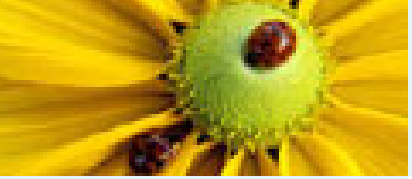
Decision Procedures for Queues with Integer Constraints

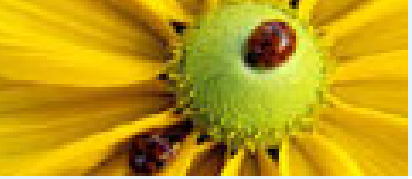
Ting Zhang, Henny B. Sipma, Zohar Manna

Stanford University

tingz,sipma,zm@cs.stanford.edu

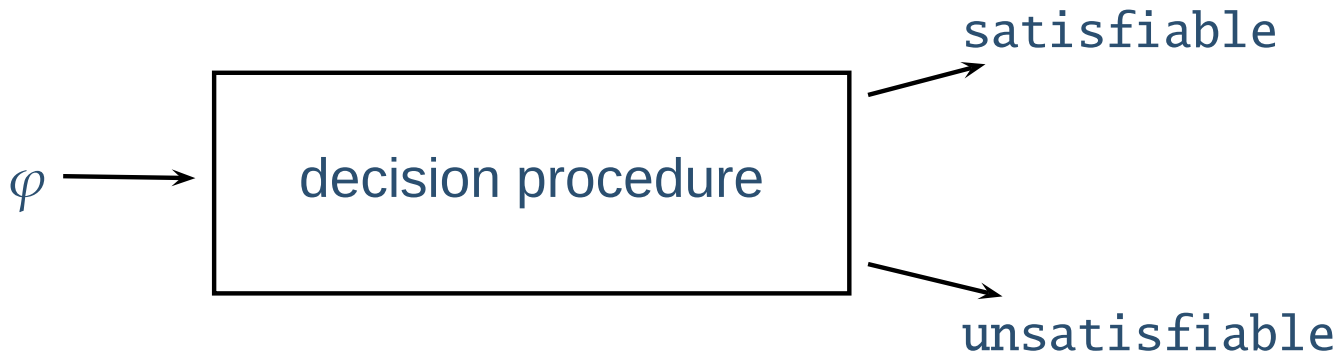


Introduction



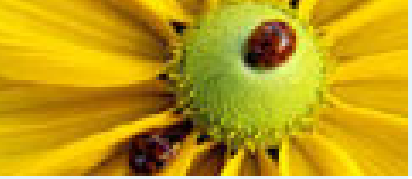
What is a Decision Procedure?

An algorithm that checks whether a formula is valid in a given decidable theory.



Always terminates with either a positive or a negative answer.

Relieve users from tedious interaction with theorem prover.



Why Do We Need New Decision Procedures?

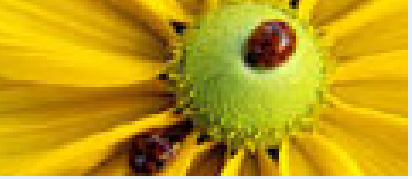
Decision procedures exist for specific theories

- Arithmetic: integers, reals, . . . ,
- Data types: lists, queues, arrays, sets, multisets, . . . ,
- Algebraic structures: linear dense orders . . . ,

But

- programming languages involve multiple theories.
- verification conditions do not belong to a single theory.

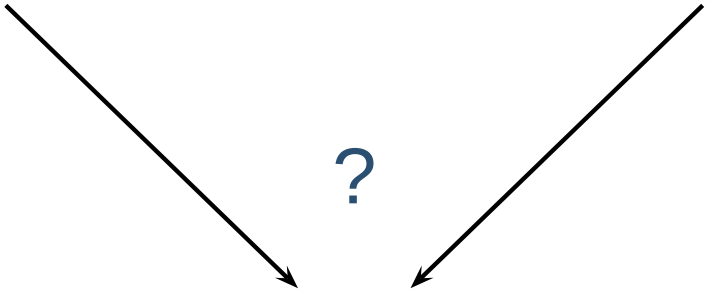
☞ We need to reason about *mixed* constraints from multiple theories.



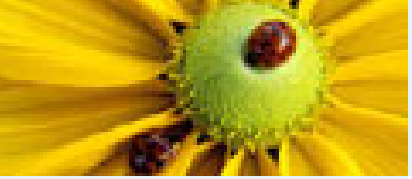
What is Combining Decision Procedure?

Σ_1-theory T_1 | Σ_2-theory T_2

$\boxed{P_1}$ for T_1-satisfiability | $\boxed{P_2}$ for T_2-satisfiability



$\boxed{P}$ for $(T_1 \cup T_2)$-satisfiability



Combination of Theories

General Framework:

Nelson-Oppen Combination Method [NO79]

Recent Advances:

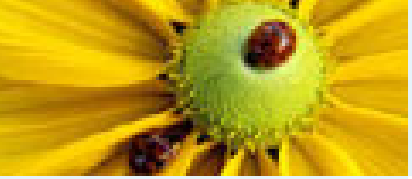
- Non-disjoint Signature.

  Tinelli and Ringeissen [TR03]
- Model-theoretic.

  Ghilardi [Ghi05]
- Proof-theoretic.

  Zarba [Zar02]

  Armando, Ranise and Rusinowitch [ARR01]

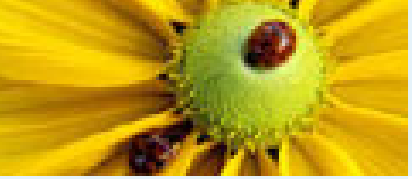


Limitation

- All existing combination techniques impose severe restrictions on the theories to be combined.
- None of the techniques is applicable to multi-sorted theories with functions connecting the different sorts.

☞ Logic theories are *fragile*.

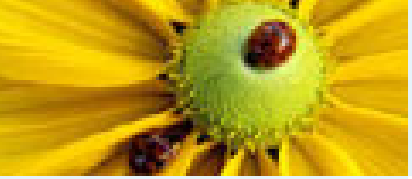
- Nelson-Oppen combination should be viewed as exceptional.
- Why should modular combinations always exists?
- Concentrate on concrete problems instead of looking for grand scheme.



Our Contributions

- Integration of recursive data structures with integers
  - Term algebras (tree-like objects) + integers

    [ZSM04a] IJCAR04, [ZSM04b] TPHOLs04,
  - Queues (linear objects)+ integers
- Why? To automatically decide the validity of verification conditions arising in the analysis of any property involving data structures and size, for example
  - buffer overflows
  - array out of bounds
  - memory overflow
  - . . .



Our Approach

Exploit the algebraic properties of constituting theories.

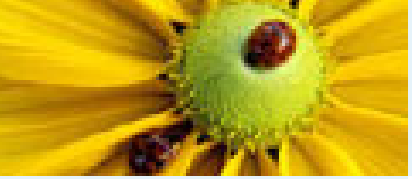
- For quantifier-free combinations:

  Extract exact integer constraints **induced** by constraints of data types.

- For quantified combinations:

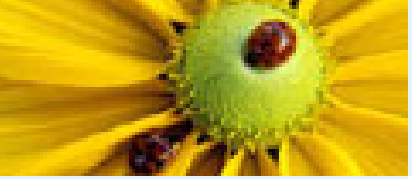
  Reduce quantifiers on data types to quantifiers on integers.

☞ Reduce theories of data domain to the theory of integer domain.



Outline

- Previous Work on Queues.
- Queues: Language, Structures and Properties.
- Decision Procedure for Queues.
- Decision Procedure for Queues with Integers.
- Conclusion and Future Work.



Previous Work on Queues

- Quantifier-free theory.

  Bjørner [Bjø98]

- Quantified theory.

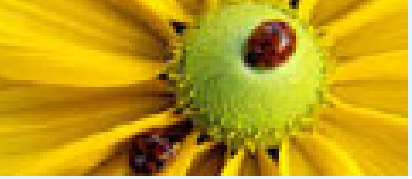
  Rybina and Voronkov [RV00] [RV03]

- With Prefix Relation.

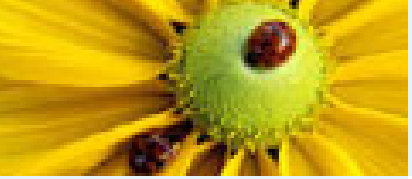
  Benedikt, Libkin, Schwentick and Segoufin [BLSS01]

- WS1S with cardinality constraints.

  Klaedtke and Ruess [KR03]



Queues: Language, Structures and Properties



Queues (1)

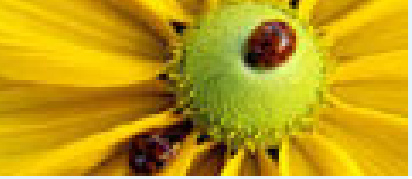
$\mathfrak{Q}$: $\langle Q; \mathcal{A}, C, S \rangle$:

1. $\mathcal{A}$: Constants: $a, b, c, \ldots$
2. Q: Sequences of constants. ϵ_Q: the empty queue.
3. C: Constructors:

Left Insertion $\mathrm{la} : \mathcal{A} \times Q \to Q$

Right Insertion $\mathrm{ra} : \mathcal{A} \times Q \to Q$, s.t.

$$\mathrm{la}(a, \epsilon_Q) = \mathrm{ra}(a, \epsilon_Q) = \langle a \rangle,$$
$$\mathrm{la}(a, \langle s_1, \ldots, s_n \rangle) = \langle a, s_1, \ldots, s_n \rangle,$$
$$\mathrm{ra}(a, \langle s_1, \ldots, s_n \rangle) = \langle s_1, \ldots, s_n, a \rangle.$$



Queues (2)

4 $\mathcal{S}$: Selectors:

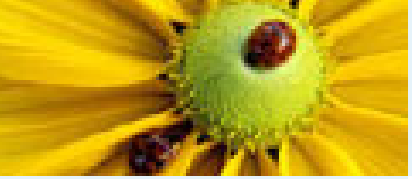
Left Head $\mathrm{lh} : \mathcal{Q} \to \mathcal{A}$, *Left Tail* $\mathrm{lt} : \mathcal{Q} \to \mathcal{Q}$,

Right Head $\mathrm{rh} : \mathcal{Q} \to \mathcal{A}$, *Right Tail* $\mathrm{rt} : \mathcal{Q} \to \mathcal{Q}$, s.t.

$$\mathrm{lh}(\langle s_1, \ldots, s_n \rangle) = s_1,$$
$$\mathrm{lt}(\langle s_1, \ldots, s_n \rangle) = \langle s_2, \ldots, s_n \rangle,$$
$$\mathrm{rh}(\langle s_1, \ldots, s_n \rangle) = s_n,$$
$$\mathrm{rt}(\langle s_1, \ldots, s_n \rangle) = \langle s_1, \ldots, s_{n-1} \rangle.$$

Convention: use *concatenation operator* $\circ$.

$a \circ X \circ b$ *stands for* $\mathrm{ra}(b, \mathrm{la}(a, X))$ *or* $\mathrm{la}(a, \mathrm{ra}(b, X))$.



Orbit

$\mathrm{orb}(\beta)$ is the *orbit* of all words of the form $\beta^*\beta[1..i]$ ($i < |\beta|$) and $\mathrm{orb}(\beta, k)$ is the subtrack of $\mathrm{orb}(\beta)$ ending with $\beta[1..k]$.

$$\mathrm{ext}(\beta, m, k) \stackrel{\Delta}{=} \beta^m \beta[1..k],$$

$$\mathrm{orb}(\beta, k) \stackrel{\Delta}{=} \{\, \mathrm{ext}(\beta, m, k) \mid m \geq 0 \,\}$$

$$\mathrm{orb}(\beta) \stackrel{\Delta}{=} \bigcup_{k \geq 0} \mathrm{orb}(\beta, k).$$

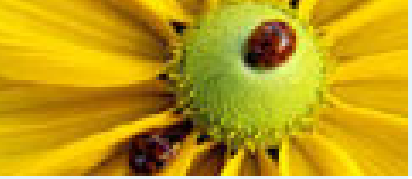
Example:

$$\mathrm{ext}(aba, 1, 2) = abaab,$$

$$\mathrm{orb}(aba, 0) = (aba)^*,$$

$$\mathrm{orb}(aba, 1) = (aba)^* a,$$

$$\mathrm{orb}(aba, 2) = (aba)^* ab.$$

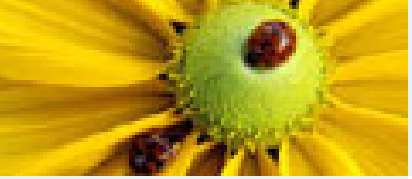


Primitive Words

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 $\beta \notin \mathrm{orb}(\alpha)$.

Example:

$abab$	non-primitive
aba	primitive
abb	strongly-primitive

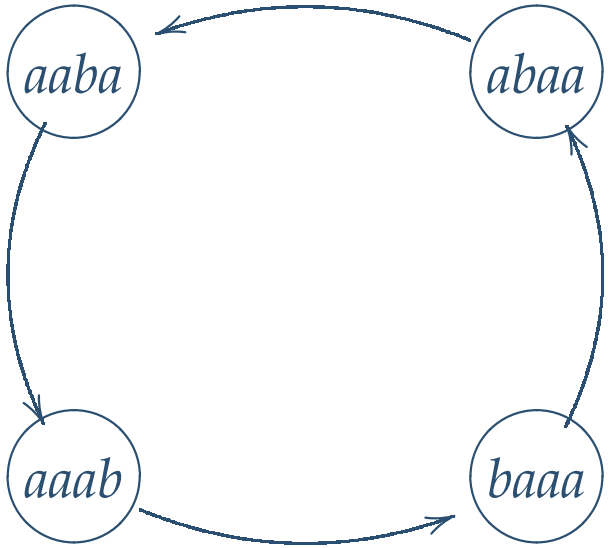


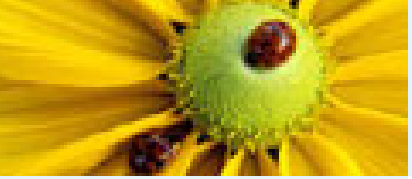
Conjugacy (1)

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 $|u| = k$.

Example:





Conjugacy (2)

Existence:

Two words α and β are conjugate if and only if

$$(\exists \gamma)\, \alpha\gamma = \gamma\beta.$$

Moreover, α and β are k-conjugate if and only if

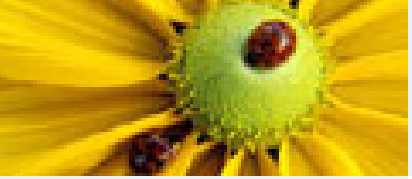
$$(\forall \gamma)\, [\alpha\gamma = \gamma\beta \;\leftrightarrow\; \gamma \in \mathrm{orb}(\alpha, k)].$$

If $\alpha X = X\beta$ has a solution, then

$$\exists u_1, u_2\, [\alpha = u_1 u_2 \wedge \beta = u_2 u_1],$$

and all solutions are of the form

$$(u_1 u_2)^* u_1.$$



Conjugacy (3)

Mutual Exclusion:

A conjunction of literals of the form

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

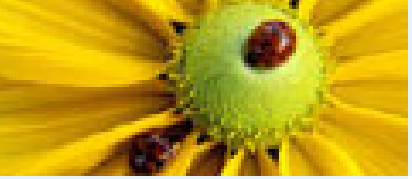
can be simplified such that

$$m = 0 \ \vee \ (m = 1 \wedge n = 0).$$

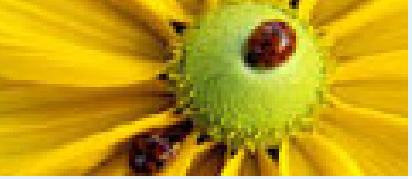
Example:

$$X \in \mathrm{orb}(ab) \wedge X \in \mathrm{orb}(aba) \mapsto X \in \{a, ab, aba\},$$

$$X \in \mathrm{orb}(ab) \wedge X \notin \mathrm{orb}(aba) \mapsto X \in \mathrm{orb}(ab) \wedge X \notin \{a, ab, aba\}.$$



Decision Procedure for Queues

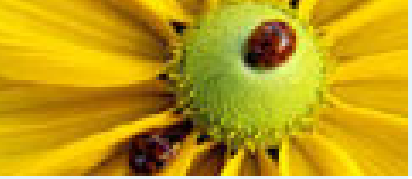


Decision Procedure for Queues (Bjørner)

Input: $\Phi \equiv \mathcal{E} \cup \mathcal{D}$.

1. Normalize Φ to $\Phi' : \mathcal{E}' \cup \mathcal{D}'$.
2. Return *FAIL*, if inconsistency is discovered;

   Return *SUCCESS*.



Normal Form in Ω

A queue constraint Φ_Q is in *normal form* if

- all equalities are in triangular form,
- for each X there exists at most one literal $X \in \mathrm{orb}(\alpha, k)$,
- if $X \in \mathrm{orb}(\alpha, k)$ occurs, then no $X \notin \mathrm{orb}(\alpha', k')$ occurs, and
- disequalities are in the form $\alpha X \neq Y\beta$ for $X \not\equiv Y$.

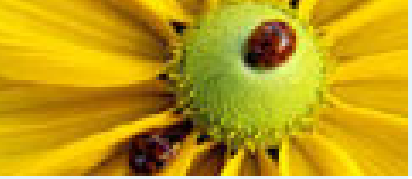
Example:

$$\surd \quad Y = bZ \wedge Z = ba \wedge X \in orb(ab, 1) \wedge X \neq Yab$$

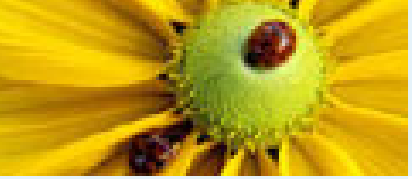
$$\times \quad Y = bZ \wedge Z = ba \wedge X \in orb(ab, 1) \wedge X \notin orb(aba, 0)$$

$$\times \quad Y = bZ \wedge Z = ba \wedge X \in orb(ab, 1) \wedge X \in orb(ba, 1)$$

$$\times \quad Y = bZ \wedge Z = ba \wedge X \in orb(ab, 1) \wedge Xa \neq Yab$$



Decision Procedure for Queues with Integers

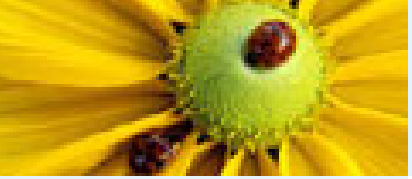


Queues with Integers

Presburger arithmetic (PA): $\mathscr{L}_{\mathbb{Z}}$, PA.

Two-sorted language $\Sigma = \Sigma_{\mathcal{Q}} \cup \Sigma_{\mathbb{Z}} \cup \{|\cdot|\}$:

1. $\Sigma_{\mathcal{Q}}$: signature of queues.
2. $\Sigma_{\mathbb{Z}}$: signature of Presburger arithmetic.
3. $|\cdot| : \mathfrak{Q} \to \mathbb{N}$, the length function.



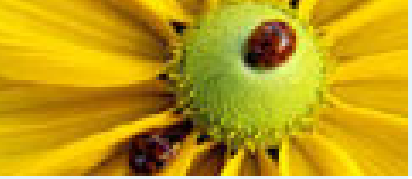
Problem I

The presence of $\Phi_{\mathbb{Z}}$ restricts solutions to Φ_Q.

Example: Suppose $\mathcal{A} = \{a, b\}$. Then

$$\Phi_Q : \quad Xba \neq abY \ \wedge \ Xab \neq baY \ \wedge \ Xaa \neq baY \ \wedge \ Xab \neq aaY$$

is not satisfiable with $\Phi_{\mathbb{Z}} : |X| = |Y| = 1$.



Problem I

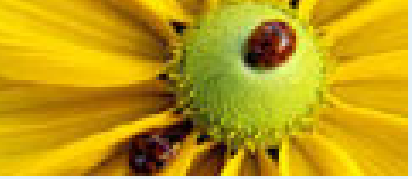
The presence of $\Phi_{\mathbb{Z}}$ restricts solutions to Φ_Q.

Example: Suppose $\mathcal{A} = \{a, b\}$. Then

$$\Phi_Q : \quad Xba \neq abY \ \wedge \ Xab \neq baY \ \wedge \ Xaa \neq baY \ \wedge \ Xab \neq aaY$$

is not satisfiable with $\Phi_{\mathbb{Z}} : |X| = |Y| = 1$.

Compute an *accurate length constraint*!



Length Constraint Completion (LCC)

A formula $\Phi_\Delta(\bar{X})$ is an *LCC* for $\Phi_Q(\bar{X}) \wedge \Phi_{\mathbb{Z}}(\bar{X})$, if the following formulae are valid:

$$\Phi_Q(\bar{X}) \wedge \Phi_{\mathbb{Z}}(\bar{X}) \rightarrow (\exists \bar{z} : \mathbb{Z}) \left(\Phi_\Delta(\bar{z}) \wedge |\bar{X}| = \bar{z} \right),$$

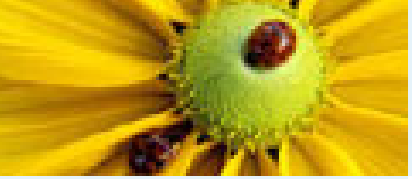
$$\Phi_\Delta(\bar{z}) \rightarrow (\exists \bar{X} : Q) \left(\Phi_Q(\bar{X}) \wedge \Phi_{\mathbb{Z}}(\bar{X}) \wedge |\bar{X}| = \bar{z} \right).$$

Informally,

$$\Phi_Q(\bar{X}) \wedge \Phi_{\mathbb{Z}}(\bar{X}) \quad \text{“} \leftrightarrow \text{”} \quad \Phi_\Delta(\bar{X})$$

☞ $\Phi_\Delta(\bar{X})$ fully characterizes $\Phi_Q(\bar{X}) \wedge \Phi_{\mathbb{Z}}(\bar{X})$.

☞ We reduce the combined constraint to the integer domain!



LCC (2)

Let $\Phi_{\Delta+}$ be the formula that (when in place of Φ_{Δ}) satisfies

$$\Phi_Q(\bar{X}) \wedge \Phi_{\mathbb{Z}}(\bar{X}) \rightarrow (\exists \bar{z} : \mathbb{Z}) \left(\Phi_{\Delta}(\bar{z}) \wedge |\bar{X}| = \bar{z} \right).$$

$\Phi_{\Delta+}$ is *sound*:

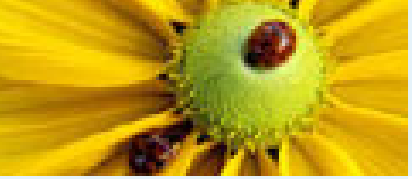
$|\cdot|$ maps a satisfying σ_Q in Q to a satisfying $\sigma_{\mathbb{Z}}$ in PA.

Let $\Phi_{\Delta-}$ be the formula that (when in place of Φ_{Δ}) satisfies

$$\Phi_{\Delta}(\bar{z}) \rightarrow (\exists \bar{X} : Q) \left(\Phi_Q(\bar{X}) \wedge \Phi_{\mathbb{Z}}(\bar{X}) \wedge |\bar{X}| = \bar{z} \right)$$

$\Phi_{\Delta-}$ is *complete*:

any satisfying $\sigma_{\mathbb{Z}}$ in PA is an image under $|\cdot|$ of a satisfying σ_Q in Q.



LCC (3)

Identify constraints with the corresponding solution set.

$\Phi_{\Delta+}$ is an *over-approximation* of Φ_{Δ}:

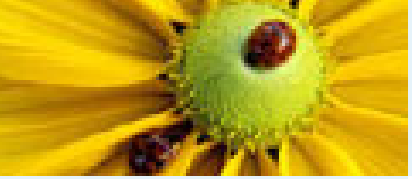
$$\Phi_{\Delta} \subseteq \Phi_{\Delta+}$$

$\Phi_{\Delta-}$ is an *under-approximation* of Φ_{Δ}:

$$\Phi_{\Delta-} \subseteq \Phi_{\Delta}$$

Φ_{Δ} is *unique* up to equivalence:

$$\Phi_{\Delta'} \subseteq \Phi_{\Delta} \subseteq \Phi_{\Delta'}$$



Example: LCC

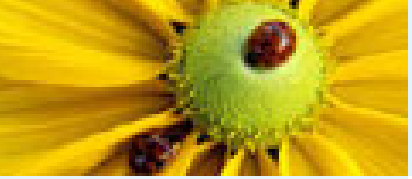
$$\Phi_{\mathcal{Q}}: \quad Xba \neq abY \ \wedge \ Xab \neq baY \ \wedge \ Xaa \neq baY \ \wedge \ Xab \neq aaY$$

$$\Phi_{\mathbb{Z}}: \quad |X| = |Y|$$

$$\Phi_{\Delta+}: \quad |X| \geq 0 \ \wedge \ |X| = |Y|$$

$$\Phi_{\Delta-}: \quad |X| = 3 \ \wedge \ |X| = |Y|$$

$$\Phi_{\Delta}: \quad |X| \neq 1 \ \wedge \ |X| = |Y|$$



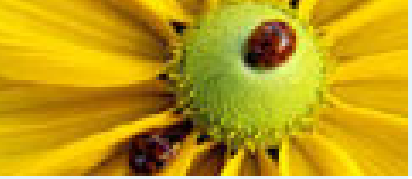
Main Theorem

Given $\Phi_Q \wedge \Phi_{\mathbb{Z}}$.

Let Φ_Δ be an LCC for $\Phi_Q \wedge \Phi_{\mathbb{Z}}$. Then

$$\mathfrak{Q}_{\mathbb{Z}} \models_\exists \Phi_Q \wedge \Phi_{\mathbb{Z}} \quad \Leftrightarrow \quad \mathfrak{Q} \models_\exists \Phi_Q \ \& \ \mathrm{PA} \models_\exists \Phi_\Delta.$$

Decision Problem $\mapsto$ Computation of LCC.

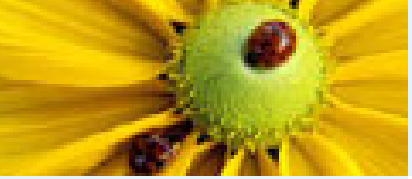


Generic Decision Procedure

Input: $\Phi_Q \wedge \Phi_{\mathbb{Z}}$.

1. Return *FAIL* if $\mathfrak{Q} \not\models_{\exists} \Phi_Q$.
2. For each partition $\Phi_Q^{(i)} \wedge \Phi_{\mathbb{Z}}^{(i)}$ of $\Phi_Q \wedge \Phi_{\mathbb{Z}}$:
   (a) Compute an LCC $\Phi_{\Delta}^{(i)}$ for $\Phi_Q^{(i)} \wedge \Phi_{\mathbb{Z}}^{(i)}$.
   (b) Return *SUCCESS* if $\mathrm{PA} \models_{\exists} \Phi_{\Delta}^{(i)}$.
3. Return *FAIL*.

How to compute LCC?



Problem II

We cannot partition terms into *stratified clusters* and construct a satisfying assignment inductively.

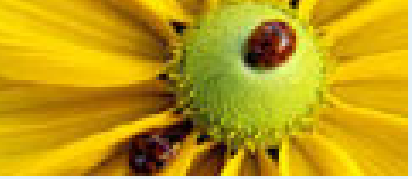
Example: Consider

$$X \neq Y \ \wedge \ aX \neq Yb \ \wedge \ Xa \neq bY$$

Infinitely many assignments of the form

$$X = (ba)^n b, \qquad\qquad Y = a(ba)^n$$

satisfy $X \neq Y$, but neither $aX \neq Yb$ nor $Xa \neq bY$.



Problem II

We cannot partition terms into *stratified clusters* and construct a satisfying assignment inductively.

Example: Consider

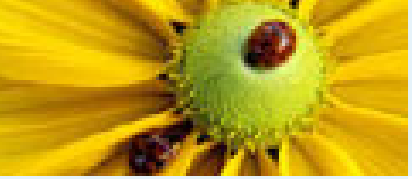
$$X \neq Y \ \wedge \ aX \neq Yb \ \wedge \ Xa \neq bY$$

Infinitely many assignments of the form

$$X = (ba)^n b, \qquad Y = a(ba)^n$$

satisfy $X \neq Y$, but neither $aX \neq Yb$ nor $Xa \neq bY$.

Find a *cut length*!



Cut Length

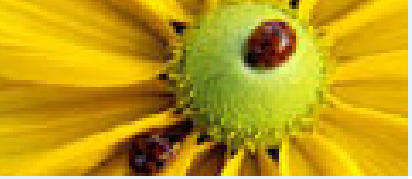
1. Φ_Q can be satisfied by sufficiently long queues.
2. There exists a *cut length* δ such that any solution $(l_i)_n$ with $l_i \geq \delta$ is realizable.
3. But δ is not the smallest $max\{(\mu_i)_n\}$ such that

$$\mathfrak{Q}_{\mathbb{Z}} \models_{\exists} \Phi_Q \ \wedge \ \bigwedge_{i=1}^{n} |X_i| = \mu_i$$

Example: $\{X := \epsilon_Q, Y := \epsilon_Q\}$ is a solution for

$$Xba \neq abY \ \wedge \ Xab \neq baY \ \wedge \ Xaa \neq baY \ \wedge \ Xab \neq aaY$$

while there is no solution σ such that $|\sigma(X)| = |\sigma(Y)| = 1$.



Computation of Cut Length

PRE_Φ : the set of all words α s.t. αX or α is a proper term in Φ_Q.

d_Φ: the shortest strongly primitive word d such that

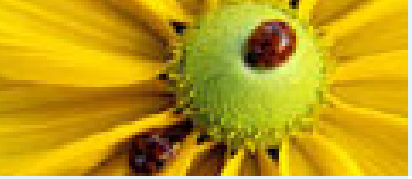
$$(\forall \alpha \in \mathrm{PRE}_\Phi)\ d \notin \mathrm{orb}(\alpha).$$

L_d: the length of d_Φ.

L_c: the smallest number of letters to create a unique identifying word, called a *color*, for each queue variable in Φ_Q.

L_t: $L_c + L_d$.

We claim that $L_t \geq \delta$.



Example

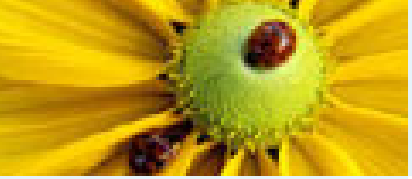
Consider

$$Xba \neq abY \ \wedge \ Xab \neq baY \ \wedge \ Xaa \neq baY \ \wedge \ Xab \neq aaY$$

Then

1. $\mathrm{PRE}_{\Phi} = \{ab, ba, aa\}$.
2. $L_d = 3$; $d_{\Phi} = aab$.
3. $L_c = 1$; Φ_Q includes two queue variables.

So $L_t = L_c + L_d = 4$.

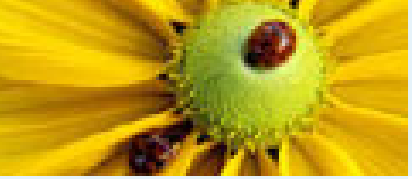


Computation of LCC

Input: $\Phi_Q \wedge \Phi_{\mathbb{Z}}$ in normal form of $\mathfrak{Q}_{\mathbb{Z}}$.

Initially set $\Phi_\Delta = \Phi_{\mathbb{Z}}$. Add to Φ_Δ:

- $|t_1| = |t_2|$, if $t_1 \neq t_2$ or $t_1 = t_2$;
- $|X| + |\alpha| = |\alpha X| = |X\alpha|$, if αX or $X\alpha$ occurs;
- $|X| \equiv k(\bmod |\alpha|)$, if $X \in \mathrm{orb}(\alpha, k)$.
- $|X| = i$ (for some $i < L_t$) or $|X| \geq L_t$ for each X in Φ_Q.



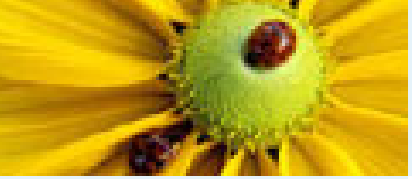
Normal Form in $\mathfrak{Q}_{\mathbb{Z}}$

Φ_Q is in *normal form* in $\mathfrak{Q}_{\mathbb{Z}}$ if

1. Φ_Q is in normal form in Q;
2. Φ_Q is equality complete;
3. if $\alpha X \neq Y \beta$ occurs with either $X \in \mathrm{orb}(\alpha', k)$ or $Y \in \mathrm{orb}(\beta', l)$, then $\alpha \equiv \epsilon_Q$;
4. $\alpha X \neq Y \beta$ does not occur with both $X \in \mathrm{orb}(\alpha', k)$ and $Y \in \mathrm{orb}(\beta', l)$.

Φ_Q is *equality complete* if

$t_1 \neq t_2 \in \Phi_Q$ precisely when $|t_1| = |t_2| \in \Phi_{\mathbb{Z}}$.

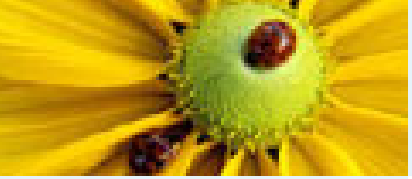


Quantifier Elimination for Queues with Integ

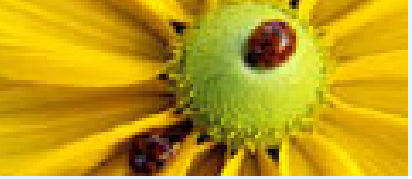
New Normalization. To deal with parameters $\bar{Y}$.

Blockwise Elimination. Remove a block of quantifiers in one step.

$$(\exists x_1, \ldots, \exists x_n)\Phi(x_1, \ldots, x_n, y_1, \ldots, y_m) \quad \mapsto \quad \Phi'(y_1, \ldots, y_m)$$



Conclusion and Future Work

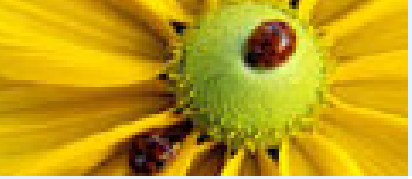


Conclusion

Decision procedures for the combination of queues with integer constraints

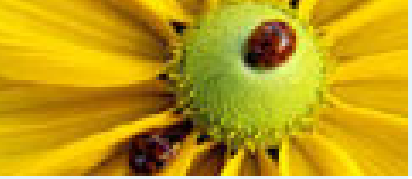
- Express memory safety property.
- Essential for practical program verification.

Exploit algebraic properties of concrete domains.



Future Work (1)

- Implementation and experimentation.
- More expressive languages.
  - Term algebras with subterm relation
  - Queues with subsequence relations, namely, prefix $\preceq_p$, subqueue $\preceq$ and suffix $\preceq_s$



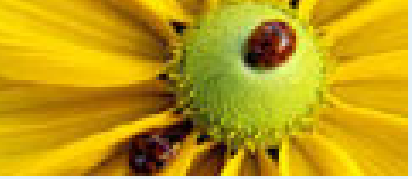
Future Work (1)

- Implementation and experimentation.
- More expressive languages.
  - Term algebras with subterm relation
  - Queues with subsequence relations, namely, prefix $\preceq_p$, subqueue $\preceq$ and suffix $\preceq_s$

    With our decision procedures for

    $$\mathfrak{Q}_{\mathbb{Z}}+\preceq_p+\preceq \quad \textit{and} \quad \mathfrak{Q}_{\mathbb{Z}}+\preceq_s+\preceq,$$

    the next step is $\boxed{\mathfrak{Q}_{\mathbb{Z}}+\preceq_p+\preceq_s}$!



Future Work (2)

$\mathfrak{Q}_{\mathbb{Z}}{+}{\preceq_p}{+}{\preceq_s}$ is a very expressive theory.

1. Equivalent to the theory of concatenation with integers. (Open problem since 80's, Büchi and Senger [BS88])

$$uv^2 = vuv \ \wedge \ |u| < |v|$$

2. Interpret the theory of arrays.

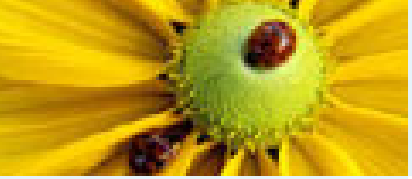
$$q[i] = a \leftrightarrow \exists p\,(pa \preceq_p q \ \wedge \ |pa| = i)$$

3. Interpret Presburger arithmetic with divisibility predicate.

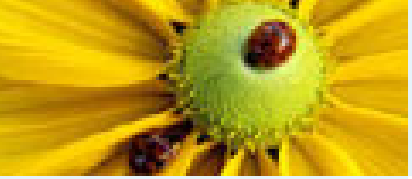
$$x = y + 2 \ \wedge \ y \mid x$$

4. Augmentable to theory of unbounded bit-vectors.

$$u \oplus v = w \wedge uv = ww$$



Thank You!



Normalization in $\mathfrak{Q}$ (1)

Input $\Phi_{\mathcal{Q}} : \mathcal{E} \cup \mathcal{D}$.

1. Reduce literals of the form $\alpha X \beta = \alpha' Y \beta', \alpha X \beta \neq \alpha' Y \beta'$.
   Example:

$$abXcd = abcYdd \mapsto \text{false},$$

$$abXcd \neq abcYd \mapsto Xc \neq cY.$$

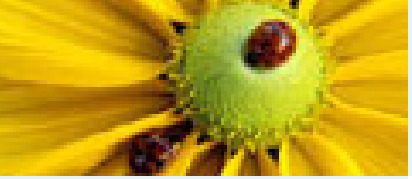
2. Eliminate equalities of the form $\alpha X = Y\beta$ with $X \not\equiv Y$.

   For $|X| < |\beta|$,

$$\alpha X = Y\beta \mapsto X = \beta[|\beta| - |X| + 1..|\beta|] \ \wedge \ Y = \alpha\beta[1..|\beta| - |X|].$$

   For $|X| \geq |\beta|$,

$$\alpha X = Y\beta \mapsto X = X'\beta \ \wedge \ Y = \alpha X'.$$



Normalization in $\mathfrak{Q}$ (2)

3 Eliminate equalities of the form $\alpha X = X\beta$.
If α, β are not conjugate,

$$\alpha X = X\beta \mapsto \text{false}.$$

If α, β are k-conjugate,

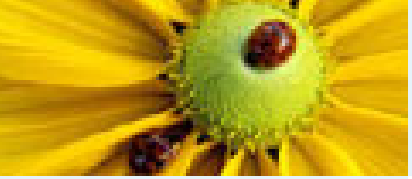
$$\alpha X = X\beta \mapsto X \in \text{orb}(\alpha, k).$$

4 Eliminate disequalities of the form $\alpha X \neq X\beta$.
If α, β are not conjugate,

$$\alpha X \neq X\beta \mapsto \text{true}$$

If α, β are k-conjugate,

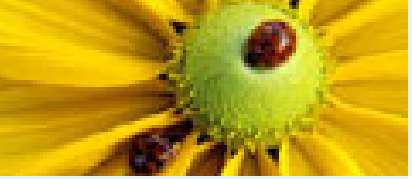
$$\alpha X \neq X\beta \mapsto X \notin \text{orb}(\alpha, k).$$



Normalization in $\mathfrak{Q}_{\mathbb{Z}}$ (1)

1. Normalize Φ_Q as in $\mathfrak{Q}$.
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 $|Y| \geq |\alpha|$.



Normalization in $\mathfrak{Q}_{\mathbb{Z}}$ (2)

3 Consider each subconstraint of the form

$$\alpha X \neq Y\beta \ \wedge \ X \in \mathrm{orb}(\alpha', k) \ \wedge \ Y \in \mathrm{orb}(\beta', l). \qquad (1)$$

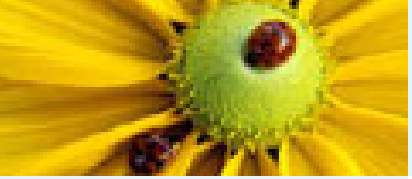
Assuming β is a suffix of X and α is a prefix of Y,

$$\alpha X \neq Y\beta \mapsto X' \in \mathrm{orb}(\alpha', k') \ \wedge \ Y' \in \mathrm{orb}(\beta'', l') \ \wedge \ X' \neq Y',$$

where

$$k' = \Big(k + |\alpha'| - (|\beta| \bmod |\alpha'|)\Big) \bmod |\alpha'|,$$
$$\beta'' = \beta'[(|\alpha| \bmod |\beta'|) + 1..|\beta'|] \circ \beta'[1..(|\alpha| \bmod |\beta'|)],$$
$$l' = \Big(l + |\beta'| - (|\alpha| \bmod |\beta'|)\Big) \bmod |\beta'|.$$

If $\alpha' = \beta''$, then (1) is false.
If $\alpha' \neq \beta''$, then (1) reduces to a finite set of solutions.



Normalization in $\mathfrak{Q}_{\mathbb{Z}}$ (3)

Example 1: Consider

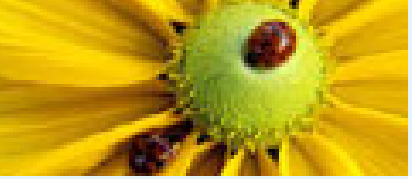
$$abX \neq Yba \ \wedge \ X \in \mathrm{orb}(ab,1) \ \wedge \ Y \in \mathrm{orb}(ab,1).$$

$$abX \neq Yba \mapsto X' \in \mathrm{orb}(ab,1) \ \wedge \ Y' \in \mathrm{orb}(ab,1) \ \wedge \ X' \neq Y',$$
$$\mapsto \mathrm{false}.$$

Example 2: Consider

$$abX \neq Yba \ \wedge \ X \in \mathrm{orb}(abb,1) \ \wedge \ Y \in \mathrm{orb}(ab,0).$$

$$abX \neq Yba \mapsto X' \in \mathrm{orb}(abb,2) \ \wedge \ Y' \in \mathrm{orb}(ab,0) \ \wedge \ X' \neq Y',$$
$$\mapsto X' \neq ab \ \vee \ Y' \neq ab$$
$$\mapsto X \neq abba \ \vee \ Y \neq abab.$$



Normalization in $\mathfrak{Q}_{\mathbb{Z}}$ (4)

4 Consider each subconstraint of the form

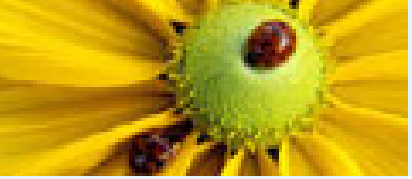
$$\alpha X \neq Y\beta \ \wedge \ X \in \mathrm{orb}(\alpha', k).$$

Guess a word α'' such that $|\alpha''| = |\alpha|$.
If $\alpha'' \equiv \alpha$,

$$\alpha X \neq Y\beta \mapsto Y = \alpha'' Y'$$

Otherwise,

$$\alpha X \neq Y\beta \mapsto Y = \alpha Y' \ \wedge \ X \neq Y'\beta.$$



Normalization in $\mathfrak{Q}_{\mathbb{Z}}$ (5)

5 Consider each subconstraint of the form

$$\alpha X \neq Y\beta \ \wedge \ Y \in \mathrm{orb}(\beta', l).$$

Assuming α is a prefix of Y,

$$\alpha X \neq Y\beta \mapsto Y' \in \mathrm{orb}(\beta'', l') \ \wedge \ X \neq Y'\beta,$$

where

$$\beta'' = \beta'[(|\alpha| \bmod |\beta'|) + 1..|\beta'|] \circ \beta'[1..(|\alpha| \bmod |\beta'|)],$$

$$l' = \Big(l + |\beta'| - (|\alpha| \bmod |\beta'|)\Big) \bmod |\beta'|.$$

[ARR01] Alessandro Armando, Silvio Ranise, and Michaël Rusinowitch. Uniform derivation of decision procedures by superposition. *Lecture Notes in Computer Science*, 2142:513–527, 2001.

[Bjø98] Nikolaj S. Bjørner. *Integrating Decision Procedures for Temporal Verification*. PhD thesis, Computer Science Department, Stanford University, November 1998.

[BLSS01] Michael Benedikt, Leonid Libkin, Thomas Schwentick, and Luc Segoufin. A model-theoretic approach to regular string relations. In *Proc. 16th IEEE Symp. Logic in Comp. Sci.*, pages 431–440. IEEE Computer Society Press, 2001.

[BS88] J. Richard Büchi and Steen Senger. Definability in the existential theory of concatenation and undecidable extensions of this theory. *Zeitschr. f. math. Logik und Grundlagen d. Math.*, 34:337–342, 1988.

[Ghi05] Silvio Ghilardi. Model-theoretic methods in combined constraint satisfiabil-

ity. *Journal of Automated Reasoning*, 33(3-4):221–249, 2005.

[KR03] Felix Klaedtke and Harald Rueß. Monadic second-order logics with cardinalities. In Jos C. M. Baeten, Jan Karel Lenstra, Joachim Parrow, and Gerhard J. Woeginger, editors, *30th International Colloquium on Automata, Languages and Programming, ICALP'2003*, volume 2719 of *Lecture Notes in Computer Science*. Springer-Verlag, 2003.

[NO79] Greg Nelson and Derek C. Oppen. Simplification by cooperating decision procedures. *ACM Trans. Prog. Lang. Sys.*, 1(2):245–257, October 1979.

[RV00] Tatiana Rybina and Andrei Voronkov. A decision procedure for term algebras with queues. In *Proc. 15th IEEE Symp. Logic in Comp. Sci.*, pages 279 – 290, 2000.

[RV03] Tatiana Rybina and Andrei Voronkov. Upper bounds for a theory of queues. In *Proceedings of 30th International Collo-*

quium on Automata, Languages and Programming (ICALP'03), volume 2719 of *LNCS*, pages 714–724. Springer-Verlag, 2003.

[TR03] Cesare Tinelli and Christophe Ringeissen. Unions of non-disjoint theories and combinations of satisfiability procedures. *Theoretical Computer Science*, 290(1):291–353, January 2003.

[Zar02] Calogero G. Zarba. A tableau calculus for combining non-disjoint theories. volume 2381 of *Lecture Notes in Artificial Intelligence*, pages 315–329. Springer, 2002.

[ZSM04a] Ting Zhang, Henny B. Sipma, and Zohar Manna. Decision procedures for recursive data structures with integer constraints. In *the* 2^{nd} *International Joint Conference on Automated Reasoning (IJCAR'04)*, volume 3097 of *LNCS*, pages 152–167. Springer-Verlag, 2004.

[ZSM04b] Ting Zhang, Henny B. Sipma, and Zohar Manna. Term algebras with length

function and bounded quantifier alternation. In *the* 17^{th} *International Conference on Theorem Proving in Higher Order Logics (TPHOLs'04)*, volume 3223 of *LNCS*, pages 321–336. Springer-Verlag, 2004.