

Decision Procedures for Queues with Integer Constraints

Ting Zhang, Henny B. Sipma, Zohar Manna*

Computer Science Department
Stanford University
{tingz, sipma, zm}@theory.stanford.edu

Abstract. Queues are a widely used data structure in programming languages. They also provide an important synchronization mechanism in modeling distributed protocols. In this paper we extend the theory of queues with a length function that maps a queue to its size, resulting in a combined theory of queues and Presburger arithmetic. This extension provides a natural but tight coupling between the two theories, and hence the general Nelson-Oppen combination method for decision procedures is not applicable. We present a decision procedure for the quantifier-free theory and a quantifier elimination procedure for the first-order theory that can remove a block of existential quantifiers in one step.

1 Introduction.

Queues are a widely used data structure in programming languages. They also provide an important synchronization mechanism in modeling distributed protocols. To verify programs or protocols using queues we must be able to reason about this data structure. Single theory decision procedures, however, are usually not applicable, because programming languages often involve multiple data domains. A natural example of such “mixed” constraints are combinations of queues with integer constraints on the size of queues.

In this paper we extend the theory of queues with a length function that maps a queue to its size, resulting in a combined theory of queues and Presburger arithmetic (PA). The language is the set-theoretic union of the language of queues and that of PA. Formulae are formed from *atom*, *queue*, and *integer literals* using logical connectives and quantifiers. The two theories are connected by the length function $|\cdot| : Q \rightarrow \mathbb{N}$. With the expressive power of PA, we can express linear relations between sizes of queues. E.g., in a network with n input queues q_i , the property that the influx is bounded by B can be expressed as $\sum_{i=0}^{n-1} |q_i| < B$.

We present a decision procedure for the quantifier-free theory of queues. The method extracts accurate integer constraints from queue constraints. Thus, we can utilize decision procedures for queues and integers to derive the new decision procedure. We also present a quantifier elimination procedure for the

* This research was supported in part by NSF grants CCR-01-21403, CCR-02-20134, CCR-02-09237, CNS-0411363, and CCF-0430102, by ARO grant DAAD19-01-1-0723, and by NAVY/ONR contract N00014-03-1-0939.

first-order theory of queues with integers. The elimination procedure removes a block of existential quantifiers in one step. In all developments, we assume that the atom domain is finite; the decision problems in an infinite domain are considerably easier.

Related Work and Comparison. Presburger arithmetic (PA) was first shown to be decidable in 1929 by the quantifier elimination method [4]. Efficient algorithms were later discovered by [3] and further improved in [9].

[2] gave a decision procedure for the quantifier-free theory of queues with subsequence relations which consist of prefix, suffix and sub-queue relations. It also discussed the integer combination for the case of infinite atom domain without the subsequence relations. The decidability and the complexity of the first-order theory of queues were given by [10, 11]. By the decidability of $WS1S$ and a standard encoding, the theory of words with the prefix relation and the successor operator (i.e., a theory of queues) is decidable and admits quantifier elimination [1]. [1, 12] studied theories of words with “equal length” predicate which can be viewed as special integer constraints.

This arithmetic extension provides a natural but tight coupling between the two theories, and hence the general Nelson-Oppen combination method [8] for decision procedures is not applicable. Recently [6] showed the decidability of a fragment of $WS1S$ with cardinality constraints ($WS1S^{card}$) and the undecidability of $WS1S^{card}$ for the fragments with alternation of second-order quantifiers. By a standard encoding (in which a queue is represented as sets of natural numbers), the theory of queues with integers can be interpreted in $WS1S^{card}$. Even the quantifier-free theory of queues with integers, however, is unlikely to be interpreted in the quantifier-free fragment of $WS1S^{card}$, because encoding a queue by sets of natural numbers necessarily involves quantification. Moreover, though interpretation in general renders elegant decidability results, it produces less efficient decision procedures in practice, especially if the host theory has high complexity (in our case even the existential $WS1S$ is non-elementary).

In [13, 14] we gave decision procedures for the theory of term algebras with integer constraints. The method relies on a key normalization process to extract integer constraints from term constraints. The normalization partitions terms into stratified clusters such that (i) each cluster consists of pairwise unequal terms (trees) of the same length, and (ii) disequalities between composite terms (proper trees) in a cluster are implied by disequalities in the clusters of lower ranks. Property (ii) allows the construction of a satisfying assignment in a bottom-up fashion, while providing integer constraints that express the satisfiability of the clusters. Thus, (i) and (ii) allow us to reduce the satisfiability of the original formula to the satisfiability of computable integer constraints. The decision procedure presented here relies on the same idea, but for queues disequalities cannot be normalized into stratified clusters, because queues are not uniquely generated. Consider, for example, the constraint

$$X \neq Y \wedge aX \neq Yb \wedge Xa \neq bY \wedge |X| = |Y|.$$

Infinitely many assignments of the form $\{X = (ba)^n b, Y = a(ba)^n\}$ satisfy $X \neq Y$, but neither $aX \neq Yb$ nor $Xa \neq bY$. Therefore, we cannot construct a satisfying

assignment inductively. In this paper we present new normalization procedures that allow the computation of a *cut length* L_t for all queue variables: below L_t all satisfying assignments can be enumerated, above L_t integer constraints can be computed that are equisatisfiable with the original formula.

Paper Organization. Sec. 2 defines the language and structure of queues and presents some word properties. Sec. 3 describes a decision procedure for the quantifier-free theory of queues [2], the basis for our decision procedures. Sec. 4 introduces the theory of queues augmented with Presburger arithmetic and presents the technical machinery for the decision procedures. Sec. 5 presents the main contribution of this paper: it adapts the technique in [13, 14] to derive a decision procedure for the extended theory of queues. Sec. 6 applies the technique to give a quantifier elimination procedure for the extended first-order theory of queues. Sec. 7 concludes with a discussion on complexity and some ideas for future work. Due to space limitation all proofs and some algorithms have been omitted. An extended version of this paper is available from the first author’s webpage.

2 The Theory of Queues

We present a two-sorted language and structure of queues. For notation convenience, we do not distinguish syntactic terms in the language from semantic terms in the structure. The meaning should be clear from the context.

Definition 1. *The structure of queues* $\mathfrak{Q} : \langle \mathcal{Q}; \mathcal{A}, \mathcal{C}, \mathcal{S} \rangle$ consists of

1. \mathcal{A} : A finite set of atoms: a, b, c, \dots . We use $\epsilon_{\mathcal{A}}$ to denote the “phantom atom” whose only purpose is to keep functions on queues total.
2. \mathcal{Q} : The domain of queues, consisting of sequences of atoms. We use $\epsilon_{\mathcal{Q}}$ to denote the empty queue.
3. \mathcal{C} : Two constructors: the left insertion $la : \mathcal{A} \times \mathcal{Q} \rightarrow \mathcal{Q}$ and the right insertion $ra : \mathcal{A} \times \mathcal{Q} \rightarrow \mathcal{Q}$ such that for $\alpha \in \mathcal{Q}$, $la(\epsilon_{\mathcal{A}}, \alpha) = ra(\epsilon_{\mathcal{A}}, \alpha) = \alpha$, and for $a \in \mathcal{A} \setminus \{\epsilon_{\mathcal{A}}\}$, $\langle s_1, \dots, s_n \rangle \in \mathcal{Q}$, $la(a, \epsilon_{\mathcal{Q}}) = ra(a, \epsilon_{\mathcal{Q}}) = \langle a \rangle$, and $la(a, \langle s_1, \dots, s_n \rangle) = \langle a, s_1, \dots, s_n \rangle$, $ra(a, \langle s_1, \dots, s_n \rangle) = \langle s_1, \dots, s_n, a \rangle$.
4. \mathcal{S} : Four selectors: the left head $lh : \mathcal{Q} \rightarrow \mathcal{A}$, the left tail $lt : \mathcal{Q} \rightarrow \mathcal{Q}$, the right head $rh : \mathcal{Q} \rightarrow \mathcal{A}$, and the right tail $rt : \mathcal{Q} \rightarrow \mathcal{Q}$ such that for $\langle s_1, \dots, s_n \rangle \in \mathcal{Q}$, $lh(\epsilon_{\mathcal{Q}}) = rh(\epsilon_{\mathcal{Q}}) = \epsilon_{\mathcal{A}}$, $lt(\epsilon_{\mathcal{Q}}) = rt(\epsilon_{\mathcal{Q}}) = \epsilon_{\mathcal{Q}}$, and

$$\begin{aligned} lh(\langle s_1, \dots, s_n \rangle) &= s_1, & lt(\langle s_1, \dots, s_n \rangle) &= \langle s_2, \dots, s_n \rangle, \\ rh(\langle s_1, \dots, s_n \rangle) &= s_n, & rt(\langle s_1, \dots, s_n \rangle) &= \langle s_1, \dots, s_{n-1} \rangle. \end{aligned}$$

We use $\mathcal{L}_{\mathcal{Q}}$ for the language of queues.

Queues are finite *words* constructed from *letters* in \mathcal{A} , i.e., $\mathcal{Q} = \mathcal{A}^*$. We assume $|\mathcal{A}| > 1$ as queue constraints trivially reduce to integer constraints if \mathcal{A} is a singleton. We use “word”, “letter” in semantic discussions and use “queue”, “atom” to refer to their counterparts in the formal language. For a word α , $|\alpha|$ denotes the length of α ; $\alpha[i]$ ($1 \leq i \leq |\alpha|$) denotes the letter at position i ; $\alpha[m..n]$ ($1 \leq m, n \leq |\alpha|$) denotes the consecutive fragment from position m to position n ;

α^m denotes the word obtained by concatenating m copies of α ; α^* (α^+) denotes the set $\{\alpha^m \mid m \geq 0\}$ ($\{\alpha^m \mid m > 0\}$).

Because of finiteness of \mathcal{A} , we assume only constant atoms appear in formulas (i.e., no occurrences of atom variables). For clarity, X, Y, Z, \dots are reserved for queue variables, a, b, c, \dots for constant atoms and $\alpha, \beta, \gamma, \dots$ for constant queues. We use concatenation \circ to express constructor operations. For example, $a \circ X \circ b$ stands for either $\text{ra}(b, \text{la}(a, X))$ or $\text{la}(a, \text{ra}(b, X))$. Often we even omit \circ unless necessary for clarity.

The expressive power of the constructor language (the language without selectors) is the same as that of \mathcal{L}_Q .

Proposition 1 (Elimination of Selectors). *For any φ in \mathcal{L}_Q , one can effectively compute an equivalent φ' such that (i) φ' contains no selectors, and (ii) if φ is quantifier-free, then φ' can be put into either \exists_1 or \forall_1 form.*

So in terms of satisfiability or validity, even in the quantifier-free fragment of \mathcal{L}_Q , selectors are dispensable without compromising expressiveness. From now on we assume \mathcal{L}_Q is the constructor language except in Sec. 6 where selectors are used in quantifier elimination. In a constructor language, a queue variable can occur at most once in a term, and hence we can assume all terms of sort Q are in the form $\alpha X \beta$, where α, β are constant words and X is a queue variable.

The equations in \mathcal{L}_Q can express certain “circular properties” on queues.

Definition 2 ([7]). *Two words α, β are conjugate if there exist words u, v ($v \neq \epsilon_Q$) such that $\alpha = uv$ and $\beta = vu$. In other words, α is obtained from β by circular shift, and vice versa. We say that α is k -conjugate with β if $|u| = k$.*

Let $\text{ext}(\beta, m, k)$ denote $\beta^m \beta[1..k]$, $\text{orb}(\beta, k)$ the set $\{\text{ext}(\beta, m, k) \mid m \geq 0\}$, and $\text{orb}(\beta)$ the set $\bigcup_{k \geq 0} \text{orb}(\beta, k)$. Note that $\text{orb}(\beta)$ is the orbit of all words of the form $\beta^* \beta[1..i]$ ($i < |\beta|$) and $\text{orb}(\beta, k)$ is the subtrack of $\text{orb}(\beta)$ ending with $\beta[1..k]$.

Example 1. Let $\beta = aba$. Then $\text{ext}(\beta, 1, 2) = abaab$, $\text{orb}(\beta)$ are words in one of the following forms $(aba)^*$, $(aba)^*a$, $(aba)^*ab$, which are $\text{orb}(\beta, 0)$, $\text{orb}(\beta, 1)$ and $\text{orb}(\beta, 2)$.

Proposition 2 ([7]). *Two words α and β are conjugate if and only if there exists γ such that $\alpha\gamma = \gamma\beta$. Moreover, α and β are k -conjugate if and only if for all γ , $\alpha\gamma = \gamma\beta$ if and only if $\gamma \in \text{orb}(\alpha, k)$.*

This proposition says that if $\alpha = u_1 u_2$, $\beta = u_2 u_1$, then the solution set of $\alpha X = X \beta$ is $(u_1 u_2)^* u_1$. As a consequence, we define $X \in \text{orb}(\alpha, k)$ as “syntactic sugar” for $\alpha X = X \alpha[k+1..|\alpha|] \alpha[1..k]$; similarly $X \notin \text{orb}(\alpha, k)$ for $\alpha X \neq X \alpha[k+1..|\alpha|] \alpha[1..k]$.

Definition 3 (Primitive Words). *A word β is primitive if $\beta \neq \alpha^n$ ($n \geq 1$) for any proper prefix α of β , and is strongly primitive if in addition $\beta \notin \text{orb}(\alpha)$.*

Example 2. Consider $\alpha \equiv aba$, $\beta \equiv abab$ and $\gamma \equiv abb$. It is clear that β is non-primitive, α is primitive but not strongly primitive and γ is strongly primitive.

If β is non-primitive, then there exists α such that $\beta \in \alpha^*$. We call the shortest such α the *generator* of β , denoted by $\text{gen}(\beta)$. It is easily seen that $\text{orb}(\beta) = \text{orb}(\text{gen}(\beta))$, i.e., every orbit is uniquely generated. Thus, without loss of generality, we always assume the occurrences of β in $\text{orb}(\beta, k)$ to be primitive.

Proposition 3 ([2, 10]). *Let α, β be two distinct primitive words and γ a word of length n . Then $\gamma \in \text{orb}(\alpha) \cap \text{orb}(\beta)$ implies $n < |\alpha| + |\beta| - 1$.*

This proposition says that $X \in \text{orb}(\alpha)$ and $X \in \text{orb}(\beta)$ (where $\alpha \neq \beta$), are *mutually exclusive* except for a finite number of cases which can be enumerated by comparing two orbits of α and β coordinate-wise up to $|\alpha| + |\beta| - 2$. We have

Proposition 4 ([2, 10]). *A conjunction of literals of the form*

$$\bigwedge_{i=1}^n X \in \text{orb}(\alpha_i) \wedge \bigwedge_{j=1}^m X \notin \text{orb}(\beta_j) \quad (1)$$

can be simplified to a formula in which at most one of $X \in \text{orb}(\alpha_i)$ appears, and if this happens, no $X \notin \text{orb}(\beta_j)$ occurs. In addition, if $n > 1$, (1) simplifies to either false or a finite set of solutions.

Example 3. $X \in \text{orb}(ab) \wedge X \in \text{orb}(aba)$ simplifies to $X \in \{a, b, aba\}$, and $X \in \text{orb}(ab) \wedge X \notin \text{orb}(aba)$ simplifies to $X \in \text{orb}(ab) \wedge X \notin \{a, b, aba\}$.

3 Decision Procedure for $\text{Th}^\forall(\mathfrak{Q})$

The basis of the decision procedures for the combined theory is the decision procedure for the quantifier-free theory of queues, $\text{Th}^\forall(\mathfrak{Q})$ [2]. This decision procedure is *refutation-based*; to determine the validity of a formula φ , it determines the unsatisfiability of $\neg\varphi$, which further reduces to determining the unsatisfiability of each disjunct in the *disjunctive normal form* of $\neg\varphi$. A key constituent of all decision procedures is *equality elimination*.

Definition 4 (Solved Form). *A set of equalities \mathcal{E} is in solved form if every $E \in \mathcal{E}$ has the form $x = t(\bar{x})$ where x neither occurs in \bar{x} nor in any other equations in \mathcal{E} .*

Obviously a set of equalities in solved form pose no restriction on the solution, and hence those equalities can be considered “virtually eliminated”.

Definition 5 (Normal Form in \mathfrak{Q}). *A queue constraint $\Phi_{\mathfrak{Q}}$ is in normal form if (i) all equalities are in solved form, (ii) for each queue variable X there exists at most one literal $X \in \text{orb}(\alpha, k)$, and (iii) disequalities are in the form $\alpha X \neq Y\beta$ for $X \neq Y$.*

The following algorithm, a simplified version of [2], reduces a set of equalities and inequalities to normal form.

Algorithm 1 (Normalization in \mathfrak{Q} , cf. [2]). *Input $\Phi_{\mathfrak{Q}} : \mathcal{E} \cup \mathcal{D}$ where \mathcal{E}, \mathcal{D} are sets of equalities and disequalities, respectively.*

1. *Reduce literals of the form $\alpha X\beta = \alpha'Y\beta'$, $\alpha X\beta \neq \alpha'Y\beta'$, where $\alpha, \beta, \alpha', \beta'$ are constant queues and X, Y are queue variables, to $\alpha X = Y\beta$ and $\alpha X \neq Y\beta$ by position-wise removing prefixes and suffixes. For example, $abXcd = abcYdd$ reduces to false and $abXcd \neq abcYd$ to $Xc \neq cY$.*

2. Eliminate equalities of the form $\alpha X = Y\beta$ with $X \not\equiv Y$. For $|X| < |\beta|$, $\alpha X = Y\beta$ reduces to $X = \beta[|\beta| - |X| + 1..|\beta|] \wedge Y = \alpha\beta[1..|\beta| - |X|]$. For $|X| \geq |\beta|$, $\alpha X = Y\beta$ reduces to $X = X'\beta \wedge Y = \alpha X'$, where X' is a fresh queue variable.
3. Eliminate equalities of the form $\alpha X = X\beta$. By Prop. 2 if α, β are not conjugate, then $\alpha X = X\beta$ simplifies to false. If α, β are k -conjugate, $\alpha X = X\beta$ is replaced by $X \in \text{orb}(\alpha, k)$.
4. Eliminate disequalities of the form $\alpha X \neq X\beta$. Again by Prop. 2, if α, β are not conjugate, $\alpha X \neq X\beta$ simplifies to true. If α, β are k -conjugate, $\alpha X \neq X\beta$ is replaced by $X \notin \text{orb}(\alpha, k)$.

Although the literals $X \in \text{orb}(\alpha, k)$, $X \notin \text{orb}(\alpha, k)$, introduced in steps 3 and 4, are implicit equalities (disequalities, resp.), Prop. 4 ensures that a set of such equalities is either inconsistent or a finite set of solutions can be computed, and that in the presence of $X \in \text{orb}(\alpha, k)$, all occurrences $X \notin \text{orb}(\alpha', k')$ can be eliminated.

We claim that a constraint in normal form is satisfiable: a satisfiable assignment can be constructed incrementally by assigning each queue variable a queue with length distinct from all previously assigned terms. This justifies the following algorithm.

Algorithm 2 ([2]). *Input:* $\Phi \equiv \mathcal{E} \cup \mathcal{D}$.

1. Transform Φ to $\Phi' : \mathcal{E}' \cup \mathcal{D}'$ which is normal.
2. If inconsistency is discovered, return FAIL; otherwise, return SUCCESS.

4 The Theory of Queues with Integers

Definition 6. *The structure of queues with integers is $\mathfrak{Q}_{\mathbb{Z}} : \langle \mathfrak{Q}, \text{PA}; |\cdot| \rangle$ where \mathfrak{Q} is the structure of queues, PA is Presburger arithmetic, and $|\cdot| : \mathfrak{Q} \rightarrow \mathbb{N}$ is the length function such that $|X|$ denotes the number of atoms in the queue X .*

We use subscripts \mathcal{Q} and \mathcal{Z} (or prefixes \mathcal{Q} - and PA -) to denote notions related to queue sort and integer sort, respectively. For example, $\Phi_{\mathcal{Q}}$ denotes a queue formula and $\mathcal{V}_{\mathcal{Q}}$ denotes the collection of queue variables. We use integer terms $|t(X)|$ in two ways; as the function value of $t(X)$ when $t(X)$ is in discussion, and as purely syntactic integer variable (called *pseudo integer variable*). In the latter case, suppose $\Phi_{\mathcal{Z}}(\bar{X})$ is given, then $\Phi_{\mathcal{Z}}(\bar{z})$ is the formula obtained by substituting each pseudo integer variable $|X|$ ($X \in \bar{X}$) for a real integer variable z ($z \in \bar{z}$). $|\bar{X}| = \bar{z}$ denotes $\bigwedge_i |X_i| = z_i$. If σ is an assignment for $\mathcal{V}_{\mathcal{Q}}$, then $|\sigma|$ denotes the corresponding assignment for pseudo integer variables.

In a combined constraint $\Phi_{\mathcal{Q}} \wedge \Phi_{\mathcal{Z}}$, $\Phi_{\mathcal{Z}}$ restricts solutions to $\Phi_{\mathcal{Q}}$.

Example 4. The constraint $\Phi_{\mathcal{Q}} : Xba \neq abY \wedge Xab \neq baY \wedge Xaa \neq baY \wedge Xab \neq aaY$ is not satisfiable with $\Phi_{\mathcal{Z}} : |X| = |Y| = 1$, in $\mathfrak{Q}_{\mathbb{Z}}$ with $\mathcal{A} = \{a, b\}$. It can be easily verified by enumerating all four combinations. On the other hand, both $\Phi_{\mathcal{Q}}$ and $\Phi_{\mathcal{Z}}$ are obviously satisfiable in their respective domains.

A simple but crucial observation is that Φ_Q induces an “implicit” length constraint, in addition to the “explicit” constraint Φ_Z given in the input. For example, in Ex. 4, Φ_Q implies $\Phi_\Delta : |X| = |Y| \rightarrow |X| = |Y| \neq 1$ and thus Φ_Δ contradicts Φ_Z . If we can extract from Φ_Q the implicit Φ_Δ that exactly characterizes the solution set of Φ_Q , then the satisfiability of $\Phi_Q \wedge \Phi_Z$ reduces to the satisfiability of $\Phi_\Delta \wedge \Phi_Z$. As a consequence, we can derive decision procedures for the combined theory by utilizing the decision procedures for PA and queues.

Definition 7 (Length Constraint Completion (LCC)). A formula $\Phi_\Delta(\bar{X})$ in \mathcal{L}_Z is a length constraint completion (LCC) for $\Phi_Q(\bar{X})$ if the following formulae are valid:

$$(\forall \bar{X} : \mathcal{Q}) \left[\Phi_Q(\bar{X}) \rightarrow (\exists \bar{z} : \mathcal{Z}) \left(\Phi_\Delta(\bar{z}) \wedge |\bar{X}| = \bar{z} \right) \right], \quad (2)$$

$$(\forall \bar{z} : \mathcal{Z}) \left[\Phi_\Delta(\bar{z}) \rightarrow (\exists \bar{X} : \mathcal{Q}) \left(\Phi_Q(\bar{X}) \wedge |\bar{X}| = \bar{z} \right) \right]. \quad (3)$$

(2) states that an LCC Φ_Δ for Φ_Q is *sound*: $|\cdot|$ maps a satisfying assignment in \mathcal{Q} to a satisfying assignment in PA. (3) states that Φ_Δ is *realizable*: any satisfying assignment in PA is an image under $|\cdot|$ of a satisfying assignment in \mathcal{Q} . Given Φ_Q , let Φ_Δ be an LCC, $\Phi_{\Delta+}$ (resp. $\Phi_{\Delta-}$) be the formula (when substituted for Φ_Δ) satisfying (2) (resp. (3)). If we identify these constraints with their corresponding solution sets, we have $\Phi_{\Delta-} \subseteq \Phi_\Delta \subseteq \Phi_{\Delta+}$. Thus Φ_Δ is the exact projection of Φ_Q from \mathcal{Q} to PA, while $\Phi_{\Delta+}$, $\Phi_{\Delta-}$ are over and under approximations of Φ_Δ respectively.

Example 5. Consider Φ_Q in Ex. 4 and $\Phi_{\Delta+} : \text{true}$, $\Phi_{\Delta-} : |X| = |Y| = 2$, and $\Phi_\Delta : |X| \neq |Y| \vee (|X| = |Y| \wedge |X| \neq 1)$. $\Phi_{\Delta+}$ is not realizable by Φ_Q because the integer assignment $\sigma_\Delta : \{|X| = |Y| = 1\}$ can not be realized. On the other hand, $\Phi_{\Delta-}$ is not sound because it does not satisfy the queue assignment $\sigma_Q : \{X = \epsilon_Q, Y = \epsilon_Q\}$. Finally, Φ_Δ is both sound and realizable w.r.t. Φ_Q and hence is an LCC for Φ_Q .

We have a decision procedure for $\text{Th}(\mathcal{Q}_Z)$ if Φ_Δ can be computed from Φ_Q .

Theorem 1. Let Φ_Δ be an LCC for Φ_Q . Then $\mathcal{Q}_Z \models \exists \Phi_Q \wedge \Phi_Z$ if and only if $\text{PA} \models \exists \Phi_\Delta \wedge \Phi_Z$.

To obtain an LCC, we need to normalize Φ_Q into an equivalent disjunction in which each disjunct is of the form $\Phi'_Q \wedge \theta'_Z$ with θ'_Z a newly generated integer constraint. We do not require the disjuncts to be mutually exclusive. First, we extend Def. 7 to deal with newly generated integer constraints in the normalization.

Definition 8 (Relativized Length Constraint Completion (RLCC)). A formula $\Phi_\Delta(\bar{X})$ is a length constraint completion for $\Phi_Q(\bar{X})$ relativized to $\theta_Z(\bar{X})$, (in short, $\Phi_\Delta(\bar{X})$ is an RLCC for $\Phi_Q(\bar{X})/\theta_Z(\bar{X})$), if the following formulae are valid:

$$(\forall \bar{X} : \mathcal{Q}) \left[\Phi_Q(\bar{X}) \wedge \theta_Z(\bar{X}) \rightarrow (\exists \bar{z} : \mathcal{Z}) \left(\Phi_\Delta(\bar{z}) \wedge |\bar{X}| = \bar{z} \right) \right], \quad (4)$$

$$(\forall \bar{z} : \mathcal{Z}) \left[\Phi_\Delta(\bar{z}) \rightarrow (\exists \bar{X} : \mathcal{Q}) \left(\Phi_Q(\bar{X}) \wedge \theta_Z(\bar{X}) \wedge |\bar{X}| = \bar{z} \right) \right]. \quad (5)$$

It is easily seen that an LCC is an RLCC with $\theta_Z \equiv \text{true}$ and RLCCs have the “additive” property.

Proposition 5. *If Φ_Δ is an RLCC for Φ_Q/θ_Z , then for any θ'_Z , $\Phi_\Delta \wedge \theta'_Z$ is also an RLCC for $\Phi_Q/(\theta_Z \wedge \theta'_Z)$.*

In particular, if $(\theta'_Z := \Phi_Z, \theta_Z := \text{true})$, Φ_Δ is an LCC for Φ_Q , then $\Phi_\Delta \wedge \Phi_Z$ is an RLCC for Φ_Q/Φ_Z . So Thm. 1 is in fact a special case of the following theorem.

Theorem 2. *Let Φ_Δ be an RLCC for Φ_Q/Φ_Z . Then $\mathfrak{Q}_Z \models \exists \Phi_Z \wedge \Phi_Q$ if and only if $\text{PA} \models \exists \Phi_\Delta$.*

This theorem motivates the strategy of our decision procedures. In the normalization process, with introduction of auxiliary integer constraints, we partition the original search space for Φ_Q such that $\Phi_Q \leftrightarrow \bigcup_i \Phi_Q^{(i)} \wedge \theta_Z^{(i)}$, until we easily compute the RLCC $\Phi_\Delta^{(i)}$ for each $\Phi_Q^{(i)}/\theta_Z^{(i)}$. By Prop. 5, $\Phi_\Delta^{(i)} \wedge \Phi_Z$ is an RLCC for $\Phi_Q^{(i)}/(\theta_Z^{(i)} \wedge \Phi_Z)$. Then $\mathfrak{Q}_Z \models \exists \Phi_Q \wedge \Phi_Z$ if and only if for some i , $\mathfrak{Q}_Z \models \exists \Phi_Q^{(i)} \wedge \theta_Z^{(i)} \wedge \Phi_Z$, which, by Thm. 2 (set $\Phi_Q := \Phi_Q^{(i)}, \Phi_\Delta := \Phi_\Delta^{(i)} \wedge \Phi_Z, \Phi_Z := \Phi_Z \wedge \theta_Z^{(i)}$), reduces to determining whether $\text{PA} \models \exists \Phi_\Delta^{(i)} \wedge \Phi_Z$ and $\mathfrak{Q} \models \exists \Phi_Q$. This leads to the following generic decision procedure.

Algorithm 3 (Generic Decision Procedure). *Input: $\Phi_Q \wedge \Phi_Z$.*

1. Return FAIL if $\mathfrak{Q} \not\models \exists \Phi_Q$.
2. For each partition $\Phi_Q^{(i)} \wedge \theta_Z^{(i)}$ of Φ_Q :
 - (a) Compute an RLCC $\Phi_\Delta^{(i)}$ for $\Phi_Q^{(i)}/\theta_Z^{(i)}$.
 - (b) Return SUCCESS if $\text{PA} \models \exists \Phi_\Delta^{(i)} \wedge \Phi_Z$.
3. Return FAIL.

Example 6. Revisiting Ex. 5, we partition Φ_Q into $(\Phi_Q \wedge |X| \neq |Y|) \vee (\Phi_Q \wedge |X| = |Y|)$. The first disjunct simplifies to $|X| \neq |Y|$ as $|X| \neq |Y|$ implies Φ_Q . Now consider the second disjunct. It is clear that the RLCC for $\Phi_Q/(|X| = |Y|)$ is $|X| = |Y| \wedge |X| \neq 1$.

5 Decision Procedure for $\text{Th}^\forall(\mathfrak{Q}_Z)$

We partition the search space for Φ_Q in a series of steps. When $|X|$ is known to be bounded by a constant l , we can instantiate X with a constant queue of length l . As \mathcal{A} is finite, there are only finitely many such queues.

First we assume $\Phi_Q \wedge \theta_Z$ satisfies the following condition.

Definition 9 (Equality Completion). Φ_Q is equality complete if $t_1 \neq t_2 \in \Phi_Q$ if and only if $|t_1| = |t_2| \in \theta_Z$.

To satisfy this condition, we first set $\theta_Z := \emptyset$ and for each $t_1 \neq t_2$, add either $|t_1| = |t_2|$ or $|t_1| \neq |t_2|$ to θ_Z . In the latter case, $t_1 \neq t_2$ can be removed from Φ_Q .

Definition 10 (Normal Form in \mathfrak{Q}_Z). Φ_Q is in normal form in \mathfrak{Q}_Z if Φ_Q satisfies Def. 5 and satisfies (i) if $\alpha X \neq Y\beta$ occurs with either $X \in \text{orb}(\alpha', k)$ or $Y \in \text{orb}(\beta', l)$, then $\alpha \equiv \epsilon_Q$; (ii) $\alpha X \neq Y\beta$ does not occur with both $X \in \text{orb}(\alpha', k)$ and $Y \in \text{orb}(\beta', l)$.

Algorithm 4 (Normalization in \mathfrak{Q}_Z).

1. Call Alg. 1 to normalize Φ_Q .
2. For all disequalities $\alpha X \neq Y\beta$ with $|X| < |\beta|$ or $|Y| < |\alpha|$, replace X and Y by instantiations. In the remaining steps we assume $|X| \geq |\beta|$ and $|Y| \geq |\alpha|$.

3. Consider each constraint of the form

$$\alpha X \neq Y\beta \wedge X \in \text{orb}(\alpha', k) \wedge Y \in \text{orb}(\beta', l), \quad (6)$$

which asserts that X is of the form $(\alpha')^* \alpha' [1..k]$ and similar for Y . If β is not a prefix of X or α is not a prefix of Y , $\alpha X \neq Y\beta$ simplifies to **true**. Otherwise $\alpha X \neq Y\beta$ can be replaced by $X = X'\beta \wedge Y = \alpha Y' \wedge X' \neq Y'$ which can be further reduced to

$$X' \in \text{orb}(\alpha', k') \wedge Y' \in \text{orb}(\beta'', l') \wedge X' \neq Y', \quad (7)$$

where

$$\begin{aligned} k' &= (k + |\alpha'| - (|\beta| \bmod |\alpha'|)) \bmod |\alpha'|, \\ \beta'' &= \beta' [(|\alpha| \bmod |\beta'|) + 1..|\beta'|] \circ \beta' [1..(|\alpha| \bmod |\beta'|)], \\ l' &= (l + |\beta'| - (|\alpha| \bmod |\beta'|)) \bmod |\beta'|. \end{aligned}$$

If $\alpha' = \beta''$, then $k' = l'$, because $|X'| = |Y'|$. Thus (7) is false and so is (6). If $\alpha' \neq \beta''$, then there are only finitely many cases that $X' = Y'$ which can be computed and excluded.

4. Consider each constraint of the form $\alpha X \neq Y\beta \wedge X \in \text{orb}(\alpha', k)$. Guess a word α'' such that $|\alpha''| = |\alpha|$ and set $Y = \alpha'' Y'$. For $\alpha \neq \alpha''$, replace $\alpha X \neq Y\beta$ by $Y = \alpha'' Y'$, otherwise, replace $\alpha X \neq Y\beta$ by $Y = \alpha Y' \wedge X \neq Y'\beta$.
5. Consider each constraint of the form $\alpha X \neq Y\beta \wedge Y \in \text{orb}(\beta', l)$. If α is not a prefix of Y (which has the form $(\beta')^* \beta' [1..l]$), $\alpha X \neq Y\beta \wedge Y \in \text{orb}(\beta', l)$ simplifies to **true**. Otherwise $\alpha X \neq Y\beta$ can be replaced by $Y = \alpha Y' \wedge X \neq Y'\beta$, which can be further simplified to $Y' \in \text{orb}(\beta'', l') \wedge X \neq Y'\beta$, with β'' and l' the same as in step 3.

Algorithm 5 (Computation of $\Phi_{\Delta+}$). *Input:* $\Phi_Q \wedge \theta_Z$. Initially set $\Phi_{\Delta+} = \emptyset$. Add to $\Phi_{\Delta+}$: (1) $|t_1| = |t_2|$, if $t_1 \neq t_2$ or $t_1 = t_2$; (2) $|X| + |\alpha| = |\alpha X| = |X\alpha|$, if αX or $X\alpha$ occurs; (3) $|X| \equiv k \pmod{|\alpha|}$, if $X \in \text{orb}(\alpha, k)$.

Φ_Q can be satisfied by sufficiently long queues: there exists a *cutpoint* δ such that if $\mathfrak{Q} \models \exists \Phi_Q$, then for any solution $(l_i)_n$ (i.e., l_0, \dots, l_n) for $\Phi_{\Delta+}$ such that $l_i \geq \delta$, there exists a solution $(\alpha_i)_n$ for Φ_Q such that $|\alpha_i| = l_i$. Let $C_\Phi(\delta)$ denote $\bigwedge_{X \in \mathcal{V}_Q(\Phi_Q)} |X| \geq \delta$. It is clear that $\Phi_{\Delta+} \wedge C_\Phi(\delta) \wedge \theta_Z$ is an RLCC for Φ_Q / θ_Z . It is not true, however, that δ is the smallest $\max\{(\mu_i)_n\}$ such that $\mathfrak{Q}_Z \models \exists \Phi_Q \wedge \bigwedge_{i=1}^n |X_i| = \mu_i$ where $(X_i)_n$ enumerate $\mathcal{V}_Q(\Phi_Q)$. Ex. 4 shows an anomaly where $\{X := \epsilon_Q, Y := \epsilon_Q\}$ is a solution for Φ_Q (with $|X| = |Y| = 0$), while there exists no solution for Φ_Q such that $|X| = |Y| = 1$. To avoid such anomalies we separate the search for a satisfying assignment into two cases. We compute a cut length $L_t \geq \delta$ and enumerate all assignments σ with $\llbracket |X| \rrbracket \sigma < L_t$, while for $\llbracket |X| \rrbracket \sigma \geq L_t$ satisfiability of the queue constraints is reduced to satisfiability of integer constraints as in [13, 14].

The computation of L_t is based on the observation that an assignment σ is satisfying if every $\llbracket |X| \rrbracket \sigma$ includes a unique “marker” at the same, fixed, position. Such a marker can be constructed by concatenating a “shortest unused prefix” and a unique identifier for each queue variable. Let PRE_Φ denote the set of all words α such that αX or α is a proper term in Φ_Q . A word q is called a *delimiter* of Φ_Q if q is strongly primitive and $q \notin \text{orb}(\alpha)$ for any $\alpha \in \text{PRE}_\Phi$. Let d_p denote an

arbitrary shortest delimiter (there can be more than one) and let $L_p = |d_p|$. Let L_c be the smallest number of letters necessary to create a unique identifying word, called a *color*, for each queue variable in Φ_Q . We claim that $L_c + L_p = L_t \geq \delta$.

Example 7 (Computation of L_t). Consider again Φ_Q in Ex. 4. Here $\text{PRE}_\Phi = \{ab, ba, aa\}$; a shortest delimiter is *aab*, and thus $L_p = 3$. Φ_Q includes two queue variables, requiring one letter to identifying them with two letters in the alphabet. Thus, we need four letters to construct a unique identifying word, resulting in $L_t = 4$.

Proposition 6 (RLCC in \mathfrak{Q}_Z). $\Phi_{\Delta^+} \wedge C_\Phi(L_t) \wedge \theta_Z$ is an RLCC for Φ_Q/θ_Z .

Definition 11 (Length Configuration in \mathfrak{Q}_Z). A length configuration for Φ_Q (in \mathfrak{Q}_Z) is a conjunction $\bigwedge_{X \in \mathcal{V}_Q(\Phi_Q)} A_X$, where A_X is either $|X| = i$ (for some $i < L_t$) or $|X| \geq L_t$.

Let C be the set of all configurations. Clearly C creates a finite partition of the search space that includes $C_\Phi(L_t)$. A partial assignment ∂ is *compatible* with a configuration C if for any variable X , $\llbracket X \rrbracket \partial$ is defined iff $|X| = i$ (for some $i < L_t$) occurs in C . The empty assignment is vacuously a satisfying partial assignment, the only one compatible with $C_\Phi(L_t)$. As a consequence of Prop. 6, we have

Algorithm 6 (Decision Procedures for \mathfrak{Q}_Z). *Input:* $\Phi_Q \wedge \theta_Z \wedge \Phi_Z$ where $\Phi_Q \wedge \theta_Z$ denotes one of the partitions.

1. For each $C \in C$,
 - (a) Guess a satisfying ∂ compatible with C and update Φ_{Δ^+} , C , θ_Z and Φ_Z .
 - (b) If succeed, return SUCCESS if $\text{PA} \models \exists \Phi_{\Delta^+} \wedge C \wedge \theta_Z \wedge \Phi_Z$.
2. Return FAIL.

6 Quantifier Elimination for $\text{Th}(\mathfrak{Q}_Z)$

In this section we present a quantifier elimination for the first-order theory of queues with integers, $\text{Th}(\mathfrak{Q}_Z)$. The procedure removes a block of quantifiers of the same type in a single step.

It is well-known that eliminating arbitrary quantifiers reduces to eliminating existential quantifiers from formulae in the form $(\exists \bar{x})[A_1(\bar{x}) \wedge \dots \wedge A_n(\bar{x})]$, where $A_i(\bar{x})$ ($1 \leq i \leq n$) are literals [5]. By parameters we mean the *implicitly universally quantified variables*. We use \bar{Y} to denote a sequence of Q -parameters.

The elimination procedure consists of the following two subprocedures.

Elimination of Quantifiers on Integer Variables We assume formulas with quantifiers on integer variables are in the form

$$(\exists \bar{u} : \mathbb{Z}) \left[\Phi_Q(\bar{X}) \wedge \Phi_Z(\bar{u}, \bar{v}, \bar{X}) \right], \quad (8)$$

where $\bar{X} \subseteq \mathcal{V}_Q$ and $\bar{v}, \bar{u} \subseteq \mathcal{V}_Z$. Since $\Phi_Q(\bar{X})$ does not contain \bar{u} , we can move them out of the scope of $(\exists \bar{u})$, and then obtain

$$\Phi_Q(\bar{X}) \wedge (\exists \bar{u} : \mathbb{Z}) \Phi_Z(\bar{u}, \bar{v}, \bar{X}). \quad (9)$$

Since in $\Phi_Z(\bar{u}, \bar{v}, \bar{X})$, \bar{X} occurs as pseudo integer variables, $(\exists \bar{u} : \mathbb{Z}) \Phi_Z(\bar{u}, \bar{v}, \bar{X})$ is essentially a Presburger formula and we can proceed to remove the quantifier using Cooper's method [3]. In fact we can defer the elimination until all other types of quantifiers are removed.

Elimination of Quantifiers on Queue Variables We assume formulas with quantifiers on queue variables are in the form

$$(\exists \bar{X} : \mathcal{Q}) \left[\Phi_{\mathcal{Q}}(\bar{X}, \bar{Y}) \wedge \Phi_{\mathcal{Z}}(\bar{u}, \bar{X}, \bar{Y}) \right], \quad (10)$$

where $\bar{X}, \bar{Y} \subseteq \mathcal{V}_{\mathcal{Q}}$, $\bar{u} \subseteq \mathcal{V}_{\mathcal{Z}}$, and $\Phi_{\mathcal{Z}}(\bar{u}, \bar{X}, \bar{Y})$ can be an arbitrary Presburger formula (not necessarily quantifier-free). By Prop. 1, we can assume \bar{X} does not occur in selectors. Though elimination of selectors in general adds more existential quantifiers of sort queue or atom, the newly added quantifiers will be removed together with the original ones.

We need to extend the notion of RLCC to deal with parameters.

Definition 12 (RLCC with parameters). Consider $(\exists \bar{X} : \mathcal{Q}) \left[\Phi_{\mathcal{Q}}(\bar{X}, \bar{Y}) \wedge \theta_{\mathcal{Z}}(\bar{X}, \bar{Y}) \right]$, where \bar{Y} are parameters. Let $\Phi_{\mathcal{Q}}^{(2)}(\bar{Y})$ be the maximum subset of $\Phi_{\mathcal{Q}}(\bar{X}, \bar{Y})$ not containing \bar{X} and $\Phi_{\mathcal{Q}}^{(1)}(\bar{X}, \bar{Y}) := \Phi_{\mathcal{Q}}(\bar{X}, \bar{Y}) \setminus \Phi_{\mathcal{Q}}^{(2)}(\bar{Y})$. A formula $\Phi_{\Delta}(\bar{X}, \bar{Y})$ is an RLCC in \bar{X} for $\Phi_{\mathcal{Q}}(\bar{X}, \bar{Y})$ relativized to $\theta_{\mathcal{Z}}(\bar{X}, \bar{Y})$, (in short, $\Phi_{\Delta}(\bar{X}, \bar{Y})$ is an RLCC for $\Phi_{\mathcal{Q}}(\bar{X}, \bar{Y})/\bar{X}/\theta_{\mathcal{Z}}(\bar{X}, \bar{Y})$), if the following hold:

$$(\forall \bar{X}, \bar{Y} : \mathcal{Q}) \left[\Phi_{\mathcal{Q}}(\bar{X}, \bar{Y}) \wedge \theta_{\mathcal{Z}}(\bar{X}, \bar{Y}) \rightarrow (\exists \bar{z} : \mathcal{Z}) \left(\Phi_{\Delta}(\bar{z}, \bar{Y}) \wedge |\bar{X}| = \bar{z} \right) \right], \quad (11)$$

$$\begin{aligned} (\forall \bar{Y} : \mathcal{Q})(\forall \bar{z} : \mathcal{Z}) \left[\Phi_{\mathcal{Q}}^{(2)}(\bar{Y}) \wedge \Phi_{\Delta}(\bar{z}, \bar{Y}) \right. \\ \left. \rightarrow (\exists \bar{X} : \mathcal{Q}) \left(\Phi_{\mathcal{Q}}(\bar{X}, \bar{Y}) \wedge \theta_{\mathcal{Z}}(\bar{X}, \bar{Y}) \wedge |\bar{X}| = \bar{z} \right) \right]. \end{aligned} \quad (12)$$

We also need to update the notion of normal form for parameters.

Definition 13 (Normal Form in $\mathfrak{Q}_{\mathcal{Z}}$ with \bar{Y}). $\Phi_{\mathcal{Q}}$ is in normal form in $\mathfrak{Q}_{\mathcal{Z}}$ (with parameters) if $\Phi_{\mathcal{Q}}$ satisfies Def. 10 and the following condition: if $\alpha X \beta \neq t(Y)$ (where Y is a parameter) appears in $\Phi_{\mathcal{Q}}$, then $\alpha \equiv \beta \equiv \epsilon_{\mathcal{Q}}$ and X does not occur in literals of the form $X \in \text{orb}(\alpha, k)$.

We treat \mathcal{Q} -terms of the form $t(Y)$ as distinct variables. Let L_c, L_p and L_t be as defined in Sec. 5 and we obtain $C_{\phi}(L_t)$ and $\Phi_{\Delta^+}(\bar{X}, \bar{Y})$ accordingly.

Proposition 7 (RLCC in $\mathfrak{Q}_{\mathcal{Z}}$ with \bar{Y}). $\Phi_{\Delta^+}(\bar{X}, \bar{Y}) \wedge C_{\phi}(L_t) \wedge \theta_{\mathcal{Z}}(\bar{X}, \bar{Y})$ is an RLCC for $\Phi_{\mathcal{Q}}(\bar{X}, \bar{Y})/\bar{X}/\theta_{\mathcal{Z}}(\bar{X}, \bar{Y})$.

We guess and add a $C \in \mathcal{C}$ to (10). First we remove each X such that $|X| = i$ ($i < L_t$) occurs in C . For the variables left in \bar{X} , we have $|X| \geq L_t$ in C and so we can assume C is $C_{\phi}(L_t)$. Then (10) is rewritten as

$$(\exists \bar{X} : \mathcal{Q}) \left[\Phi_{\mathcal{Q}}(\bar{X}, \bar{Y}) \wedge \theta_{\mathcal{Z}}(\bar{X}, \bar{Y}) \wedge \Phi_{\mathcal{Z}}(\bar{u}, \bar{X}, \bar{Y}) \right], \quad (13)$$

which is equivalent to

$$\Phi_{\mathcal{Q}}^{(2)}(\bar{Y}) \wedge (\exists \bar{v} : \mathcal{Z}) \left[\Phi_{\Delta^+}(\bar{v}, \bar{Y}) \wedge C_{\phi}(L_t) \wedge \theta_{\mathcal{Z}}(\bar{v}, \bar{Y}) \wedge \Phi_{\mathcal{Z}}(\bar{u}, \bar{v}, \bar{Y}) \right]. \quad (14)$$

7 Conclusion

We presented decision procedures for the theory of queues with integer constraints. Our method combines the extraction of integer constraints from queue constraints, and in case of the quantified theory, with a reduction of quantifiers on queue variables to quantifiers on integer variables.

Complexity Clearly $\text{Th}^\forall(\mathcal{Q}_Z)$ is NP-hard as it is a super theory of $\text{Th}^\forall(\mathcal{Q})$ and $\text{Th}^\forall(\mathbb{Z})$, which are both NP-complete. Alg. 5 computes Φ_{Δ^+} in $O(n)$ and L_t, L_t^+ are also in $O(n)$. By the nondeterministic nature of our algorithms, we can show that each branch of computation in the normalization procedures and in Algs. 2, 3, 6 is in P. Therefore $\text{Th}^\forall(\mathcal{Q}_Z)$ is NP-complete and consequently, for $\text{Th}(\mathcal{Q}_Z)$, the elimination of a block of existential quantifiers, regardless of the size of the block, can be done in $O(2^n)$.

Future Work We plan to extend our results to the theory of queues in a more expressive signature, e.g., in the language with prefix, suffix and subqueue relation, and investigate decidability of the first-order theory of queues with integers and prefix or suffix relation. Note that for the first-order theory we cannot obtain decidability with both prefix and suffix relations, nor in a signature with prefix (or suffix) relation and all constructors, because both extensions are sufficiently expressive to interpret the theory of arrays.

References

1. Michael Benedikt, Leonid Libkin, Thomas Schwentick, and Luc Segoufin. A model-theoretic approach to regular string relations. In *Proceedings of 16th IEEE Symposium on Logic in Computer Science (LICS'01)*, pages 431–440. IEEE Computer Society Press, 2001.
2. Nikolaj S. Bjørner. *Integrating Decision Procedures for Temporal Verification*. PhD thesis, Computer Science Department, Stanford University, November 1998.
3. D. C. Cooper. Theorem proving in arithmetic without multiplication. In *Machine Intelligence*, volume 7, pages 91–99. American Elsevier, 1972.
4. H. B. Enderton. *A Mathematical Introduction to Logic*. Academic Press, 2001.
5. Wilfrid Hodges. *Model Theory*. Cambridge University Press, Cambridge, UK, 1993.
6. Felix Klaedtke and Harald Rueß. Monadic second-order logics with cardinalities. In *30th International Colloquium on Automata, Languages and Programming (ICALP'03)*, volume 2719 of LNCS. Springer-Verlag, 2003.
7. M. Lothaire. *Combinatorics on Words*. Addison-Wesley, Massachusetts, USA, 1983.
8. Greg Nelson and Derek C. Oppen. Simplification by cooperating decision procedures. *ACM Transaction on Programming Languages and Systems*, 1(2):245–257, October 1979.
9. C. R. Reddy and D. W. Loveland. Presburger arithmetic with bounded quantifier alternation. In *Proceedings of the 10th Annual Symposium on Theory of Computing (STOC'78)*, pages 320–325. ACM Press, 1978.
10. Tatiana Rybina and Andrei Voronkov. A decision procedure for term algebras with queues. In *Proceedings of 15th IEEE Symposium on Logic in Computer Science (LICS'00)*, pages 279 – 290. IEEE Computer Society Press, 2000.
11. Tatiana Rybina and Andrei Voronkov. Upper bounds for a theory of queues. In *Proceedings of 30th International Colloquium on Automata, Languages and Programming (ICALP'03)*, volume 2719 of LNCS, pages 714–724. Springer-Verlag, 2003.
12. Wolfgang Thomas. Infinite trees and automaton-definable relations over ω -words. *Theoretical Computer Science*, 103:143–159, 1992.
13. Ting Zhang, Henny B. Sipma, and Zohar Manna. Decision procedures for recursive data structures with integer constraints. In *the 2nd International Joint Conference on Automated Reasoning (IJCAR'04)*, volume 3097 of LNCS, pages 152–167. Springer-Verlag, 2004.
14. Ting Zhang, Henny B. Sipma, and Zohar Manna. Term algebras with length function and bounded quantifier alternation. In *the 17th International Conference on Theorem Proving in Higher Order Logics (TPHOLS'04)*, volume 3223 of LNCS, pages 321–336. Springer-Verlag, 2004.