A Logic for Parameterized Octagonal Constraints with $\pm\infty$

Deepak Kapur

Department of Computer Science
University of New Mexico
Albuquerque, NM, USA
and
Institute of Software
Chinese Academy of Sciences
Beijing, China

Outline

- Octagonal Formulae, Parameterized Octagonal Formulae, Problems
- Motivation and Background
- Logic
- Concluding Remarks

Parameterized Octagonal Formulae

- ```
  while .... {
    x := −x +1
  }
  return x
  ```

  $(l \leq u \wedge l \leq x \leq u) \implies l \leq -x + 1 \leq u$: yes,
  $l + u = 1 \wedge l \leq u$.

- ```
  while ... {
    x := x + 1
  }
  return x
  ```

  $(l \leq u \wedge (l \leq x \leq u)) \implies (l \leq x + 1 \leq u)$: yes, provided
  $u = \infty$.

Motivation and Background

- Quantifier Elimination Approach for Generating (Loop) Invariants (Kapur 2003-2004) - Review with examples.
- Scaling QE based methods: Geometric and Local Heuristics
  - Octagonal Invariants - simple Convex Linear Constraints
  - $O(k * n^2)$ algorithm to generate the strongest octagonal invariants, where k is the number of program paths and n is the number of program variables.
    - jointly with Zhihai Zhang and Hengjun Zhao
  - Quantifier elimination to eliminate program variables from verification conditions (parameterized formulae).
  - Solve for parametric constraints and get a satisfying assignment to generate invariants.

Invariants: Integer Square Root

Example

```
x := 1, y := 1, z := 0;
while (x <= N) {
   x := x + y + 2;
   y := y + 2;
   z := z + 1
}
return z
```

Generating Loop Invariant: Approach

- Guess/fix the shape of invariants of interest at various program locations with some parameters which need to be determined.
- Here is an illustration of generation of nonlinear invariants.
  $\mathbf{I}: \quad A\ x^2 + B\ y^2 + C\ z^2 + D\ xy + E\ xz + F\ yz + G\ x + H\ y + J\ z + K = 0.$
- Generate verification conditions using the hypothesized invariants from the code.
  - **VC1:** At first possible entry of the loop (from initialization):

    $$A + B + D + G + H + K = 0.$$
  - **VC2:** For every iteration of the loop body:

    $$(I(x, y, z) \wedge x \leq N) \implies I(x + y + 2, y + 2, z + 1).$$

- Using quantifier elimination, find constraints on parameters $A, B, C, D, E, F, G, H, J, K$ which ensure that the verification conditions are valid for all possible program variables.

Quantifier Elimination from Verification Conditions

Considering **VC2:**

- $(\ A\,x^2+B\,y^2+C\,z^2+D\,xy+E\,xz+F\,yz+G\,x+H\,y+J\,z+K=0) \implies (\ A\,(x+y+2)^2+B\,(y+2)^2+C\,(z+1)^2+D\,(x+y+2)(y+2)+E\,(x+y+2)(z+1)+F\,(y+2)(z+1)+G\,(x+y+2)+H\,(y+2)+J\,(z+1)+K=0)$
- Expanding the conclusion gives:
  $Ax^2+(A+B+D)y^2+cz^2+(D+2A)xy+Exz+(E+F)yz+(G+4A+2D+E)x+(H+4A+4B+4D+E+F+G)y+(J+2C+2E+2F)z+(4A+4B+C+4D+2E+2F+2G+2H+J+K)=0$
- Simplifying using the hypothesis gives:
  $(A+D)y^2+2Axy+Eyz+(4A+2D+E)x+(4A+4B+4D+E+F+G)y+(2C+2E+2F)z+(4A+4B+C+4D+2E+2F+2G+2H+J)=0$
- Since this should be 0 for all values of x, y, z: we have:
  $A+D=0; A=0; E=0$ which implies $D=0$; using these gives:
  $2C+2F=0$ which implies $C=-F$; using all these:
  $G=-4B-F$, $H=-G-K-B$ and $J=-2B-F+2K$.

Generating the Strongest Invariant

- Constraints on parameters are:

  $$C = -F, \quad J = -2B - F + 2K, \quad G = -4B - F, \quad H = 3B + F - K.$$

- Every value of parameters satisfying the above constraints leads to an invariant (including the trivial invariant **true** when all parameter values are 0).
- 7 parameters and 4 equations, so 3 independent parameters, say B, F, K. Making every independent parameter 1 separately with other independent parameters being 0, derive values of dependent parameters.
- $K = 1, H = -1, J = 2$ gives $-y + 2z + 1 = 0$.
- $F = 1, C = -1, J = -1, G = -1, H = 1$ gives $-z^2 + yz - x + y - z - 0$.
- $B = 1, J = -2, G = -4, H = 3$ gives $y^2 - 4x + 3y - 2z = 0$.
- The most general invariant describing all invariants of the above form is a conjunction of:

  $$y = 2z + 1; \quad z^2 - yz + z + x - y = 0 \quad y^2 - 2z - 4x + 3y = 0,$$

  from which $x = (z + 1)^2$ follows.

Method for Automatically Generating Invariants by Quantifier Elimination

- Hypothesize assertions, which are parametrized formulas, at various points in a program.
  - Typically entry of every loop and entry and exit of every procedure suffice.
  - Nested loops and procedure/function calls can be handled.
- Generate verification conditions for every path in the program (a path from an assertion to another assertion including itself).
  - Depending upon the logical language chosen to write invariants, approximations of assignments and test conditions may be necessary.
- Find a formula expressed in terms of parameters eliminating all program variables (using quantifier elimination).

Quality of Invariants: Soundness and Completeness

- Every assignment of parameter values which make the formula true, gives an inductive invariant.
  - If no parameter values can be found, then invariants of hypothesized forms may not exist. Invariants can be guaranteed not to exist if no approximations are made, while generating verification conditions.
- If all assignments making the formula true can be finitely described, invariants generated may be the strongest of the hypothesized form. Invariants generated are guaranteed to be the strongest if no approximations are made, while generating verification conditions.

Domains Admitting Quantifier-Elimination

- Presburger Arithmetic and Generalized Presburger Arithmetic (in which coefficients are polynomials in parameters) (Weispfenning, 1990;1997).
- Polynomials over an algebraic closed field of characteristic 0: Parametric Gröbner Basis Algorithm (Kapur, 1994), Comprehensive Gröbner Basis System (Weispfenning, 1992).
- Quantifier Elimination Techniques for Real Closed Fields (REDLOG, QEPCAD)
- Combination of Theories–Presburger Arithmetic with Theory of Equality over Uninterpreted Symbols (Shostak, 1979; Nelson, 1981), and with Boolean Algebra (Kuncak, 2007).
  - Uninterpreted function symbols can be used for modeling array manipulations and other memory operations.
- Reduction Approach to Decision Procedures for Theories over Abstract Data Structures, including Finite Lists, Finite Sets, Finite Arrays, Finite Multisets (Kapur and Zarba, 2005).

How to Scale this Approach

- Quantifier Elimination Methods typically do not scale up due to high complexity even in this restricted case of $\exists\forall$.
  - Even for Presburger arithmetic, complexity is doubly exponential in the number of quantifier alternations and triply exponential in the number of quantified variables
  - Output is huge and difficult to decipher.
  - In practice, they often do not work (i.e., run out of memory or hang).
- Linear constraint solving on rationals and reals (polyhedral domain), while of polynomial complexity, has been found in practice to be inefficient and slow, especially when used repeatedly as in abstract interpretation approach [Miné]

Making QE based Method Practical

- Identify (atomic) formulas and program abstractions resulting in verification conditions with good shape and structure.
- Develop QE heuristics which exploit *local* structure of formulas (e.g. two variables at a time) and geometry of state space defined by formulas.
- Among many possibilities in a result after QE, identify those most likely to be useful.
- **Octagonal formulas** : $l \leq \pm x \pm y \leq h$, a highly restricted subset of linear constraints (at most two variables with coefficients from $\{-1, 0, 1\}$).
  - This fragment is the most expressive fragment of linear arithmetic over the integers with a polynomial time decision procedure.
  - Extending constraints to contain three variables (with just unit coefficients) per inequality makes satisfiability check over the integers NP-complete.
  - Two variable inequalities with non-unit coefficients over the integers makes the satisfiability check NP-complete.

Octagonal Formulas

- Octagonal formulas over two variables have a fixed shape. Its parameterization can be given using 8 parameters.
- Given n variables, the most general formula (after simplification) is of the following form
  $\bigwedge_{i,j}(\ Octa_{i,j} : \ \ a_{i,j} \leq x_i - x_j \leq b_{i,j}, \ \ c_{i,j} \leq x_i + x_j \leq d_{i,j}$
  $\quad e_i \leq x_i \leq f_i \quad g_j \leq x_j \leq h_j)$
  for every pair of variables x_i, x_j, where $a_{i,j}$, $b_{i,j}$, $c_{i,j}$, $d_{i,j}$, e_i, f_i, g_j, h_j are parameters.
- Class of programs that can be analyzed are very restricted. Still using octagonal constraints (and other heuristics), ASTREE is able to successfully analyze hundreds of thousands of lines of code of numerical software for array bound check, memory faults, and related bugs.
  - Algorithms used in ASTREE are of $O(n^3)$ complexity (sometimes, $O(n^4)$), where n is the number of variables (Miné, 2003).
- **Goal:** Performance of QE heuristic should be at least as good.

A Simple Example

Example

```
 x := 4; y := 6;
while (x + y >= 0) do
 if (y >= 6) then { x := -x;  y := y - 1 }
     else { x := x - 1; y := -y }
endwhile
```

VC0: $I(4, 6)$

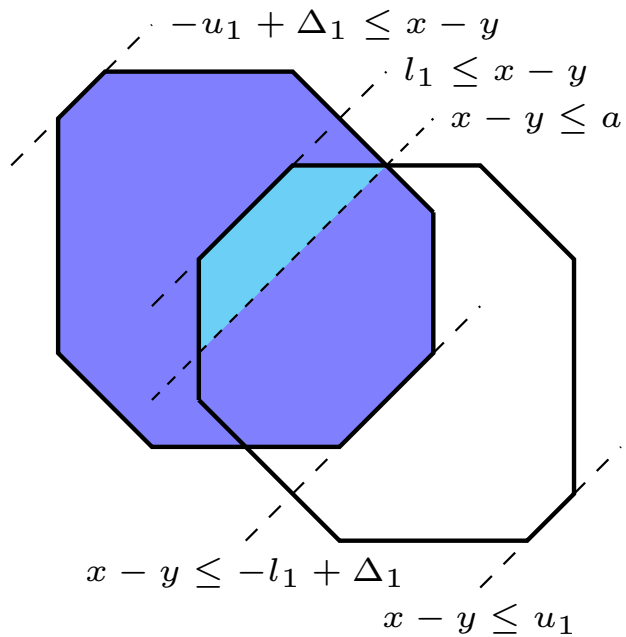
VC1: $(I(x, y) \wedge (x + y) \geq 0 \wedge y \geq 6) \implies I(-x, y - 1).$

VC2: $(I(x, y) \wedge (x + y) \geq 0 \wedge y < 6) \implies I(x - 1, -y).$

Table 1: Assignments with signs of variables reversed

$$x := -x + A$$
$$y := -y + B$$

$$\Delta_1 = A - B, \quad \Delta_2 = A + B.$$



constraint	present	absent
$x - y \le a$	$a \le \Delta_1 - l_1$	$u_1 \le \Delta_1 - l_1$
$x - y \ge b$	$\Delta_1 - u_1 \le b$	$\Delta_1 - u_1 \le l_1$
$x + y \le c$	$c \le \Delta_2 - l_2$	$u_2 \le \Delta_2 - l_2$
$x + y \ge d$	$\Delta_2 - u_2 \le d$	$\Delta_2 - u_2 \le l_2$
$x \le e$	$e \le A - l_3$	$u_3 \le A - l_3$
$x \ge f$	$A - u_3 \le f$	$A - u_3 \le l_3$
$y \le g$	$g \le B - l_4$	$u_4 \le B - l_4$
$y \ge h$	$B - u_4 \le h$	$B - u_4 \le l_4$

A Simple Example

Example

```
 x := 4; y := 6;
while (x + y >= 0) do
 if (y >= 6) then { x := -x;  y := y - 1 }
     else { x := x - 1; y := -y }
endwhile
```

VC0: $I(4, 6)$

VC1: $(I(x, y) \wedge (x + y) \geq 0 \wedge y \geq 6) \implies I(-x, y - 1).$

VC2: $(I(x, y) \wedge (x + y) \geq 0 \wedge y < 6) \implies I(x - 1, -y).$

Generating Constraints on Parameters

- **VC0:**
  $l_1 \leq -2 \leq u_1 \wedge l_2 \leq 10 \leq u_2 \wedge l_3 \leq 4 \leq u_3 \wedge l_4 \leq 6 \leq u_4$.
- **VC1:** $x - y$: $-u_2 - 1 \leq l_1 \wedge u_1 \leq -l_2 - 1$.
  $x + y$: $-u_1 + 1 \leq 0 \wedge u_2 \leq -l_1 + 1$.
  x: $l_3 + u_3 = 0$.
  y: $l_4 \leq 5$.
- **VC2:** $x - y$: $-u_2 - 1 \leq -u_1 \wedge 10 \leq -l_2 - 1$.
  $x + y$: $l_1 + 1 \leq 0 \wedge u_2 \leq u_1 + 1$.
  x: $l_3 \leq -6$.
  y: $-u_4 \leq l_4 \wedge 5 \leq -l_4$.
- Values of $l_1, u_1, l_2, u_2, l_3, u_3, l_4, u_4$ which satisfy the above parametric constraints give an invariant.
- Make l_i's as large as possible and u_i's as small as possible:
  $l_1 = -10, u_1 = 9, l_2 = -11, u_2 = 10,$
  $l_3 = -6, u_3 = 6, l_4 = -5, u_4 = 6.$
- The corresponding invariant is:
  $-10 \leq x - y \leq 9 \wedge -11 \leq x + y \leq 10$
  $\wedge -6 \leq x \leq 6 \wedge -5 \leq y \leq 6.$

Generating Invariants using Table Look-ups

- Parameter constraints corresponding to a specific program path are read from the corresponding entries in tables.
- Accumulate all such constraints on parameter values. They are also **octagonal**.
- Every parameter value that satisfies the parameter constraints leads to an invariant.
- Maximum values of lower bounds and minimal values of upper bounds satisfying the parameter constraints gives the strongest invariants. Maximum and minimum values can be computed using Floyd-Warshall's algorithm.

Complexity and Parallelization

- Overall Complexity: $O(k * n^2)$:
  - For every pair of program variables, parametric constraint generation is constant time: 8 constraints, so 8 entries.
  - Parametric constraints are decomposed based on parameters appearing in them: there are $O(n^2)$ such constraints on disjoint blocks of parameters of size ≤ 4.
- Program paths can be analyzed in parallel. Parametric constraints can be processed in parallel.

Technical Problems to be addressed

- Given a set of parametric octagonal constraints, check whether they are satisfiable? Parameters could take the value $\pm\infty$ along with integer values.
- If satisfiable, find a (nontrivial) satisfying assignment to generate a nontrivial invariant.
- If satisfiable, find the largest possible value a lower bound can take and the smallest value an upper bound can take to get the tightest bound on octagonal expressions, thus generating the strongest invariant.
- Given an octagonal formula with parameters and program variables in which program variables are universally quantified, generate an equivalent (or its approximation) quantifier-free formula over the parameters.

A Calculus for Parameterized Octagonal Constraints

Language: $-\infty, \infty, 0, 1, -1, 2, -2, \cdots, -, +, \leq$
There are two types of variables: program variables x, y which range over $\mathbb{Z}$, and parameters l, u which range over $\mathbb{Z}_{\pm\infty}$. We will write them as $x, y : \mathbb{Z}, l, u : \mathbb{Z}_{\pm\infty}$.

Three kinds of atomic formulae:

1. PA formulae: $c \leq e$, where e is an octagonal expression in program variables and $c \in \mathbb{Z}$; expressions $x, x - y, x + y, -x - y$ are called **octagonal** expressions; 0 is the trivial octagonal expression. e is assumed to be in its simplified form (after canceling and collecting identical terms).
2. $PA_{\mathbb{Z}_{\pm\infty}}$ formulae: $a + p \leq 0$, where p is a linear polynomial in parameters but without a constant and $a \in \mathbb{Z}_{\pm\infty}$. Cancellation of terms in p is not performed.
3. Mixed formulae: $a + p \leq e$, $e \neq 0, p \neq 0$.

Properties of $\pm\infty$

$+, -, \leq$ have the standard semantics on integers. $p = q$ stands for $p \leq q \wedge q \leq p$. $p - q$ stands for $p + -(q)$.

$$-(-\infty) = \infty, -(\infty) = -\infty,$$

$$c + -\infty = -\infty, c + \infty = \infty, c \in \mathbb{Z}$$

$$x : \mathbb{Z} \neq \infty, x : \mathbb{Z} \neq -\infty, x : \mathbb{Z} + \infty = \infty, x : \mathbb{Z} + -\infty = -\infty,$$

$$a \leq \infty, -\infty \leq a, a \in \mathbb{Z}_{\pm\infty}$$

$$l : \mathbb{Z}_{\pm\infty} \leq \infty, -\infty \leq l : \mathbb{Z}_{\pm\infty}$$

Properties that do not hold

$$\infty - \infty = 0$$

$$(p + e_1 = p + e_2) \implies (e_1 = e_2)$$

$$(c_1 + l + p_1 \leq c_2 + l + p_2) \implies (c_1 + p_1 \leq c_2 + p_2)$$

Transitivity property of $\leq$ (as well as $=$) is not valid.

We can thus solve:
$l + 3 = l + 4 \implies (l = \infty \vee l = -\infty)$ and
$l + 4 \leq l + 3 \implies (l = \infty \vee l = -\infty)$

Other Axioms

Properties of $+, \leq, -$ on integer constants.

Strengthening Axioms:

$$(a + p \leq e \wedge b + p \leq e) \equiv a + p \leq e, \quad \text{where} \quad b \leq a,$$

Elimination/Resolution/Propagation Axioms:

$$(a_1 + p_1 \leq e_1 \wedge a_2 + p_2 \leq e_2) \implies (a_1 + a_2) + (p_1 + p_2) \leq f, \quad \text{where}$$

f is a normal form of $e_1 + e_2$ and is either an octagonal expression in program variables or $f = 2x \vee f = -2x$ for some program variable x; if $f = 2x \vee f = -2x$, then congruence modulo 2 relation is used to eliminate the coefficient 2 from $2x, -2x$.

Elimination/Resolution/Propagation Axioms:

$$(a_1 + p_1 \leq e_1 \wedge a_2 + p_2 \leq e_2) \implies (a_1 + a_2) + (p_1 + p_2) \leq f,$$

replaces

$$(a_1 + p_1 \leq x - y \wedge a_2 + p_2 \leq y - z) \implies (a_1 + a_2) + (p_1 + p_2) \leq x - z,$$

$$(a \leq u + v \wedge u + v \leq b) \implies$$
$$((a \leq b) \vee (u = \infty \wedge v = -\infty) \vee (u = -\infty \wedge v = \infty)),$$

$$(a_1 + p_1 \leq x \wedge a_2 + p_2 \leq y) \implies (a_1 + a_2) + (p_1 + p_2) \leq x + y.$$
$$(a_1 + p_1 \leq x \wedge a_2 + p_2 \leq -y) \implies (a_1 + a_2) + (p_1 + p_2) \leq x - y.$$

For every pair of variables and parameters, we have:

$$(a_1 + p_1 \leq x + y \wedge a_2 + p_2 \leq x - y) \Rightarrow \lceil \frac{a_1 + a_2 + p_1 + p_2}{2} \rceil \leq x.$$

$$(a_1 + p_1 \leq -x + y \wedge a_2 + p_2 \leq -x - y) \Rightarrow \lceil \frac{a_1 + a_2 + p_1 + p_2}{2} \rceil \leq -x.$$

Elimination of Program Variables

$$(a_1+p_1 \leq x-y \wedge a_2+p_2 \leq y-z) \implies ((a_1+a_2)+(p_1+p_2) \leq x-z).$$

$$(a_1+p_1 \leq x-y \wedge a_2+p_2 \leq -x+y) \implies (a_1+a_2)+(p_1+p_2) \leq 0.$$

$$(a_1+p_1 \leq x-y \wedge a_2+p_2 \leq x+y) \implies (a_1+a_2)+(p_1+p_2) \leq 2x.$$

Normal form of a parametric expression

A normal form of an arithmetic expression: $c_0 + \Sigma_{i=1}^{k} c_i x_i$ or $\Sigma_{i=1}^{k} c_i x_i$ where c_0, c_i are non-zero integer constants.
For octagonal expressions, we only have
$0, x + y, x - y, -x + y, -x - y$.

A normal form of a parametric expression:
$a_0 + \Sigma_{i=1}^{k} c_i l_i + \Sigma_{i=k+1}^{m} (l_i - l_i)$, or $\Sigma_{i=1}^{k} c_i l_i + \Sigma_{i=k+1}^{m} (l_i - l_i)$,
where $l_i' s$ are parameter variables, a_0 is an nonzero integer constant, ∞ or $-\infty$ or $(\infty - \infty)$; c_i is a nonzero integer constant.

e.g. $3 + l - u$, $\infty + l - u$, $-\infty + l - u$, $\infty - \infty + l + u$,
$1 + (l - l) + (u - u)$.

Normal form equivalences

$$(l = \infty \vee l = -\infty) \equiv (\neg(l - l = 0)) \equiv ((1 \leq l - l) \vee (l - l \leq -1))$$

$$\neg(p_1 \leq p_2) \equiv (p_2 + 1 \leq p_1 \vee p_1 = \infty \vee p_2 = -\infty)$$

$$p_1 \leq p_2 \equiv (p_1 - p_2 \leq 0 \vee p_2 = \infty \vee p_1 = -\infty)$$

$$(a_0 + \Sigma_{i=1}^{k} c_i l_i + \Sigma_{i=k+1}^{m} (l_i - l_i) = \infty) \equiv$$

$$(a_0 = \infty \vee_{i=1}^{k} (c_i > 0 \wedge l_i = \infty) \vee (c_i < 0 \wedge l_i = -\infty))$$

$$(a_0 + \Sigma_{i=1}^{k} c_i l_i + \Sigma_{i=k+1}^{m} (l_i - l_i) = -\infty) \equiv$$

$$(a_0 = -\infty \vee_{i=1}^{k} ((c_i > 0 \wedge l_i = -\infty) \vee (c_i < 0 \wedge l_i = \infty)))$$

$$(a_0 + \Sigma_{i=1}^{k} c_i l_i + \Sigma_{i=k+1}^{m} (l_i - l_i) \leq 0) \equiv$$

$$(a_0 + \Sigma_{i=1}^{k} c_i l_i \leq 0) \vee a_0 = -\infty \vee_{i=1}^{k} ((c_i > 0 \wedge l_i = -\infty) \vee (c_i < 0 \wedge l_i = \infty))$$

$$\vee_{i=k+1}^{m} (l_i = -\infty \vee l_i = \infty))$$

Satisfiable Formulae which are not satisfiable in PA

- Using the above properties, it can be shown that

$$4 \leq l+u \leq 3 \implies ((l = \infty \wedge u = -\infty) \vee (l = -\infty \wedge u = \infty))$$

- Equalities

$$(l+u = 4 \wedge l+u = 3) \equiv (l+u \leq 4 \wedge -l-u \leq -4 \wedge l+u \leq 3 \wedge -l-u \leq -$$

$$(l + u \leq 3 \wedge -l - u \leq -4) \equiv 4 \leq l + u \leq 3$$

is satisfiable with the assignment $l = \infty \wedge u = -\infty$ as well as with the assignment $l = -\infty \wedge u = \infty$,

Equivalences

Using $a_0 + \Sigma_{i=1}^{k} c_i l_i + \Sigma_{i=k+1}^{m} (l_i - l_i) \leq 0) \equiv$

$$(a_0 + \Sigma_{i=1}^{k} c_i l_i \leq 0) \vee a_0 = -\infty \vee_{i=1}^{k} ((c_i > 0 \wedge l_i = -\infty) \vee (c_i < 0 \wedge l_i = \infty))$$

$$\vee_{i=k+1}^{m} (l_i = -\infty \vee l_i = \infty))$$

$$l + u \leq 3 \equiv (-3 + l + u \leq 0 \vee l = -\infty \vee u = -\infty)$$

$$4 \leq l + u \equiv (4 - l - u \leq 0 \vee l = \infty \vee u = \infty)$$

$$(l + u \leq 3 \wedge 4 \leq l + u) \equiv ((-3 + l + u \leq 0 \vee l = -\infty \vee u = -\infty) \wedge (4 - l - u \leq 0$$

$$\equiv ((l = -\infty \wedge u = \infty) \vee (l = \infty \wedge u = -\infty))$$

A Satisfiability Algorithm

Given a parametric formula that is a conjunction of atomic parametric formula, determining whether it is satisfiable or not.

- Use the strengthening axiom schema to refine lower bounds and upper bounds on octagonal expressions.
- Declare unsatisfiability if there is a parameter ranging over an empty interval, i.e. $a \leq l \leq b \wedge b < a$.
- Use the resolution/elimination axiom schema to propagate bounds on related octagonal expressions (much like Floyd Warshall algorithm or DBM matrix algorithm) ($O(n^3)$).
- A cycle with a negative weight in a graph does not imply constraints are unsatisfiable. Have to keep track of parameters used in the application of the resolution axiom schema while propagating bounds in Floyd Warshall algorithm.

Propagation of bounds on parametric formulae

$$(a_1+l-u \leq 0 \wedge a_2+u-v \leq 0) \implies ((a_1+a_2)+(l-v)+(u-u) \leq 0).$$

$$(a_1+l-u \leq 0 \wedge a_2-l+u \leq 0) \implies ((a_1+a_2)+(l-l)+(u-u) \leq 0).$$

$$(a_1+l-u \leq 0 \wedge a_2+l+u \leq 0) \implies (\lceil \frac{(a_1+a_2)}{2} \rceil+l+(u-u) \leq 0).$$

Satisfiability Check Algorithm Contd.

During the application of the previous step, either unsatisfiability is detected or some parameters can be assigned $\pm\infty$ based on the property that

$$(a \leq l+u \leq b \wedge b < a) \implies ((l = -\infty \wedge u = \infty) \vee (l = \infty \wedge u = -\infty))$$

or a more general property

$$(a_0 + \Sigma_{i=1}^{k} c_i l_i + \Sigma_{i=k+1}^{m} (l_i - l_i) \leq 0) \equiv$$

$$(a_0 + \Sigma_{i=1}^{k} c_i l_i \leq 0) \vee a_0 = -\infty \vee_{i=1}^{k} ((c_i > 0 \wedge l_i = -\infty) \vee (c_i < 0 \wedge l_i = \infty))$$
$$\vee_{i=k+1}^{m} (l_i = -\infty \vee l_i = \infty)).$$

To avoid splitting (many possibility), it is useful the context to eliminate some of the cases.
Once values are substituted for parameters, propagate them and simplify formulae using properties of $\pm\infty$.

Extensions

- Can logic be simplified? Can program variables and parameters be treated uniformly more elegantly, i.e., single sort but then may have to resort to unary predicates or $x \neq \infty \wedge x \neq -\infty$?
- Simpler proofs of structural properties
- Finding satisfying assignments efficiently
- Doing quantifier elimination efficiently
- Computing satisfying assignments which minimize the interval $u_i - l_i$ for various octagonal expressions
- Extension to full first-order Presburger arithmetic with $\pm\infty$
- Extension of Presburger arithmetic implementation to handle $\pm\infty$

Concluding Remarks

- Quantifier elimination techniques are general as well as flexible for automatically generating program invariants.
- Scaling is key. Works quite well for octagonal invariants: $O(n^2)$
- Quantifier elimination is done offline only once since allowable assignments are limited. Invariants are generated using table look-ups.