb{Z}^n$ and $m_{q+i} \in \mathbb{N} \setminus \{0\}$, then the set of equations is $a_j(x - x_0) = 0$, $j = q+1, \ldots, n$.

It is well-known that the coefficients of B^{-1} are quotients of determinants of submatrices of B. Since the determinant of B is a $n!$ sum of products of n elements of B, its size is bounded by $n \log n + kn$, and therefore, the sizes of the elements in B^{-1} are bounded by $\mathcal{O}(n \log n + nk)$, and the sizes of the elements in CB^{-1} are bounded by $\mathcal{O}(n \log n + nk)$.

Note finally that $x \in x_0 + \mathrm{lin}_{\mathbb{Q}}(T) \Leftrightarrow \bigwedge_{i=q+1,\ldots,n} a_i x \equiv \beta_i$. $\square$

9.2 Formal description of the algorithm QAffineHull

Algorithm QAffineHull

input: reduced NDD $\mathcal{A} = (Q, \Sigma_r^n, \delta, s_{init}, Q_F)$ representing set $S_{\mathcal{A}} \in \mathbb{Z}^n$

output: **set of** vector G and sequence $x_{s_1}, \ldots, x_{s_{|Q|}}$, such that $x_s \in S_{\mathcal{A}(s_{init} \to s)}$ for each state $s \in Q$, and $x_s + \mathrm{lin}_{\mathbb{Q}}(G) = \mathrm{aff}_{\mathbb{Q}}(S_{\mathcal{A}})$ for any $s \in Q_F$.

```
 1: $W := \emptyset$
 2: for each $s \in Q$ do $x_s := \bot$; od
 3: $B := \emptyset$
 4: for each $s \in Q$, for each $u \in \Sigma_r^n$ such that $\delta(s_{init}, u) = s$ do $W := W \cup \{(s, \langle u \rangle_r))\}$ od
 5: while $W \neq \emptyset$ do
 6:     $C := W$;
 7:     $W := \emptyset$;
 8:     for each $(s, x) \in C$ do
 9:         if $(x_s = \bot)$ then
10:             $x_s := x$;
11:             for each $s' \in Q$, for each $u \in \Sigma_r^n$ such that $\delta(s, u) = s'$ do
12:                 $W := W \cup \{(s', r \cdot x + \langle 0u \rangle_r)\}$;
13:             od
14:         fi
15:         $G := G \cup \{x - x_s\}$;
16:     od
17: od
18: $x_F := \bot$;
19: for each $s \in Q_F$ do
20:     if $x_F = \bot$ then $x_F := x_s$ fi ;
21:     $G := G \cup \{x_s - x_F\}$
22: od
23: return $(G, x_F)$;
```

9.3 Proof of Theorem 9

Lemma 14 *In algorithm* `QAffineHull`*, all pairs (s, x) added to W before the kth iteration of the main* **while** *-loop are such that $s \in Q$ and there exists $w \in L_{\mathcal{A}}(s_{init} \rightarrow s)$ with $\langle w \rangle_r = x$ and $|w| \leq k$.*

Proof. This is proved by induction on the number of iteration of the main **while** -loop.

Before the fi rst iteration, all pairs (s, x) added W are such that there exists $u \in \Sigma_r^n$ with $\delta(s_{init}, u) = s$ and $\langle u \rangle_r = x$.

Suppose the property holds after k iterations and let $W^{(i)}$ denotes the set W after i iterations. If (s', x') is added to W during the $k + 1$ iteration, by construction, there exists $(s, x) \in W^{(i)}$ and $u \in \Sigma_r^n$ such that $\delta(s, u) = s'$ and $x' = r \cdot x + \langle 0u \rangle_r$. By inductive hypothesis, there exists $w \in L_{\mathcal{A}}(s_{init} \rightarrow s)$ such that $\langle w \rangle_r = x$ and $|w| \leq k$. Therefore, $x' = r \cdot \langle w \rangle_r + \langle 0u \rangle_r$ and by defi nition of the encoding scheme, $x' = \langle wu \rangle_r$, with $wu \in L_{\mathcal{A}}(s_{init}, s' \rightarrow)$ and $|wu| \leq k + 1$. □

Lemma 15 *In algorithm* `QAffineHull`*, for all states s, if the smallest non-empty path from s_{init} to s is of length l_s, then x_s is set during the l_s iteration of the* **while** *-loop and for all states s' and $u \in \Sigma_r^n$ with $\delta(s, u) = s'$, $(s', r \cdot x_s + \langle 0u \rangle_r)$ is added to W during the l_s iteration.*

Proof. We prove the claim by induction.

From Lemma 14, for all pair (s, x) stored in W, $x \in S_{\mathcal{A}(s_{init} \rightarrow s)}$. Therefore, for all states s, x_s is not set after i iterations if $i < l_s$.

By inspection the claim holds for the states reachable from s_{init} via a path of length 1. If the property holds for states reachable via paths of length of at most k, then, for all states s_{k+1} reachable via a path of length $k + 1$ but not via smaller paths, there is a state s_k such that $(\delta(s_k, u) = s_{k+1}$ for some $u \in \Sigma_r^n$ and the smallest path reaching s_k is of length k. By inductive hypothesis, $(s_{k+1}, r \cdot x_{s_k} + \langle 0u \rangle_r)$ is added to W during iteration k, and therefore, $x_{s_{k+1}}$ is set at iteration $k+1$ and for all states s' and $u' \in \Sigma_r^n$ with $\delta(s_{k+1}, u') = s'$, $(s', r \cdot x_{s_{k+1}} + \langle 0u' \rangle_r)$ is added at iteration $k + 1$. □

Lemma 16 *Let $l_{\min} \leq |Q|$ be the smallest positive integer such that for all states $s \in Q$, there exists an encoding w_s such that $\hat{\delta}(s_{init}, w_s) = s$ with $|w_s| \leq l_{\min}$.*

The algorithm `QAffineHull` *terminates after $l_{\min} + 1$ iterations of the main* **while** *-loop, and, if (G, x_F) =* `QAffineHull`*($\mathcal{A}$), then,*

$$x_F + \text{lin}_{\mathbb{Q}}(G) = \text{aff}_{\mathbb{Q}}(S_{\mathcal{A}}).$$

Proof. Thanks to Lemma 15, we note that the algorithm terminates after $l_{\min} + 1$ iterations since for all states $s \in Q$, x_s is set only once and pairs are added to W only when some x_s is set.

We now prove that $\text{aff}_{\mathbb{Q}}(S_{\mathcal{A}}) = x_F + \text{lin}_{\mathbb{Q}}(G)$. Note fi rst that $x_F = x_{s_F}$ for some state $s_F \in Q_F$, and therefore, thanks to Lemma 14, $x_F \in S_{\mathcal{A}}$.

– By construction, for all $g \in G$, either $g = x - x_s$ for some $s \in Q$ such that (s, x) has been added to W, or $g = x_s - x_F$ such that $s \in Q_F$.

- If $g = x - x_s$, then, from Lemma 14, $x, x_s \in S_{\mathcal{A}(s_{init} \to s)}$, and from Lemma 8, $x - x_s + x_F \in \mathrm{aff}_{\mathbb{Q}}(S_{\mathcal{A}})$, i.e., $x_F + g \in \mathrm{aff}_{\mathbb{Q}}(S_{\mathcal{A}})$.
- If $g = x_s - x_F$, then $x_s \in S_{\mathcal{A}}$ and it is immediate that $x_F + g \in \mathrm{aff}_{\mathbb{Q}}(S_{\mathcal{A}})$.

We deduce that for each $g \in G$, $x_F + g \in \mathrm{aff}_{\mathbb{Q}}(S_{\mathcal{A}})$, and therefore, we have

$$\mathrm{aff}_{\mathbb{Q}}(S_{\mathcal{A}}) \supseteq x_{s_F} + \mathrm{lin}_{\mathbb{Q}}(G) \tag{5}$$

– We prove that for all $s \in Q$ and $w \in L_{\mathcal{A}}(s_{init} \to s)$ with $|w| \geq 1$, $\langle w \rangle_r \in x_s + \mathrm{lin}_{\mathbb{Q}}(G)$ by induction on the size of w. If $|w| = 1$, then if $\hat{\delta}(s_{init}, w) = s$, then $(s, \langle w \rangle_r)$ is stored in W before the first iteration, and therefore, by inspection, $x_s - \langle w \rangle_r \in G$.

Suppose the property holds for encodings of length smaller of equal to k, and let $w_{k+1} = w_k u$ with $w_k \in (\Sigma_r^n)^k$, $u \in \Sigma_r^n$, and let $s_{k+1} = \hat{\delta}(s_{init}, w_{k+1})$. By definition, there is a state $s_k \in Q$ such that $s_k = \hat{\delta}(s_{init}, w_k)$. Since s_k and s_{k+1} are reachable via non-zero paths, thanks to Lemma 15, x_{s_k} and $x_{s_{k+1}}$ are set and by inspection, $(s_{k+1}, r \cdot x_{s_k} + \langle 0u \rangle_r)$ is added to W, and we have

$$r \cdot x_{s_k} + \langle 0u \rangle_r - x_{s_{k+1}} \in G. \tag{6}$$

In addition, by inductive hypothesis, we have

$$\langle w_k \rangle_r \in x_{s_k} + \mathrm{lin}_{\mathbb{Z}}(G). \tag{7}$$

Also, by definition of the encoding scheme,

$$\langle w_k u \rangle_r = r \cdot \langle w_k \rangle_r + \langle 0u \rangle_r \tag{8}$$

Combining (6), (7) and (8), we conclude that

$$\langle w_k u \rangle_r \in x_{s_{k+1}} + \mathrm{lin}_{\mathbb{Q}}(G). \tag{9}$$

By definition, for all $x \in S_{\mathcal{A}}$, there exists an encoding w and a state $s_F \in Q_F$ such that $\langle w \rangle_r = x$ and $\hat{\delta}(s_{init}, w) = s_F$. From above, we have that $\langle w \rangle_r \in x_{s_F} + \mathrm{lin}_{\mathbb{Z}}(G)$. By inspection, we see that $x_{s_F} - x_F \in G$, and so, we conclude that $\langle w \rangle_r \in x_F + \mathrm{lin}_{\mathbb{Q}}(G)$. □

We can now prove the correctness of `QAffineHull` and its complexity.

Theorem 17 *Let $l_{\min} \leq |Q|$ be the smallest positive integer such that for all states $s \in Q$, there exists an encoding w_s such that $\hat{\delta}(s_{init}, w_s) = s$ with $|w_s| \leq l_{\min}$. Let $x_F \in \mathbb{Z}^n$ and $G \subseteq \mathbb{Z}^n$ such that $(G, x_F) =$ `QAffineHull`$(\mathcal{A})$. We have*

– $x_F + \mathrm{lin}_{\mathbb{Q}}(G) = \mathrm{aff}_{\mathbb{Q}}(S_{\mathcal{A}})$,
– $|G| \leq |Q| \cdot \Sigma_r^n$,
– *The time complexity of* `QAffineHull` *is* $\mathcal{O}(|Q| \cdot |\Sigma_r^n| \cdot n)$,
– *The size of the numbers in G are bounded by* $\mathcal{O}(l_{\min})$.

Proof. The first assertion is a direct consequence of Lemma 16. Then, we note that one explores transitions outgoing from a state s only when s is met for the first time, and each state is reached after at most $l_{\min}$ iterations according to Lemma 16, and therefore numbers in x_s are bounded $r^{l_{\min}}$. Since all vectors $g \in G$ are such that either $g = x - x_s$ such that (s, x_s) as been added to W or $g = x_s - x_F$ for some accepting state s, the number of vectors stored in G bounded by $\mathcal{O}(|Q| \cdot |\Sigma_r^n|)$ and the sizes of numbers in G are bounded by $\mathcal{O}(|l_{\min}|)$. □

9.4 Formal description of the algorithm `ZAffineHull`

In the description below, the function `DistanceToFinal` takes a reduced NDD $\mathcal{A}$ as input and returns the smallest number $d_{\min}$ such that for each state s, there is a path of length smaller or equal to $d_{\min}$ from s to an accepting state.

```
**Algorithm** ZAffineHull
**input**:  reduced NDD $\mathcal{A} = (Q, \Sigma_r^n, \delta, s_{init}, Q_F)$ representing set $S_{\mathcal{A}} \in \mathbb{Z}^n$
**output**: vector $g_0$, **set of** vectors $G \subseteq \mathbb{Z}^n$ such that $g_0 \in S_{\mathcal{A}}$, and $g_0 + \mathrm{lin}_{\mathbb{Z}}(G) = \mathrm{aff}_{\mathbb{Z}}(S_{\mathcal{A}})$
 1: $(G_{pre}, x_F)$ := QAffineHull($\mathcal{A}$);
 2: $\overline{G_{pre}}$ := GetTriangZBasis($G_{pre}$);
 3: $W$ := $\emptyset$;
 4: for each $s \in Q$ do $\Gamma_s$ := $\emptyset$ od ;
 5: $d_{\min}$ := DistanceToFinal($\mathcal{A}$);
 6: for each $s' \in Q$, for each $u \in \Sigma_r^n$ such that $\delta(s_{init}, u) = s'$ do $W$ := $W \cup (s, \langle w \rangle_r)$ od ;
 7: while $W \neq \emptyset$ do
 8:     $C$ := $W$; $W$ := $\emptyset$;
 9:     for each $(s, x) \in C$ do
10:         if $x_s = \perp$ then
11:             $x_s$ := $x$;
12:             For each $s' \in Q$, $u \in \Sigma_r^n$ with $\delta(s, u) = s'$ do $W$ := $W \cup \{(s', r \cdot x + \langle 0u \rangle_r)\}$ od
13:         fi
14:         Let $\gamma_1, \ldots, \gamma_{|\overline{G_{pre}}|}$ such that $\langle u \rangle_r - x_{s'} = \sum_{g_i \in \overline{G_{pre}}} \gamma_i g_i$;
15:         $\Gamma'_s$ := UpdateTriangZm($r$, $d_{\min}$, $\Gamma_s$, $(\gamma_1 \mod r^{d_{\min}}, \ldots, \gamma_{|\overline{G_{pre}}|} \mod r^{d_{\min}})$);
16:         if $\Gamma'_s \neq \Gamma_s$ then
17:             $\Gamma_s$ := $\Gamma'_s$;
18:             for each $s' \in Q$, $u \in \Sigma_r^n$ such that $\delta(s, u) = s'$ do
19:                 $W$ := $W \cup \{(s', r \cdot x + \langle 0u \rangle_r)\}$
20:             od
21:         fi
22:     od ;
23: od ;
24: $\Gamma$ := $\emptyset$; $G$ := $\emptyset$;
25: for each $s \in Q_F$, for each $c \in \Gamma_s$ do $\Gamma$ := UpdateTriangZm($r$, $d_{\min}$, $\Gamma$, $c$) od ;
26: for each $(\gamma_1, \ldots, \gamma_{|\overline{G_{pre}}|}) \in \Gamma$ do $G$ := $G \cup \{\sum_{g_i \in \overline{G_{pre}}} \gamma_i g_i\}$ od ;
27: for each $g \in \overline{G_{pre}}$ do $G$ := $G \cup \{r^{d_{\min}} g\}$ od ;
28: for each $s \neq s_F \in Q_F$ with $s_F \in Q_F$ do $G$ := $G \cup \{x_s - x_{s_F}\}$ od ;
29: return $(G, x_{s_F})$;
```

9.5 Proof of Lemma 11

Lemma 18 *Let $d_{\min}$ be the smallest positive integer such that for all states $s \in Q$, there exists an encoding w_s such that $\hat{\delta}(s, w_s) \in Q_F$ with $|w_s| \leq d_{\min}$. Let $M \subseteq \mathbb{Z}^n$ such that $\mathrm{aff}_{\mathbb{Z}}(S_{\mathcal{A}}) = x_F + M$ for some $x_F \in \mathbb{Z}^n$.*

If (G, x_F) = `QAffineHull`*($\mathcal{A}$), then*

- *for all $s \in Q$, for all $x_1, x_2 \in S_{\mathcal{A}(s_{init} \to s)}$, $x_1 - x_2 \in \mathrm{lin}_{\mathbb{Z}}(G)$, and,*
- *for all $g \in G$, $r^{d_{\min}} g \in M$.*

Proof.

- We prove that for all $s \in Q$ and $w \in L_{\mathcal{A}}(s_{init} \to s)$ with $|w| \geq 1$, $\langle w \rangle_r \subseteq x_s + \mathrm{lin}_{\mathbb{Z}}(G)$ by induction on the size of w, where x_s is the vector associated to s in `QAffineHull`.
  If $|w| = 1$ and $\hat{\delta}(s_{init}, w) = s$, then $(s, \langle w \rangle_r)$ is stored in W before the first iteration, and therefore, $x_s - \langle w \rangle_r \in G \in \mathrm{lin}_{\mathbb{Z}}(G)$.
  Suppose the property holds for encodings of length smaller of equal to k, and let $w_{k+1} = w_k u$ with $w_k \in (\Sigma_r^n)^k$, $u \in \Sigma_r^n$, and let $s_{k+1} = \hat{\delta}(s_{init}, w_{k+1})$. By definition, there is a state $s_k \in Q$ such that $s_k = \hat{\delta}(s_{init}, w_k)$. Since s_k and s_{k+1} are reachable via non-zero paths, thanks to Lemma 15, x_{s_k} and $x_{s_{k+1}}$ are set and by inspection, $(s, r \cdot x_{s_k} + \langle 0u \rangle_r)$ has been added to W, and we have

$$r \cdot x_{s_k} + \langle 0u \rangle_r - x_{s_{k+1}} \in G. \tag{10}$$

  In addition, by inductive hypothesis, we have

$$x_{s_k} - \langle w_k \rangle_r \in \mathrm{lin}_{\mathbb{Z}}(G). \tag{11}$$

  Finally, by definition of the encoding scheme,

$$\langle w_k u \rangle_r = r \cdot \langle w_k \rangle_r + \langle 0u \rangle_r \tag{12}$$

  Combining (10), (11) and (12), we conclude that

$$\langle w_k u \rangle_r - x_{s_{k+1}} \in \mathrm{lin}_{\mathbb{Z}}(G). \tag{13}$$

  If $x_1, x_2 \in S_{\mathcal{A}(s_{init} \to s)}$, by definition, there exist $w_1, w_2 \in L_{\mathcal{A}}(s_{init} \to s)$ with $\langle w_1 \rangle_r = x_1$ and $\langle w_2 \rangle_r = x_2$. From above, we deduce that $x_1, x_2 \in x_s + \mathrm{lin}_{\mathbb{Q}}(G)$, and therefore, $x_1 - x_2 \in \mathrm{lin}_{\mathbb{Q}}(G)$.
- For all $g \in G$, either $g = x - x_s$ for some state s such that (s, x) was added to W, or $g = x_s - x_{s_F}$ for some final states s and s_F. Thanks to Lemma 14, for all states s, if $x_s \neq \bot$ then $x_s \in S_{\mathcal{A}(s_{init} \to s)}$.
  - If $g = x - x_s$, then, $x, x_s \in S_{\mathcal{A}(s_{init} \to s)}$, and from Lemma 10, $r^k(x - x_s) + x_{s'_F} \in \mathrm{aff}_{\mathbb{Z}}(S_{\mathcal{A}})$ where k is the length of the shortest path from s to a final state, and s'_F is the corresponding final state. By definition, $k \leq d_{\min}$, $x_{s'_F}, x_{s_F} \in S_{\mathcal{A}}$. So, $r^{d_{\min}} g + x_{s_F} \in \mathrm{aff}_{\mathbb{Z}}(S_{\mathcal{A}})$.
  - If $g = x_s - x_{s_F}$, then $x_s \in S_{\mathcal{A}}$. So, it is immediate that $x_{s_F} + g \in \mathrm{aff}_{\mathbb{Z}}(S_{\mathcal{A}})$.

  We deduce that for each $g \in G$, $x_{s_F} + r^{d_{\min}} g \in \mathrm{aff}_{\mathbb{Z}}(S_{\mathcal{A}})$, and by definition, $r^{d_{\min}} g \in M$.

□

9.6 Proof of Theorem 12

Lemma 19 *For each pair (s, x) added to W in the algorithm* `ZAffineHull`*, we have $x \in S_{\mathcal{A}(s_{init} \to s)}$ and $x - x_s = \mathrm{lin}_{\mathbb{Z}}(\overline{G_{pre}})$.*

Proof. Proving that $x \in S_{\mathcal{A}(s_{init} \to s)}$ is done by induction on the iteration of the main **while** -loop, as it is done for the proof of Lemma 14. The second assertion is a consequence of Lemma 11 since $x, x_s \in S_{\mathcal{A}(s_{init} \to s)}$. $\square$

In the following lemmas, for each $s \in Q$, we denote by G_s the set $\{\sum_{g_i \in \overline{G_{pre}}} c[i] \cdot g_i \mid c \in \Gamma_s\}$ when the algorithm `ZAffineHull` terminates. Also, for each s, we denote by M_s the $\mathbb{Z}$-module such that $\mathrm{aff}_{\mathbb{Z}}(s) = x_s + M_s$ for some $x_s \in S_{\mathcal{A}(s_{init} \to s)}$ and we $p = |\overline{G_{pre}}|$.

Lemma 20 *For each pair (s, x) added to W,*

$$x - x_s \in \mathrm{lin}_{\mathbb{Z}}(G_s \cup r^{d_{\min}} \overline{G_{pre}}).$$

Proof. From Lemma 19, there exist $c, b \in \mathbb{Z}^p$ with $c[i] \in \{0, \ldots r^{d_{\min}} - 1\}$ and

$$x - x_s = \sum_{g_i \in \overline{G_{pre}}} (c[i] + b[i] r^{d_{\min}}) \cdot g_i. \tag{14}$$

By inspection and thanks to Proposition 7, $c \in \mathrm{lin}_{\mathbb{Z}_{r^{d_{\min}}}}(\Gamma_s)$, and therefore, by definition,

$$\sum_{g_i \in \overline{G_{pre}}} c[i] \cdot g_i \in \mathrm{lin}_{\mathbb{Z}}(G_s + r^{d_{\min}} \overline{G_{pre}}). \tag{15}$$

From (14) and (15), we conclude that $x - x_s \in \mathrm{lin}_{\mathbb{Z}}(G_s \cup r^{d_{\min}} \overline{G_{pre}})$. $\square$

Lemma 21 *When the algorithm* `ZAffineHull` *terminates, for each $s \in Q$, for each $g \in G_s$, we have*

- *$g \in \mathrm{lin}_{\mathbb{Z}}(M_s \cup r^{d_{\min}} \overline{G_{pre}})$, and*
- *$r \cdot g \in \mathrm{lin}_{\mathbb{Z}}((G_{s'} \cup r^{d_{\min}} \overline{G_{pre}})$, for all $s' \in Q$ with $\delta(s, u) = s'$ for some $u \in \Sigma_r^n$.*

Proof. – By inspection, Γ_s is updated via calls to the function `UpdateTriangZm` with r, $d_{\min}$, the current value of Γ_s and c as arguments, such that $c \in \mathbb{Z}^p$, $c[i] \in \{0, \ldots, r^{d_{\min}} - 1\}$ and there exists a pair (s, x) added to W with $x - x_s = \sum_{g_i \in \overline{G_{pre}}} (c[i] + b[i] r^{d_{\min}}) g_i$, with $b \in \mathbb{Z}^p$ and $x \in S_{\mathcal{A}(s_{init} \to s)}$ thanks to Lemma 19. So, we have

$$\sum_{g_i \in \overline{G_{pre}}} c[i] \cdot g_i = x - x_s + \sum_{g_i \in \overline{G_{pre}}} b[i] r^{d_{\min}} \cdot g_i. \tag{16}$$

Therefore, thanks to Lemma 7, for all $c \in \Gamma_s$, we have

$$\sum_{g_i \in \overline{G_{pre}}} c[i] \cdot g_i = \sum_j \lambda_j (x_j - x_s) + \sum_{g_i \in \overline{G_{pre}}} b[i] r^{d_{\min}} \cdot g_i, \text{ with } b \in \mathbb{Z}^p. \tag{17}$$

By definition, since for each $g \in G_s$, $g_s = \sum_{g_i \in \overline{G_{pre}}} c[i] \cdot g_i$ for some $c \in \Gamma_s$, we conclude that

$$g \in \mathrm{lin}_{\mathbb{Z}}(M_s \cup r^{d_{\min}} \overline{G_{pre}}). \tag{18}$$

– By inspection, whenever Γ_s is modified, the modification resulted of a call to the function `UpdateTriangZm` with r, $d_{\min}$, the current value of Γ_s and c as arguments, such that $c \in \mathbb{Z}^k$, $c[i] \in [0, r^{d_{\min}}]$ and there exists a pair (s, x) added to W with

$$x - x_s = \sum_{g_i \in \overline{G_{pre}}} (c[i] + b[i] r^{d_{\min}}) g_i, \tag{19}$$

with $b \in \mathbb{Z}^p$ and $x \in S_{\mathcal{A}(s_{init} \to s)}$ thanks to Lemma 19.
By inspection, $(s', r \cdot x + \langle 0u \rangle_r)$ is then added to W, and also, $(s', r \cdot x_s + \langle 0u \rangle_r)$ has been previously added to W. So, thanks to Lemma 20, we deduce that

$$r \cdot x_s + \langle 0u \rangle_r - x_{s'} \in \mathrm{lin}_{\mathbb{Z}}(G_{s'} \cup r^{d_{\min}} G_{pre}), \tag{20}$$

and

$$r \cdot x + \langle 0u \rangle_r - x_{s'} \in \mathrm{lin}_{\mathbb{Z}}(G_{s'} \cup r^{d_{\min}} G_{pre}). \tag{21}$$

From (20) and (21), we find that

$$r \cdot (x - x_s) \in \mathrm{lin}_{\mathbb{Z}}(G_{s'} \cup r^{d_{\min}} \overline{G_{pre}}). \tag{22}$$

Therefore, from (19), we deduce that

$$\sum_{g_i \in \overline{G_{pre}}} (r \cdot c[i]) g_i = \sum_{g'_i \in G_{s'}} \lambda_i \cdot g'_i + \sum_{g_i \in \overline{G_{pre}}} b'[i] r^{d_{\min}} \cdot g_i, \tag{23}$$

with $\lambda_i \in \mathbb{Z}$ and $b' \in \mathbb{Z}^p$.
By definition and thanks to Proposition 7, for all $g \in G_s$, we have

$$g = \sum_j \zeta_j (\sum_{g_i \in \overline{G_{pre}}} c_j[i] \cdot g_i) + \sum_{g_i \in \overline{G_{pre}}} b''[i] r^{d_{\min}} \cdot g_i, \tag{24}$$

with $\zeta_j \in \mathbb{Z}$, $b'' \in \mathbb{Z}^p$ and such that Γ_s has been updated by c_j.
From (23) and (24), we conclude that $r \cdot g \in \mathrm{lin}_{\mathbb{Z}}(G_{s'} \cup r^{d_{\min}} \overline{G_{pre}})$. □

Lemma 22 *Let $x \in \mathbb{Z}^n$, $G \subseteq \mathbb{Z}^n$ such that $(G, x_F) =$ `ZAffineHull`$(\mathcal{A})$.
For all $g \in G$, $x_F + g \in \mathrm{aff}_{\mathbb{Z}}(S_{\mathcal{A}})$.*

Proof. By construction, $x_F = x_{s_F}$ for some $s \in Q_F$, and therefore, from Lemma 11, $x_F \in S_{\mathcal{A}}$.
Regarding g there are 3 possibilities.

– $g = r^{d_{\min}} g_i$ for some $g_i \in \overline{G_{pre}}$. Thanks to Lemma 11, $x + g \in \mathrm{aff}_{\mathbb{Z}}(S_{\mathcal{A}})$.
– $g = x_s - x$ for some $s \in Q_F$. So, $x + g \in S_{\mathcal{A}} \subseteq \mathrm{aff}_{\mathbb{Z}}(S_{\mathcal{A}})$.
– $g = \sum_{g_i \in \overline{G_{pre}}} c[i] g_i$ for some $c \in \Gamma$. Thanks to Proposition 7 and by definition of Γ,

$$g = \sum_j \lambda_j \sum_{g_i \in \overline{G_{pre}}} c_j[i] \cdot g_i + \sum_{g_i \in \overline{G_{pre}}} (b[i] r^{d_{\min}}) \cdot g_i, \tag{25}$$

where $c_j \in \Gamma_{s_j}$ for some $s_j \in Q_F$, $b \in \mathbb{Z}^p$ and $\lambda_j \in \mathbb{Z}$.

Thanks to Lemma 21, we conclude that

$$g = \sum_j \lambda_j (x_{1,j} - x_{2,j}) + \sum_{g_i \in \overline{G_{pre}}} (b'[i] r^{d_{\min}}) \cdot g_i, \tag{26}$$

where $\lambda_j \in \mathbb{Z}$, $b' \in \mathbb{Z}^p$ and $x_{1,j}, x_{2,j} \in S_{\mathcal{A}}$, and from above, we deduce that $x + g \in \mathrm{aff}_{\mathbb{Z}}(S_{\mathcal{A}})$.

□

Lemma 23 *When the algorithm* `ZAffineHull` *terminates, we have that for all $s \in Q$ and for all encoding $w \in L_{\mathcal{A}}(s_{init} \to s)$,*

$$\langle w \rangle_r = x_s + \mathrm{lin}_{\mathbb{Z}}(G_s \cup r^{d_{\min}} \overline{G_{pre}}).$$

Proof. The proof is done on the size of $|w|$.

If $|w| = 1$, then, $(s, \langle w \rangle_r)$ is added to W at line 6 in the formal description of `ZAffineHull`, and thanks to Lemma 20,

$$\langle w \rangle_r - x_s \in \mathrm{lin}_{\mathbb{Z}}(G_s \cup r^{d_{\min}} \overline{G_{pre}}).$$

Suppose the claim holds for all w with $|w| \leq k$, and let $w_k u \in L_{\mathcal{A}}(s_{init} \to s)$ with $|w_k| = k$ and $u \in \Sigma_r^n$. Let $s_k = \hat{\delta}(s_{init}, w_k)$. By inductive hypothesis,

$$\langle w_k \rangle_r \in x_{s_k} + \mathrm{lin}_{\mathbb{Z}}(G_{s_k} \cup r^{d_{\min}} \overline{G_{pre}}). \tag{27}$$

Therefore, we have

$$\langle w_k u \rangle_r \in r \cdot x_{s_k} + \langle 0u \rangle_r + \mathrm{lin}_{\mathbb{Z}}(r G_{s_k} \cup r^{d_{\min}+1} \overline{G_{pre}}). \tag{28}$$

By hypothesis, $\delta(s_k, u) = s$, and by inspection, $(s, r \cdot x_s + \langle 0u \rangle_r)$ has been added to W. So, from Lemma 20, we have

$$r \cdot x_{s_k} + \langle 0u \rangle_r \in x_s + \mathrm{lin}_{\mathbb{Z}}(G_s \cup r^{d_{\min}} \overline{G_{pre}}). \tag{29}$$

In addition, from Lemma 21, we have

$$r G_{s_k} \subseteq \mathrm{lin}_{\mathbb{Z}}(G_s \cup r^{d_{\min}} \overline{G_{pre}}). \tag{30}$$

Combining (28), (29) and (30), we fi nd

$$\langle w_k u \rangle_r \in x_s + \mathrm{lin}_{\mathbb{Z}}(G_s \cup r^{d_{\min}} \overline{G_{pre}}). \tag{31}$$

□

Lemma 24 *Let $x \in \mathbb{Z}^n$, $G \subseteq \mathbb{Z}^n$ such that $(G, x) =$* `ZAffineHull`*$(\mathcal{A})$.*

For all $y \in S_{\mathcal{A}}$, $y \in x + \in \mathrm{lin}_{\mathbb{Z}}(G)$.

Proof. By definition, if $y \in S_{\mathcal{A}}$, then there is an encoding $w \in (\Sigma_r^n)^*$ such that $y = \langle w \rangle_r$ and $\hat{\delta}(s_{init}, w) = s_F$ for some $s_F \in Q_F$. Therefore, thanks to Lemma 23, when `ZAffineHull` terminates, we have

$$\langle w \rangle_r \in x_{s_F} + \mathrm{lin}_{\mathbb{Z}}(G_{s_F} \cup r^{d_{\min}} \overline{G_{pre}}).$$

By definition of G, $G_{s_F} \subseteq G$, $r^{d_{\min}} \overline{G_{pre}} \subseteq G$, and $x - x_{s_F} \in G$, and we conclude that

$$y = \langle w \rangle_r \in x + \mathrm{lin}_{\mathbb{Z}}(G).$$

□

Theorem 25 *Let $l_{\min}, d_{\min} \leq |Q|$ be the smallest positive integers such that for all states $s \in Q$, there exist encodings w_l, w_d such that $\hat{\delta}(s_{init}, w_l) = s$ with $|w_l| \leq l_{\min}$ and $\hat{\delta}(s, w_d) = s_F$ for some $s_F \in Q_F$ with $|w_d| \leq d_{\min}$. Let $x_F \in \mathbb{Z}^n$, $G \subseteq \mathbb{Z}^n$ such that $(G, x_F) =$ `ZAffineHull`$(\mathcal{A})$. We have*

- $x_F + \mathrm{lin}_{\mathbb{Z}}(G) = \mathrm{aff}_{\mathbb{Z}}(S_{\mathcal{A}})$,
- $|G| \leq |Q| + 2n$,
- *Size of numbers in G are bounded by $\mathcal{O}(n \cdot \log(\sqrt{n}) \cdot l_{\min} + d_{\min})$,*
- *The time complexity of* `ZAffineHull` *is $\mathcal{O}(n^3 \cdot |Q| \cdot |\Sigma_r^n| \cdot (\log(\sqrt{n}) \cdot l_{\min} + d_{\min}^2))$.*

Proof. The first assertion is a direct consequence of Lemmas 22 and 24. From Theorem 9, $|G_{pre}| \leq |Q| \cdot \Sigma_r^n$ and the size of the numbers are bounded by $\mathcal{O}(l_{\min})$. From Proposition 5, the time complexity of the call `GetTriangZBasis` is therefore $\mathcal{O}(n^3 \log(\sqrt{n})|Q| \cdot |\Sigma_r^n| \cdot l_{\min})$ and the sizes of numbers in $\overline{G_{pre}}$ are bounded by $\mathcal{O}(l_{\min} \cdot n \log(\sqrt{n}))$.

The call `DistanceToFinal` can be complete in time proportional to $|Q| \cdot |\Sigma_r^n|$ by performing a backward breadth first search from the final states, and by definition.

According to Proposition 7, for each state s, the set Γ_s is updated at most $n \cdot d_{\min}$ times (since $\mathrm{rank}(\Gamma_s) \leq n d_{\min}$ and it is decreased whenever $\Gamma_s' \neq \Gamma_s$) and therefore, the number of pairs $(s, \langle w \rangle_r)$ added to W is bounded by $\mathcal{O}(n \cdot d_{\min} \cdot |Q| \cdot |\Sigma_r^n|)$. So, the number of calls to `UpdateTriangZm` at line 15 is bounded by $\mathcal{O}(n \cdot d_{\min} \cdot |Q| \cdot |\Sigma_r^n|)$. The time spend in the main **while** -loop is therefore bounded by $\mathcal{O}(n^3 \cdot d_{\min}^2 \cdot |Q| \cdot |\Sigma_r^n|)$.

Finally, for all $g \in G$, either $g = x_s - x_{s_F}$ for some accepting states s and s_F, $g = r^{d_{\min}} g_i$ with $g_i \in \overline{G_{pre}}$ or $g = \sum_{g_i \in \overline{G_{pre}}} \gamma_i g_i$ for $(\gamma_1, \ldots, \gamma_{|\overline{G_{pre}}|}) \in \Gamma$. So, $|G| \leq |Q| + 2n$ and the size of the numbers in G are bounded by $\mathcal{O}(l_{\min} \cdot n \cdot \log(\sqrt{n}) + d_{\min})$.

□

9.7 Formulas of sets used in the experimental results

The following formulas correspond to the sets $S_1, \ldots, S_{12}$ presented in Section 7.

S_1	$3x_0+2x_1-5x_2+6x_3-10x_4+3x_5+2x_6=2 \wedge 1x_0+1x_1+3x_2+2x_3+7x_4+15x_5-20x_6=2 \wedge 10x_0+20x_1+30x_2=0$
S_2	$3x_0+2x_1-5x_2+6x_3-10x_4+3x_5=2 \wedge 1x_0+1x_1+3x_2+2x_3+7x_4+15x_5=2 \wedge 10x_0+20x_1+30x_2=0 \wedge x_0+2x_1=0 \mod 5$
S_3	$(\, 21x_0+3x_1-5x_2+2x_3+4x_4-1x_5=24 \wedge 5x_0+1x_1-2x_2-2x_3+6x_4+3x_5=11 \wedge x_0=0 \mod 128 \wedge x_0+x_1+x_2=3 \mod 49\,)$
S_4	$11x_0+5x_1+9x_2+19x_3+5x_4+6x_5=0 \mod 33 \wedge 2x_0+1x_1+3x_2+4x_3+6x_4+2x_5=0 \mod 33 \wedge 5x_0+21x_1+1x_2+8x_3+0x_4+1x_5=0 \mod 33$
S_5	$x_0+x_1-x_2+3x_3+4x_4+x_5+5x_6+x_7+3x_8+x_9=2 \mod 7 \wedge x_0-3x_2=3 \wedge 4x_3-5x_4=0 \wedge x_7+x_8=10 \mod 20 \wedge 10x_2-5x_9=1 \mod 16 \wedge 2x_1+3x_5+x_6=12 \wedge x_7=0 \mod 3$
S_6	$1x_0+2x_1-1x_4+1x_5=3 \wedge 1x_2+2x_4-1x_9=2 \wedge 2x_1+1x_3+2x_5+1x_6-1x_7=5 \wedge 1x_1+1x_4+2x_6+8x_8+1x_{10}=12 \mod 13 \wedge 2x_0+3x_2+1x_4+1x_7+8x_8+4x_9=0 \mod 5$
S_7	$((x_0+2x_1+3x_2-4x_5=2 \wedge 3x_0+3x_2+1x_3+5x_4-6x_5+3x_6 \leq 10\,) \vee (4x_0+5x_2+2x_3-6x_4+3x_5+4x_6=2)) \wedge x_0+1x_1+1x_3+3x_4+4x_5=4$
S_8	$(\, 3x_0+1x_1+8x_2+1x_4+1x_5+8x_6=2 \vee 3x_0+1x_1+8x_2+1x_4+1x_5+8x_6=27 \vee 3x_0+1x_1+8x_2+1x_4+1x_5+8x_6=52\,) \wedge 4x_0+7x_1+2x_2-8x_3-1x_4+4x_5=0 \wedge x_0 \geq 10 \wedge x_1 \geq 15$
S_9	$12x_0-9x_1+11x_3-2x_4=5 \mod 28 \wedge 3x_1+x_2+2x_3+5x_5=1 \mod 30 \wedge 1x_0+2x_1+5x_2+4x_4+1x_5=0 \wedge (x_0 \leq -10 \vee x_0 \geq 20) \wedge (x_0+x_1+x_2+x_3+x_4+3x_5 \geq 0 \vee x_0+x_1+x_2+x_3+x_4+3x_5 \leq 50\,)$
S_{10}	$3x_1+2x_2+x_3+2x_6=2 \mod 36 \wedge x_1-6x_2+x_4+x_6=0 \wedge (x_1 \geq 10 \vee x_1 \leq 10) \wedge (x_2+x_3=20 \vee x_2+x_3 \leq -10) \wedge (x_1+4x_2-10x_5 \leq 0)$
S_{11}	$2x_0+3x_1+15x_2+11x_4+6x_5=10 \mod 20 \wedge 1x_0+x_1-6x_2+1x_3+x_4=0 \wedge (x_0+4x_1-x_2+x_3=10 \vee x_0+4x_1-x_2+x_3=18 \vee x_0+4x_1-x_2+x_3=32) \wedge (x_1+4x_2-10x_5 \leq 10)$
S_{12}	$2x_0+3x_1+15x_2+11x_4+6x_5=3 \wedge 1x_0+x_1-6x_2+x_3+x_4=0 \wedge (\, x_0+4x_1+6x_5=3 \mod 4 \vee -x_2+x_3+3x_4=10 \mod 16 \vee x_0+5x_1+6x_2+3x_4=2 \vee x_0=0\,)$