Exotic Semiring Constraints

(Extended Abstract)

Michael Codish* Yoav Fekete* Carsten Fuhs† Jürgen Giesl‡
Johannes Waldmann§

Abstract

Exotic semiring constraints arise in a variety of applications and in particular in the context of automated termination analysis. We propose two techniques to solve such constraints: (a) to model them using Boolean functions and integer linear arithmetic and solve them using an SMT solver (QF_LIA, in certain cases also QF_IDL); and (b) to seek finite domain solutions by applying unary bit-blasting and solve them using a SAT solver. In this note, we show the structure of such systems of constraints, and report on the performance of SMT solvers and SAT encodings when solving them. In particular, we observe that good results are obtained by unary bit-blasting, improving on previous proposals to apply binary bit-blasting. Moreover, our results indicate that, for our benchmarks, unary bit-blasting leads to better results than the ones directly obtained by an SMT solver.

1 Introduction

Exotic semirings [DK09] are idempotent semirings where the elements are certain subsets of numbers (possibly including $-\infty$ and/or $+\infty$) and where the sum and product operations are min, max, or $+$. For example, in the *tropical semiring* the elements are $\mathbb{N} \cup \{+\infty\}$ and the sum and product operations are min and $+$, respectively. In the *arctic semiring* the elements are $\mathbb{N} \cup \{-\infty\}$ and the sum and product operations are max and $+$, respectively. The tropical semiring is allegedly named as such to honor Imre Simon, one of its pioneers, who comes from tropical Brazil. Correspondingly, in [Goo98] the arctic semiring is named as such because arctic latitudes are located "opposite" to tropical latitudes and "max" is the "opposite" operation to "min". Exotic semiring constraints are just like Boolean formula except that the atoms are either propositional variables or inequalities between expressions of an exotic semiring. For example, as we will formalize in the sequel, the following is an arctic semiring constraint

$$(a_{11} \geq 0) \wedge (b_{11} \geq 0) \wedge \bigwedge_{\substack{i \in \{1,2\} \\ j \in \{1,2\}}} \begin{cases} (c_{ij} = (a_{i1} \otimes b_{1j}) \oplus (a_{i2} \otimes b_{2j})) \wedge \\ (((a_{i1} \otimes a_{1j}) \oplus (a_{i2} \otimes a_{2j}) > (c_{i1} \otimes a_{1j}) \oplus (c_{i2} \otimes a_{2j})) \vee \\ (((a_{i1} \otimes a_{1j}) \oplus (a_{i2} \otimes a_{2j}) = -\infty) \wedge ((c_{i1} \otimes a_{1j}) \oplus (c_{i2} \otimes a_{2j}) = -\infty))) \wedge \\ (b_{ij} \geq b_{i1} \otimes b_{1j} \oplus b_{i2} \otimes b_{2j}) \end{cases} \tag{1}$$

where $\oplus$ and $\otimes$ correspond to max and sum. All variables are existentially quantified and the constraint is solved for example if $a_{11} = a_{12} = b_{11} = b_{21} = c_{11} = c_{11} = 0$, $a_{21} = a_{22} = c_{12} = 1$, and $b_{12} = b_{22} = c_{21} = c_{22} = -\infty$.

Exotic semirings and constraints come up in a variety of applications such as formal languages, as described in [Sim88], and optimization problems, idempotent analysis, and disjunctive invariants in static analysis as described in [GKS11]. Exotic semirings constraints have

*Department of Computer Science, Ben-Gurion University of the Negev, Israel
†Department of Computer Science, University College London, United Kingdom
‡LuFG Informatik 2, RWTH Aachen University, Germany
§Fakultät IMN, HTWK Leipzig, Germany

recently proven to be very useful in the context of *termination analysis* for string [Wal07] and term rewriting [KW09, ST10]. To simplify the presentation, we focus in this paper on the application to string rewriting, however the techniques carry over to term rewriting as well.

In string rewriting, the state of a computation is represented by a word s over an alphabet Σ. The program underlying the state transitions is a *string rewrite system (SRS)*, i.e., a set $\mathcal{R}$ of *rewrite rules* $\ell \to r$ where ℓ and r are words over Σ. A transition then corresponds to a rewrite step $u\ell v \to_{\mathcal{R}} urv$ for some rule $\ell \to r$ and words u, v (i.e., we replace the substring ℓ by r).

We say that $\mathcal{R}$ is *terminating* if no infinite sequence of rewrite steps $s_0 \to_{\mathcal{R}} s_1 \to_{\mathcal{R}} s_2 \to_{\mathcal{R}} \dots$ exists. To prove termination, a classic method is to use *interpretations*, i.e., to map letters a to elements a_D of a well-founded carrier $(D, >)$ and then to extend this mapping to words $s = a_1 \dots a_n$ by multiplication $s_D = a_{1\,D} \otimes \dots \otimes a_{n\,D}$. Then $\mathcal{R}$ is terminating if all rules are oriented (strictly), namely $\ell_D > r_D$ holds for all rules $\ell \to r \in \mathcal{R}$.

For example, termination of the system $\mathcal{R} = \{\ \mathsf{a}\,\mathsf{b} \to \mathsf{b}\,\mathsf{c}\ \}$, can be shown with an interpretation to the well-founded carrier $(\mathbb{N}, >)$ where $\mathsf{a}_{\mathbb{N}} = 2$ and $\mathsf{b}_{\mathbb{N}} = \mathsf{c}_{\mathbb{N}} = 1$ since $2 \cdot 1 > 1 \cdot 1$ (using the standard multiplication on $\mathbb{N}$). Such a termination proof could be found automatically by again fixing the carrier $(\mathbb{N}, >)$, but leaving the interpretation for the letters open, i.e., using an interpretation *template*. This yields a constraint problem over the well-founded carrier. So we seek an interpretation which orients all of the rules (strictly). For our example we get the constraint $\mathsf{a}_{\mathbb{N}} \cdot \mathsf{b}_{\mathbb{N}} > \mathsf{b}_{\mathbb{N}} \cdot \mathsf{c}_{\mathbb{N}}$ over the (positive) natural numbers, for which the above interpretation is indeed a solution.

Recently, termination techniques using interpretations to *exotic semirings* [Wal07, KW09, ST10], and their extensions to *exotic (square) matrices* (cf. also [EWZ08]) have been shown to be very powerful. The extension of multiplication to exotic matrices is analogous to standard matrix multiplication, where the operations on matrix entries have the semantics induced by the underlying semiring for the entries. To compare two matrices A and B, we extend $>$ component-wise. With these techniques, we are typically given two sets of rewrite rules, $\mathcal{R}$ and $\mathcal{S}$, and required to find an interpretation which orients all rules in $\mathcal{R} \cup \mathcal{S}$ weakly, and at least one rule in $\mathcal{R}$ strictly. We will formalize this in the sequel.

To solve the resulting template constraints over exotic matrices, [Wal07, KW09] suggest to model the operations at the binary level and to apply binary bit-blasting. Binary bit-blasting of arithmetic is convenient because of its compactness. However, size is not the only metric to regard for SAT encodings. Instead, in other applications [TTKB09, MCLS11] unary bit-blasting has been shown to result in encodings which often lead to smaller *runtimes* of the SAT solver. In this paper, we propose a unary encoding based on the, so-called, order encoding (see e.g. [CB94, BB03]) which has proven useful in a variety of applications involving arithmetic with small integer values (see e.g. [TTKB09]). We also investigate the application of SMT solvers. We observe that good results are obtained using the unary encoding, typically outperforming binary bit-blasting as well as SMT solvers.

2 Preliminaries: Exotic Semirings

A *semiring* consists of a domain D equipped with operations $\oplus, \otimes$ (addition, multiplication) and elements $0_D, 1_D \in D$ such that $(D, \oplus, 0_D)$ is a commutative monoid (i.e., $\oplus$ is associative and 0_D is its neutral element), $(D, \otimes, 1_D)$ is a monoid, $\oplus$ distributes over $\otimes$ from both sides, and 0_D is annihilating on both sides of $\otimes$. The standard example of a semiring is $(\mathbb{N}, +, \cdot, 0, 1)$. We will focus on *idempotent* semirings, where addition is idempotent: $x \oplus x = x$.

Definition 1. *The following idempotent semirings are collectively called exotic:*

- $\mathbb{A} = (\{-\infty\} \cup \mathbb{N}, \max, +, -\infty, 0)$*, the arctic semiring,*
- $\mathbb{T} = (\mathbb{N} \cup \{+\infty\}, \min, +, +\infty, 0)$*, the tropical semiring,*
- $\mathbb{F} = (\{-\infty\} \cup \mathbb{N} \cup \{+\infty\}, \min, \max, +\infty, -\infty)$*, the fuzzy semiring,*
- $\mathbb{P} = (\{-\infty\} \cup \mathbb{Z}, \max, +, -\infty, 0)$*, the $\mathbb{Z}$-arctic semiring (also known as arctic below zero) cf. [Kro98] (where arctic is called polar).*

We assume, for each of the exotic semirings, D, an order extending the standard order on $\mathbb{N}$ (or $\mathbb{Z}$), namely, $-\infty < 0 < 1 < \ldots < +\infty$.

We denote the reflexive closure of $<$ by $\leq$, and we write $x <_0 y$ for $(x < y) \vee (x = 0_D = y)$, where 0_D denotes the zero element of D. For the fuzzy semiring, we also need the following notation: $x <_1 y$ iff $(x <_0 y) \vee (x = 1_D = y)$.

For any (exotic) semiring D, and $n \in \mathbb{N}$, the set $M_n(D)$ of $n \times n$-matrices with entries from D is again a semiring, with addition and multiplication defined in the standard way, using $\oplus$ and $\otimes$ on the elements. If D is idempotent, then $M_n(D)$ is idempotent. The orders $\leq, <_0, <_1$ are extended from D to $M_n(D)$ component-wise.

We call an element $x \in D$ *positive* if $x \geq 1_D$. A matrix $A \in M_n(D)$ is called *positive*, denoted *positive*(A), if $A_{1,1}$ (the top left element) is positive.

An exotic semiring constraint is a Boolean formula in which all atoms are either propositional variables or exotic semiring inequalities. We say that a constraint is satisfiable if there exists an assignment of exotic values to exotic variables such that replacing inequalities by their implied Boolean values results in a satisfiable Boolean formula.

In the context of automated termination analysis we typically need to solve *exotic termination constraints* where given two sets of rewrite rules, $\mathcal{R}$ and $\mathcal{S}$, we need to find an interpretation to orient all rules $\ell \to r$ in $\mathcal{R} \cup \mathcal{S}$ such that $\ell \geq r$ (resp. $>_1$, for the fuzzy semiring) and at least one rule $\ell \to r$ in $\mathcal{R}$ such that $\ell >_0 r$. In this setting the matrices are required to be positive. For example, assuming the arctic semiring, if $\mathcal{R} = \{\ \mathsf{a}^2 \to \mathsf{a}\,\mathsf{b}\,\mathsf{a}\ \}$ and $\mathcal{S} = \{\ \mathsf{b} \to \mathsf{b}^2\ \}$ with $\mathsf{a}_D = A$ and $\mathsf{b}_D = B$ the constraint system for the unknown matrices A, B that make up the interpretation is positive$(A) \wedge$ positive$(B) \wedge (A^2 >_0 ABA) \wedge (B \geq B^2)$. To seek a solution of dimension 2, we obtain the exotic semiring constraint from Eq. 1, where the auxiliary variables c_{ij} indicate the contents of matrix $C = AB$ and the solution is given by

$$A = \begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix}, B = \begin{pmatrix} 0 & -\infty \\ 0 & -\infty \end{pmatrix}.$$

3 From Exotic to LIA/IDL constraints

In this section we show that exotic constraints can be faithfully translated into logic over theories that are standardized within SMT-LIB. In particular, arctic and tropical constraints can be encoded into QF_LIA, and fuzzy numbers constraints can be encoded into QF_IDL. With this new formalization we generalize the binary SAT encoding for arctic constraints proposed in [KW09]. As an application, we consider a selection of termination problems from the *International Termination Competition*[1] where for each instance we select an exotic semiring and a

[1] `http://www.termination-portal.org/wiki/Termination_Competition`

matrix dimension to generate exotic constraints. We then model these using LIA/IDL to obtain a selection of instances[2] that have been submitted to the SMT (2012) competition.

To model exotic domains we represent arctic and tropical numbers as pairs $Boolean \times \mathbb{N}$. For an arctic a represented by (m, i), if m is *true* then $a = -\infty$ and otherwise $a = i$, and for a tropical t represented by (p, i), if p is *true* then $t = +\infty$ and otherwise $t = i$. Similarly, a fuzzy number f is represented as a triplet (m, i, p) where m is *true* iff $f = -\infty$, p is *true* iff $f = +\infty$ and $f = i$ iff $m = p =$ *false*. (We denote truth values by *true*/*false* or 1/0 depending on the context.)

To encode the "maximum" operation for k arctic elements we have $(m, i) = \bigoplus_k (m_k, i_k)$ iff

$$(m = \bigwedge_k m_k) \wedge (\neg m \rightarrow \bigwedge_k (\neg m_k \rightarrow i \geq i_k)) \wedge (\neg m \rightarrow \bigvee_k (\neg m_k \wedge i = i_k)).$$

Namely, the maximum of k arctic numbers is $-\infty$ if all k numbers are, else it is greater than or equal to each finite number, and equal to one finite number. To encode arctic semiring multiplication we have $(m, i) = \bigotimes_k (m_k, i_k)$ iff

$$(m = \bigvee_k m_k) \wedge (\neg m \rightarrow (i = \sum_k i_k)).$$

As the formulas show, the target of the translation is the theory of integers with addition, as defined in QF_LIA. This encoding can also be used for the $\mathbb{Z}$-arctic semiring ($\mathbb{Z}$-arctic numbers are modeled as pairs $Boolean \times \mathbb{Z}$). Symmetric formulae as for the arctic semiring hold for the tropical semiring operations and are not detailed here. The fuzzy semiring has the minimum and maximum operations. This implies that we do not need addition, so the constraints can be formulated as *difference constraints*. In fact, we are working within the "theory of linear order" (a subset of QF_IDL, where full QF_IDL would also allow addition of constants).

4 Exotic Order Encoding

The *order encoding* (see e.g. [CB94, BB03]) is a unary representation for natural numbers. In this representation the bit vector $\bar{x} = \langle x_1, \ldots, x_k \rangle$ constitutes a monotonic decreasing sequence and represents values between 0 and k. For example, the value 3 in 5 bits is represented as $\langle 1, 1, 1, 0, 0 \rangle$. The bit x_i (for $1 \leq i \leq k$) is interpreted as the statement $\bar{x} \geq i$.

An important property of a Boolean representation for finite domain integers is the ability to represent changes in the set of values a variable can take. It is well-known that the order encoding facilitates the propagation of bounds. Consider an order encoding variable $\bar{x} = \langle x_1, \ldots, x_k \rangle$ with values in the interval $[0, k]$. To restrict $\bar{x}$ to take values in the range $[a, b]$ (for $1 \leq a \leq b \leq k$), it is sufficient to assign $x_a = 1$ and $x_{b+1} = 0$ (if $b < k$). The variables $x_{a'}$ for $1 \leq a' < a$ and $b < b' \leq k$ are then determined *true* and *false*, respectively, by *unit propagation*. For example, given $\bar{x} = \langle x_1, \ldots, x_9 \rangle$, assigning $x_3 = 1$ and $x_6 = 0$ propagates to give $\bar{x} = \langle 1, 1, 1, x_4, x_5, 0, 0, 0, 0 \rangle$, signifying that $\bar{x}$ can take values in the interval $[3, 5]$. This property is exploited in Sugar [TTKB09] which also applies the order encoding.

In [MCLS11], the authors observe an additional property of the order encoding: its ability to specify that a variable cannot take a specific value $1 \leq v \leq k$ in its domain by equating two variables: $x_v = x_{v+1}$. This indicates that the order encoding is well-suited not only to propagate lower and upper bounds, but also to represent integer variables with an arbitrary

[2] `http://www.imn.htwk-leipzig.de/~waldmann/draft/2012/smt-benchmarks/real/`

finite domain. For example, for $\bar{x} = \langle x_1, \ldots, x_9 \rangle$, equating $x_2 = x_3$ imposes that $\bar{x} \neq 2$. Likewise $x_5 = x_6$ and $x_7 = x_8$ impose that $\bar{x} \neq 5$ and $\bar{x} \neq 7$. Applying these equalities to $\bar{x}$ gives, $\bar{x} = \langle x_1, x_2, x_2, x_4, x_5, x_5, x_7, x_7, x_9 \rangle$, signifying that $\bar{x}$ can take values from the set $\{0, 1, 3, 4, 6, 8, 9\}$.

In the remainder of this section we present the standard order encoding for natural numbers and describe how we extend it, first to integers and then to arctic integers. Finally we state that the order encoding is extended to tropical and fuzzy domains in a similar way.

Encoding Naturals: Let $\bar{a} = \langle a_1, \ldots, a_k \rangle$ denote a tuple of k bits. To encode that $\bar{a}$ is in the order encoding we introduce clauses: $unary_{\mathbb{N}}(\bar{a}) = \bigwedge \{ \; a_{i+1} \rightarrow a_i \; \big| \; 0 < i < k \; \}$. We say that $\bar{a}$ is a k-bit natural number in the order encoding (or "natural" for short). It can take values between 0 and k. Let $\bar{a}$ and $\bar{b}$ be k-bit and k'-bit naturals. Their sum, $\bar{c} = \langle c_1, \ldots, c_{k+k'} \rangle$ is defined by the clauses:

$$sum_{\mathbb{N}}(\bar{a}, \bar{b}) = \bigwedge \left\{ \begin{array}{l} (a_i \rightarrow c_i) \;\wedge\; (\neg a_i \rightarrow \neg c_{k'+i}) \;\wedge\; (a_i \wedge b_j \rightarrow c_{i+j}) \;\wedge \\ (b_j \rightarrow c_j) \;\wedge\; (\neg b_j \rightarrow \neg c_{k+j}) \;\wedge\; (\neg a_i \wedge \neg b_j \rightarrow \neg c_{i+j-1}) \end{array} \;\middle|\; \begin{array}{l} 0 < i \leq k, \\ 0 < j \leq k' \end{array} \right\}$$

Let $\bar{a} = \langle a_1, \ldots, a_k \rangle$ be a k-bit natural and $k' > k$. Then the following is the corresponding k'-bit natural obtained by padding with $k' - k$ zeros: $extend_{\mathbb{N}}(\bar{a}, k') = \langle a_1, \ldots, a_k, 0, \ldots, 0 \rangle$.

Assume that $\bar{a}$ and $\bar{b}$ are k-bit naturals (if they are not in the same range then apply $extend_{\mathbb{N}}$). We define $\max(\bar{a}, \bar{b}) = \bigwedge \{ \; c_i \leftrightarrow a_i \vee b_j \; \big| \; 0 < i \leq k \; \}$. The bits $\langle c_1, \ldots, c_k \rangle$ are a k-bit natural representing the maximum. We define $(\bar{a} \geq \bar{b}) = \bigwedge \{ \; b_i \rightarrow a_i \; \big| \; 0 < i \leq k \; \}$ and $(\bar{a} > \bar{b}) = a_1 \wedge \neg b_k \wedge \bigwedge \{ \; b_i \rightarrow a_{i+1} \; \big| \; 0 < i < k \; \}$.

Encoding Integers: We propose a representation for integers in the same spirit as that for the naturals. It facilitates an encoding which is almost identical to the order encoding of the naturals. The key design decision is to represent integers in the range $(-k, +k)$ as a $2k$-bit monotonic decreasing sequence. So, for $\bar{x} = \langle x_1, \ldots, x_{2k} \rangle$, the bit x_i (for $1 \leq i \leq 2k$) is interpreted as the statement $\bar{x} \geq -k + i$. For example, $\langle 0, 0 \rangle$, $\langle 1, 0, 0, 0 \rangle$, $\langle 1, 1, 0, 0, 0, 0 \rangle$ all represent -1 in different bit lengths; $\langle 1, 0 \rangle$, $\langle 1, 1, 0, 0 \rangle$, $\langle 1, 1, 1, 0, 0, 0 \rangle$ all represent 0 in different bit lengths; and $\langle 1, 1 \rangle$, $\langle 1, 1, 1, 0 \rangle$, $\langle 1, 1, 1, 1, 0, 0 \rangle$ all represent $+1$ in different bit lengths.

Given this representation, all operations are (almost) the same as for naturals. Let $\bar{a} = \langle a_1, \ldots, a_{2k} \rangle$ denote a tuple of $2k$ bits. To encode that $\bar{a}$ is an integer in the order encoding we introduce clauses: $unary_{\mathbb{Z}}(\bar{a}) = unary_{\mathbb{N}}(\bar{a})$ Let $\bar{a}$ and $\bar{b}$ be $2k$-bit and $2k'$-bit unary order encoding integers. Their sum, $\bar{c} = \langle c_1, \ldots, c_{2(k+k')} \rangle$ is defined exactly the same as for the unary case by adding the clauses $sum_{\mathbb{Z}}(\bar{a}, \bar{b}) = sum_{\mathbb{N}}(\bar{a}, \bar{b})$. Namely, simply by viewing $\bar{a}$ and $\bar{b}$ as if they were natural numbers. For example, $\langle 0, 0 \rangle + \langle 1, 0, 0, 0 \rangle = \langle 1, 0, 0, 0, 0, 0 \rangle$ (for $-1 + -1 = -2$) or $\langle 1, 0, 0, 0 \rangle + \langle 1, 1 \rangle = \langle 1, 1, 1, 0, 0, 0 \rangle$ (for $-1 + 1 = 0$).

Let $\bar{a} = \langle a_1, \ldots, a_{2k} \rangle$ be a $2k$-bit integer and $k' > k$. Then the following is the corresponding $2k'$-bit integer: $extend_{\mathbb{Z}}(\bar{a}, k') = \langle 1, \ldots, 1, a_1, \ldots, a_k, 0, \ldots, 0 \rangle$ where the original bits are padded by $(k' - k)$ ones on the left and $(k' - k)$ zeros on the right. Now we will assume that $\bar{a}$ and $\bar{b}$ are $2k$-bit unary order encoding integers (if they are not in the same range then apply $extend_{\mathbb{Z}}$). In this setting, max, $>$, $\geq$ are implemented exactly the same as their natural number counterparts.

Encoding Arctic Integers: We propose the following representation for arctic integers. Obviously, we require an extra bit to capture the case where $\bar{a} = -\infty$. To facilitate an encoding which is similar to the order encoding for naturals and integers, we position this extra bit as the leftmost in a $(2k + 1)$-bit monotonic decreasing sequence. So, for $\bar{a} = \langle a_0, a_1, \ldots, a_{2k} \rangle$, the

bit a_0 is *false* when $\bar{a} = -\infty$ (and then also all other bits are zero). And, just like for integers, the bit a_i (for $1 \leq i \leq 2k$) is interpreted as the statement $\bar{a} \geq -k + i$.

To encode that $\bar{a}$ is an arctic integer in the order encoding we introduce clauses: $unary_{\mathbb{P}}(\bar{a}) = unary_{\mathbb{N}}(\bar{a})$ (reusing the definition for naturals). We say that $\bar{a}$ is a $(2k+1)$-bit unary order encoding integer. It can take values between $-k$ and $+k$ or $-\infty$.

Let $\bar{a} = \langle a_0, a_1, a_{2k}\rangle$ and $\bar{b} = \langle b_0, b_1, b_{2k'}\rangle$ denote $2k{+}1$ and $2k'{+}1$ order encoding arctics. Let $c_0 = a_0 \wedge b_0$ and let $\langle c_1, \ldots, c_{2(k+k')}\rangle$ denote the usual integer (or natural) sum of $\langle a_1, \ldots, a_{2k}\rangle$ and $\langle b_1, \ldots, b_{2k'}\rangle$ (each without its first bit). Then, the arctic sum $\bar{a} + \bar{b}$ is $\langle c_0, c_0 \wedge c_1, \ldots, c_0 \wedge c_{2(k+k')}\rangle$.

Let $\bar{a} = \langle a_0, a_1, \ldots, a_{2k}\rangle$ be a unary arctic number in $2k{+}1$ bits. Its extension to $2k'{+}1$ bits for $k' > k$ is obtained as $extend_{\mathbb{P}}(\bar{a}, k') = \langle a_0, \ldots, a_0, a_1, \ldots, a_{2k}, 0, \ldots, 0\rangle$ (padding by $(k'-k)$ times the a_0 bit on the left and $(k'-k)$ times a 0 bit on the right). Assume that $\bar{a}$ and $\bar{b}$ are $(2k{+}1)$-bit arctic integers (if they are not in the same range then apply $extend_{\mathbb{P}}$). Now, max, $>$, $\geq$ are implemented directly using their natural number counterparts. So everything about the encoding is "for free". Encoding $>_0$ is like this: $(\bar{a} >_0 \bar{b}) = \neg b_k \wedge \bigwedge \left\{ \; b_i \rightarrow a_{i+1} \;\middle|\; 0 < i < k \; \right\}$.

Encoding Tropical and Fuzzy Integers: Tropical numbers are handled in much the same way as arctics: with an "infinity bit", and an unary encoding of the finite value, but we put the infinity bit in the rightmost position. That way, we can use the standard "minimum" operation for unary integers, and get the correct result also for the case that some arguments are infinite. For fuzzy numbers, we represent "minus infinity" by 0, and "plus infinity" by B (the bit width). The semiring operations *are* standard minimum and maximum operations with the usual unary encodings.

5 A Knockout Example

The termination status of problem `SRS/Gebhardt/19` has been open since it was posed in 2006. Using the methods from this paper, we obtain a tropical matrix interpretation to prove its termination. A key step in the proof involves the setting where (the alphabet is $\Sigma = \left\{ \; 0, 1, 0_\#, 1_\# \; \right\}$):

$$\mathcal{R} = \left\{ \begin{array}{l} 0_\#\ 0\ 0\ 0 \ \rightarrow\ 1_\#\ 0\ 1\ 1, \\ 1_\#\ 0\ 0\ 1 \ \rightarrow\ 0_\#\ 0\ 1\ 0 \end{array} \right\} \qquad \mathcal{S} = \left\{ \begin{array}{l} 0\ 0\ 0\ 0 \ \rightarrow\ 1\ 0\ 1\ 1, \\ 1\ 0\ 0\ 1 \ \rightarrow\ 0\ 0\ 1\ 0 \end{array} \right\}$$

So we seek a tropical matrix interpretation which satisfies the constraints $0_\#0^3 \geq 1_\#01^2$, $1_\#0^21 \geq 0_\#010$, $0^4 \geq 101^2$, $10^21 \geq 0^210$ and also $(0^4 >_0 101^2) \vee (10^21 >_0 0^210)$.

In approximately one hour of SAT solving time, using the order encoding with five bits per tropical unknown we obtain the following tropical interpretation (where + indicates $+\infty$) which allows to remove the second rule from $\mathcal{R}$ to render a proof of termination.

$$0 \mapsto \begin{pmatrix} 4 & 0 & 3 & + & 4 & 4 & 4 & 4 & + \\ + & 4 & + & + & 0 & + & + & 4 & + \\ 3 & + & + & 2 & 3 & 4 & 3 & + & + \\ 3 & 0 & + & + & + & 0 & + & 0 & + \\ 4 & 2 & 0 & + & 4 & + & + & + & + \\ 0 & 2 & + & 3 & 4 & + & 3 & + & 3 \\ 2 & 0 & 0 & 4 & + & 0 & + & 2 & 3 \\ 2 & 4 & 3 & 0 & + & + & 0 & + & + \\ + & + & + & + & + & + & + & + & 0 \end{pmatrix} \qquad 1 \mapsto \begin{pmatrix} 2 & + & + & 4 & 3 & 4 & + & + & + \\ + & 3 & 0 & 2 & 2 & 0 & 4 & + & + \\ + & 2 & 0 & 4 & 0 & + & 0 & 3 & + \\ 0 & 4 & + & + & 3 & 3 & 4 & 4 & + \\ 2 & + & + & + & 4 & 4 & + & + & + \\ 0 & 4 & + & 0 & 0 & 2 & 1 & + & 3 \\ 0 & + & + & + & + & + & 3 & + & + \\ + & 0 & 0 & 0 & + & + & 2 & 0 & 1 \\ + & + & + & + & + & + & + & + & 0 \end{pmatrix}$$

$$0_{\#} \mapsto \begin{pmatrix} 0 & 1 & 1 & 2 & 0 & 0 & 3 & 0 & 2 \\ + & + & + & + & + & + & + & + & + \\ 0 & + & 0 & 1 & + & + & + & + & 2 \\ 1 & + & 1 & 2 & + & + & + & + & 3 \\ 0 & + & 1 & 1 & + & + & + & 1 & 2 \\ 0 & 1 & + & 3 & + & 3 & 2 & + & 4 \\ 0 & + & 0 & 3 & + & + & + & 2 & 2 \\ 2 & 4 & + & 4 & 2 & + & 4 & + & 4 \\ + & + & + & + & + & + & + & + & 0 \end{pmatrix} \quad 1_{\#} \mapsto \begin{pmatrix} 0 & 4 & 1 & 1 & 0 & 1 & 0 & + & + \\ + & + & + & + & + & + & + & + & + \\ 0 & 2 & 0 & 1 & 1 & + & + & + & + \\ 2 & + & 2 & 2 & 1 & + & 2 & 4 & 4 \\ 0 & + & 4 & 1 & 2 & 2 & 2 & + & 3 \\ + & + & 2 & + & 0 & 3 & 1 & + & + \\ + & + & + & 1 & 0 & + & + & + & 3 \\ 4 & + & 4 & 3 & 4 & 4 & 2 & + & + \\ + & + & + & + & + & + & + & + & 0 \end{pmatrix}$$

6 Tools and Experiments

To assess our contributions empirically, we conducted several preliminary experiments. We use three tools

(1) satchmo-smt—Johannes Waldmann enhanced his satchmo-smt [Wal] solver, which uses a binary encoding of QF_LIA (cf. e.g. [EWZ08, FGM+07]), to perform also a unary encoding similar to Sect. 4. The solver satchmo-smt makes use of the solver MiniSAT [ES03] (development version based on version 2.2). So for arctic constraints, the binary encoding essentially corresponds to that proposed in [KW09], whereas the unary encoding is new.

(2) BEE—Yoav Fekete enhanced the BEE constraint solver developed at Ben-Gurion University to compile arctic constraints to CNF. It combines constraint and CNF simplification techniques to provide a concise CNF representation [MCLS11]. BEE uses the SAT solver CryptoMiniSAT [SNC09], version 2.5.1, as a back-end solver;

(3) Z3, version 3.2—This SMT solver [dB08] is developed at MS Research and can handle QF_LIA end QF_IDL theories (among others).

Our experimentation initiates a comparison of these tools when solving exotic semiring constraints. We view this experimentation as a proof of concept and starting point for a more thorough investigation on how to solve exotic semiring constraints using SMT solvers and SAT encodings. As the benchmark set, we generated exotic termination constraints with matrix dimension up to 5 from the string rewriting problems of the *Termination Problem Data Base (TPDB)*[3], selected those that were solvable by satchmo-smt within 10 minutes, and removed those that were solvable quickly (< 2 seconds).

Exotic constraints were given directly to BEE, and for satchmo-smt and Z3 they were being translated to QF_LIA as described in Sect. 3.

The benchmark runs were performed on a Quad-core Intel i5-2400 at 3.1 GHz with 4 GB RAM. We conducted the following experiments which are summarized in Table 1. All times are in seconds. The timeout is 1800 sec. The rows of the table were selected from 194 instances. The average, max and number of timeout information in the last 3 rows is a statistic on *all* 194 instances.

Experiment 1: We apply satchmo-smt to compare binary and unary encodings for LIA constraints that stem from arctic constraints. So while here we do not directly take optimizations by dedicated domain knowledge of the arctic semiring into account, this experiment still allows us to make observations on performance of binary vs. unary bit-blasting. Here (arctic) variables

[3] http://termination-portal.org/wiki/TPDB

instance	Experiment 1		Experiment 2		Experiment 3	
	satchmo$_b$ (3)	satchmo$_u$ (7)	BEE (it)	Z3	BEE (4)	satchmo$_u$ (4)
S2.1	55.09	1.20	0.73	112.39	3.74	0.70
S2.2	0.81	4.91	0.66	411.13	7.31	11.72
S2.3	11.82	7.11	12.94	111.38	2.02	154.74
S2.4	2.81	2.81	11.6	891.97	2.23	219.14
S3.1	1.51	84.34	1.62	1754.24	9.58	1.21
S3.2	4.31	23.34	4.15	428.24	6.55	2.61
S3.3	4.81	24.14	2.10	199.43	3.94	3.31
S3.4	71.41	8.42	0.97	472.16	1.66	0.70
S3.5	108.97	16.63	0.60	359.24	1.24	0.60
S3.6	TimeOut	TimeOut	9.13	305.49	2.13	123.59
S3.7	2.31	8.82	9.43	TimeOut	1.82	128.1
S4.0	2.41	4.71	2.15	6.69	21.31	1.50
S4.1	4.91	26.04	6.00	610.73	8.83	3.51
S4.2	6.11	4.31	3.68	27.06	26.94	2.21
S4.3	TimeOut	172.18	2.55	TimeOut	35.68	7.71
S5.1	51.69	19.33	17.92	1550.72	15.23	4.11
S5.2	TimeOut	46.18	2.33	TimeOut	85.44	77.92
S5.3	TimeOut	TimeOut	3.40	TimeOut	20.59	28.85
S5.4	TimeOut	725.44	4.16	TimeOut	160.79	44.07
S5.5	18.93	17.33	3.45	TimeOut	6.5	13.12
S6.1	4.61	9.72	9.05	45.33	27.22	4.21
S6.2	7.32	4.41	4.30	41.10	29.64	2.61
S6.3	9.82	8.02	8.57	84.25	21.10	4.31
S6.4	169.98	47.58	1.62	138.57	31.04	4.41
S6.5	TimeOut	1408.21	6.80	TimeOut	731.93	TimeOut
average	61.58	5.44	2.22	224.64	10.83	14.94
max	1800	1800	17.92	1800	731.93	1800
timeouts	6	2	0	15	0	1

Table 1: results from 3 experiments

take values $\{-\infty, 0, \ldots, 7\}$. This means 3 bits for the binary representation and 7 bits for the unary as indicated in the headers of these two rows. Despite the significantly more verbose representation in the unary encoding, the corresponding average runtime is lower by an order of magnitude in comparison to the average runtime for the binary encoding. Also the worst-case behavior of unary bit-blasting is better (only 2 timeouts instead of 6 for binary bit-blasting).

Experiment 2: Here we compare unary bit-blasting as in Sect. 4 using BEE [MCLS11] with SMT solving using Z3.

SMT solvers with a dedicated theory solver like Z3 natively support *all* natural numbers as search space for the input constraint. For bit-blasting with BEE, we use an *iterative deepening* approach which doubles n for the search space $\{-\infty, 0, \ldots, n\}$ until a solution is found or n reaches a specified upper bound. This is indicated by "(it)" on the corresponding column. Here we can show satisfiability of all instances using at most 4 bits.

With BEE, we observe an improvement in the average runtime by *two* orders of magnitude over Z3. Also in the worst case BEE performs significantly better (max. runtime of less than 18

seconds, whereas Z3 shows 15 timeouts after 1800 seconds each).

Experiment 3: Here we compare unary bit-blasting using BEE and using satchmo-smt, each with 4 bits for the numbers (4). Here BEE and satchmo show similar performance (where BEE is slightly faster on average). While satchmo-smt has an approach of directly bit-blasting LIA constraints "as is" (and not taking the underlying arctic domain into account), BEE has a certain overhead due to the simplifications applied in the SAT compilation phase, which may not always pay out. Moreover, the two solvers use different SAT solvers (CryptoMiniSAT and MiniSAT) for the resulting CNFs, which may also lead to differences in runtime. Here investigations with different SAT solvers as back-ends would be interesting.

7 Discussion

In this paper, we formalize exotic semiring constraints and discuss several new ways of solving them. In addition to the existing approach based on binary bit-blasting, we also provide new approaches based on SMT-LIA and unary bit-blasting with order encoding.

We have applied these techniques to solve exotic semiring constraints that come up in the context of automated termination analysis.

The LIA and IDL constraints derived from exotic termination constraints all share the property that the only numerical constant that appears anywhere, is the number zero. Higher numbers (than zero) may of course appear in the solution, but only as consequences of strict equalities.

It appears that long chains of strict inequalities are "quite unlikely", so if an exotic termination constraint has a solution at all, then it typically also has a solution that uses only small numbers, and this explains the effectiveness of unary bit-blasting solver methods.

For more information on our experiments, we refer to our evaluation web page at the following URL: `http://www.cs.bgu.ac.il/~mcodish/Benchmarks/SMT2012/`

Ongoing research aims to provide a powerful solver for exotic semiring constraints that arise in the context of termination analysis. Here we plan to investigate the application of SMT and SAT solvers with unary encodings, where also applications other than termination analysis may benefit.

References

[BB03] Olivier Bailleux and Yacine Boufkhad. Efficient CNF encoding of boolean cardinality constraints. In Francesca Rossi, editor, *CP*, volume 2833 of *Lecture Notes in Computer Science*, pages 108–122. Springer, 2003.

[CB94] James M. Crawford and Andrew B. Baker. Experimental results on the application of satisfiability algorithms to scheduling problems. In Barbara Hayes-Roth and Richard E. Korf, editors, *AAAI*, pages 1092–1097. AAAI Press / The MIT Press, 1994.

[dB08] Leonardo de Moura and Nikolaj Bjørner. Z3: An efficient SMT solver. In C. R. Ramakrishnan and Jakob Rehof, editors, *TACAS*, volume 4963 of *Lecture Notes in Computer Science*, pages 337–340. Springer, 2008.

[DK09] Manfred Droste and Werner Kuich. Semirings and formal power series. In Manfred Droste, Werner Kuich, and Heiko Vogler, editors, *Handbook of Weighted Automata*, pages 3–28. Springer, 2009.

[ES03] Niklas Eén and Niklas Sörensson. An extensible SAT-solver. In Enrico Giunchiglia and Armando Tacchella, editors, *SAT*, volume 2919 of *Lecture Notes in Computer Science*, pages 502–518. Springer, 2003.

[EWZ08] Jörg Endrullis, Johannes Waldmann, and Hans Zantema. Matrix interpretations for proving termination of term rewriting. *J. Autom. Reasoning*, 40(2-3):195–220, 2008.

[FGM+07] Carsten Fuhs, Jürgen Giesl, Aart Middeldorp, Peter Schneider-Kamp, René Thiemann, and Harald Zankl. SAT solving for termination analysis with polynomial interpretations. In João Marques-Silva and Karem A. Sakallah, editors, *SAT*, volume 4501 of *Lecture Notes in Computer Science*, pages 340–354. Springer, 2007.

[GKS11] Stéphane Gaubert, Ricardo Katz, and Sergei Sergeev. Tropical linear-fractional programming and parametric mean payoff games. *ArXiv e-prints*, 2011. Available from `http://arxiv.org/abs/1101.3431`, extended version of a workshop paper at the Workshop on Invariant Generation (WING 2010) `http://research.microsoft.com/en-us/events/wing2010/` at FLoC '10.

[Goo98] Joshua T. Goodman. *Parsing inside-out*. PhD thesis, Harvard University, 1998.

[Kro98] Daniel Krob. Some automata-theoretic aspects of min-max-plus semirings. In Jeremy Gunawardena, editor, *Idempotency*, pages 70–79. Cambridge University Press, 1998.

[KW09] Adam Koprowski and Johannes Waldmann. Max/plus tree automata for termination of term rewriting. *Acta Cybern.*, 19(2):357–392, 2009.

[MCLS11] Amit Metodi, Michael Codish, Vitaly Lagoon, and Peter J. Stuckey. Boolean equi-propagation for optimized SAT encoding. In Jimmy Ho-Man Lee, editor, *CP*, volume 6876 of *Lecture Notes in Computer Science*, pages 621–636. Springer, 2011.

[Sim88] Imre Simon. Recognizable sets with multiplicities in the tropical semiring. In Michal Chytil, Ladislav Janiga, and Václav Koubek, editors, *MFCS*, volume 324 of *Lecture Notes in Computer Science*, pages 107–120. Springer, 1988.

[SNC09] Mate Soos, Karsten Nohl, and Claude Castelluccia. Extending SAT solvers to cryptographic problems. In Oliver Kullmann, editor, *SAT*, volume 5584 of *Lecture Notes in Computer Science*, pages 244–257. Springer, 2009.

[ST10] Christian Sternagel and René Thiemann. Certification extends termination techniques. In *Proc. 11th International Workshop on Termination (WST '10)*, 2010.

[TTKB09] Naoyuki Tamura, Akiko Taga, Satoshi Kitagawa, and Mutsunori Banbara. Compiling finite linear CSP into SAT. *Constraints*, 14(2):254–272, 2009.

[Wal] Johannes Waldmann. satchmo-smt. `https://github.com/jwaldmann/satchmo-smt`.

[Wal07] Johannes Waldmann. Weighted automata for proving termination of string rewriting. *Journal of Automata, Languages and Combinatorics*, 12(4):545–570, 2007.