Solving Linear Constraints over Real

German Vitaliy
Jr. Researcher
Glushkov Institute of Cybernetic NAS Ukraine
Email: novaua@ukr.net

Talk outline

- Introduction
- Fourier-Motzkin elimination
- The main algorithm scheme
- Equalities elimination
- Redundancies elimination, simplification
- An example
- Conclusions

Application areas

- Hardware and software verification
- Program analysis
- Compiler optimization
- Planning, scheduling

Existing techniques

- Fourier's methods
  - Omega Test
- Special cases
  - Shostk's loop residue algorithm [4]
  - Bounding boxes, Bounding differences, Octagons
    - Nelson-Oppen Decision Procedure[5]
- Modified Simplex Methods [6]
- The algorithm of Motzkin-Chernikova-Le Verge (the PPL[3] library)

Inputs

- Formula type:
  - Linear equality, inequality over Real (Rational)
- The input formula consists of
  - Boolean connectivity: Or($\vee$), And ($\wedge$), Not($\sim$,$\neg$)
  - Quantifies: Forall($\forall$), Exists($\exists$)
  - Operators: plus (+), minus(-), multiplication on number ()
  - Predicates: <, <=, >=, >, =

Projection of polytope

- Suppose we have a polytope

$$S = \{x \in R^n \mid Ax \le b\}$$

- We would like to construct the projection onto

$$\{x \in R^n \mid x_1 = 0\}$$

- Call this projection P(S)

Projection

- We would like to find inequalities that define the projection P(S)

$$P(S) = \{x_2 \mid \text{there exists } x_1 \text{ such that } \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \in S\}$$

Some other way to say this:

- We would like to find *valid inequalities* do not depend on x_1
- We would like to perform *quantifier elimination* to remove *there exists* and find a *basic semi algebraic,* i.e. defined by conjunction of polynomial inequalities, representation of *P(S)*

Fourier-Motzkin elimination

- **This procedure was invented by Fourier (1826) and rediscovered by Dines (1918) and Motzkin (1936)**
- **Similar to Gaussian elimination (1800)**

Fourier-Motzkin elimination

- Let the source system $Ax \leq b$ where

$$A = \begin{bmatrix} a_1 \\ \dots \\ a_m \end{bmatrix} \qquad b = \begin{bmatrix} b_1 \\ \dots \\ b_m \end{bmatrix}$$

- We can generate equalities of the form

$$(\lambda_1 a_1 + \dots + \lambda_m a_m)x \leq \lambda_1 b_1 + \dots + \lambda_m b_m$$

- The idea is to combine pairs of inequalities that cancel x_1. Since $\lambda_i \geq 0$ member of each pair need opposite signed coefficients of x_1

Fourier-Motzkin theorem

- Take all pairs of inequalities with opposite coefficients of x_1, and for each generate a new valid inequality that eliminates x_1
- Also take all inequalities from the original set which do not depend on x_1

This collection of inequalities defines projection of S onto $x_1 = 0$

Algorithm complexity

- **Eliminating an existential over *n* constraints we may introduce $n^2/4$ new constraints**
- **With k quantifiers to eliminate, we might end with**

$$n^{2^k} / 4^k$$

The Solver main algorithm scheme

1. Normalize the formula tree
2. repeat {
   - Convert to DNF, handle ***disequalities***
   - Eliminate bounded variables from ***equalities***
   - Eliminate bounded variables from ***inequalities***

} until the formula has bounded variables

3. Simplify result
4. End.

Note. The 'bomb' marks operations which has exponential worst case complexity

Goals

- Handle full theory over real
- To be efficient enough for handling the real-life verification problems

But how to do that?

  - Use efficient redundant elimination techniques
  - Find case when we can avoid generics but apply faster special algorithms

Tree transformation rules

$$\forall v : f \rightarrow \neg \exists v : \neg f$$

$$\exists v : (f_1 \vee \ldots \vee f_n) \rightarrow \exists v : f_1 \vee \ldots \vee \exists v : f_n$$

$$\neg (f_1 \vee \ldots \vee f_n) \rightarrow \neg f_1 \wedge \ldots \wedge \neg f_n$$

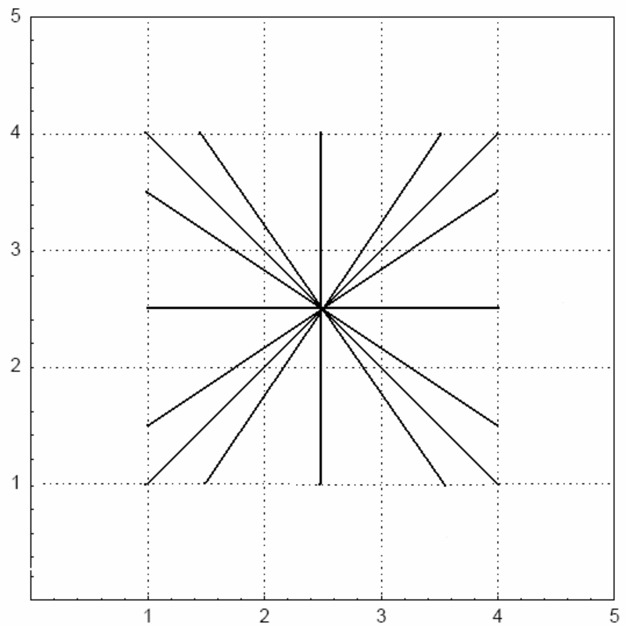
$$\neg \neg f \rightarrow f$$

Equalities elimination

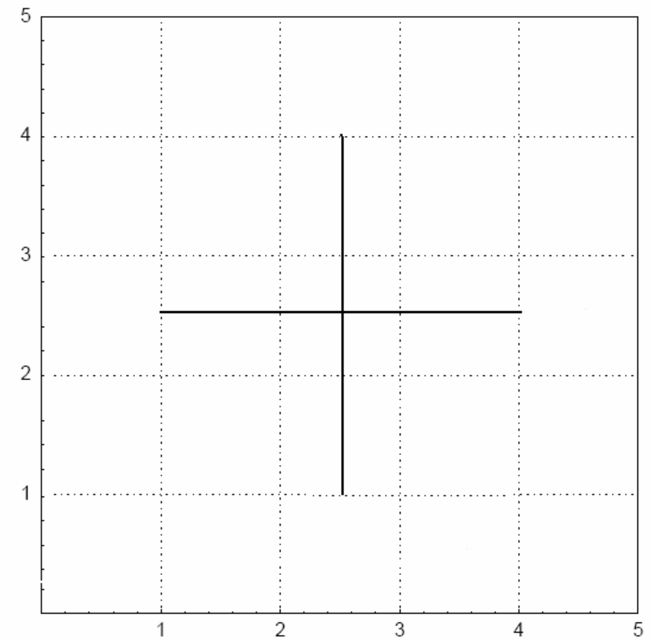
- If input system has both equalities and inequalities equalities are eliminated first
- We use Gaussian elimination procedure for the system of equalities and inequalities constraints[7]

Equalities Simplification

- In practice systems of equalities may be highly redundant



Redundant system



Simplified system

Elimination of the redundant equalities

- Redundant elimination similar to solving
- Available efficient solvers in different libraries for system were number of equalities equal to number of variables… But how to solve the redundant systems using the solvers?

Redundant equalities elimination algorithm

Let M - number of equalities, N – number of variables

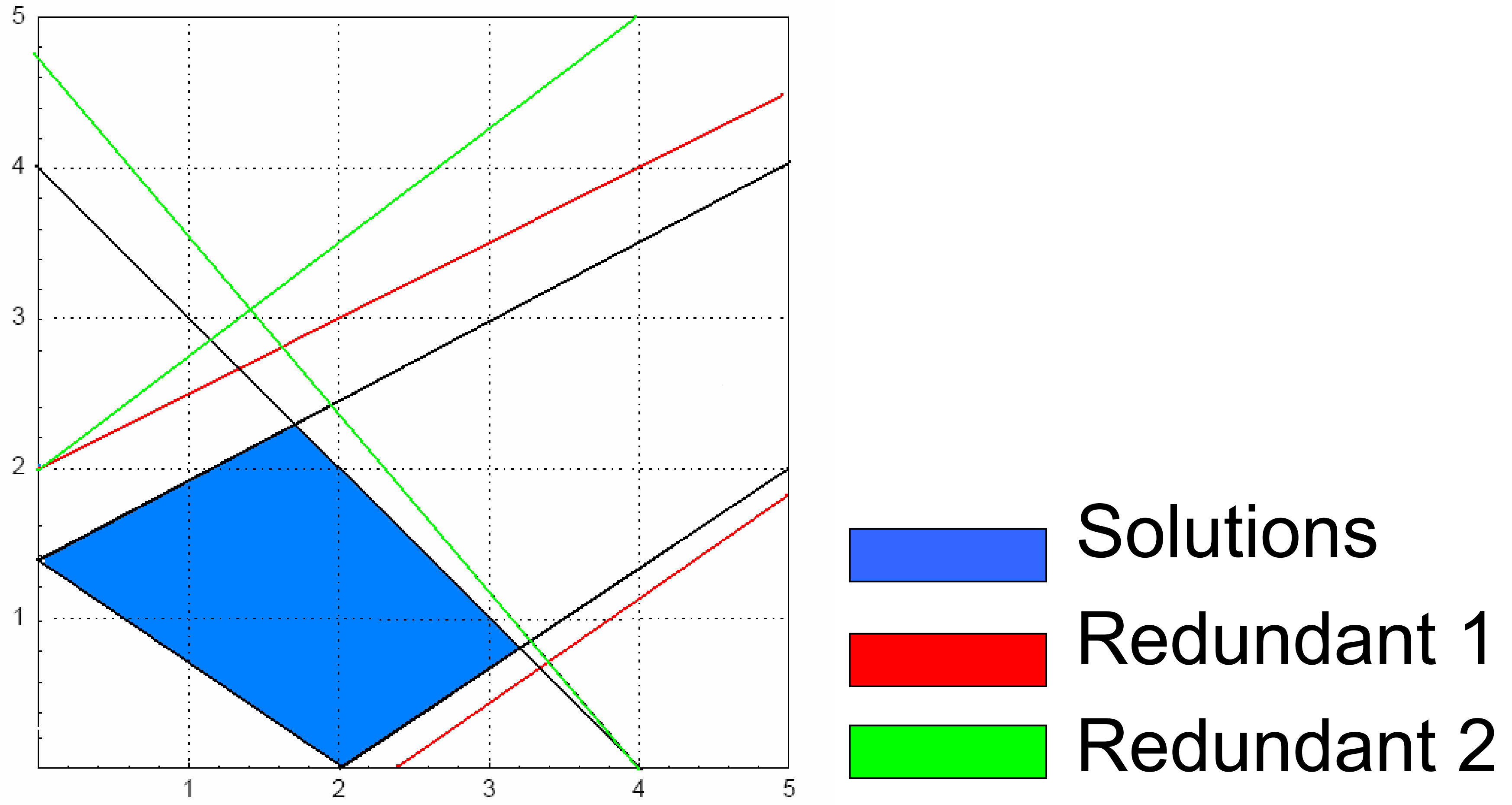
If M > N

1. Normalize by removing of the trivially same equalities
2. Solve first N equalities
3. If result found propagate it to the rest of equalities
4. If the rest is satisfiable then replace all equalities by the founded solution

If M<N the system cannot be solved and simplified.

What if there are several separate systems which depend on the different variables?

The Redundant Facets



Elimination of the Redundant 1 (parallel) facets

Let A, -A positive and negative system of inequalities, equalities respectively, which all are parallel between each other

1. Normalize the system
2. Find the upper bound inequality $A_i \geq MAX\ (b_i)$
3. Find the lower bound inequality $-A_j \geq MIN\ (b_j)$
4. Remove all other inequalities from A and -A
5. Simplify and check for consistency including equalities which may be in A and -A:

   if $-A_i \geq -b2$ and $A_i \geq b1 \Leftrightarrow b1 \leq A_i \leq b2 \rightarrow$
   consistent if $(b1 \leq b2)$

   if $-A_i \geq -b$ and $A_i \geq b \rightarrow A_i = b$

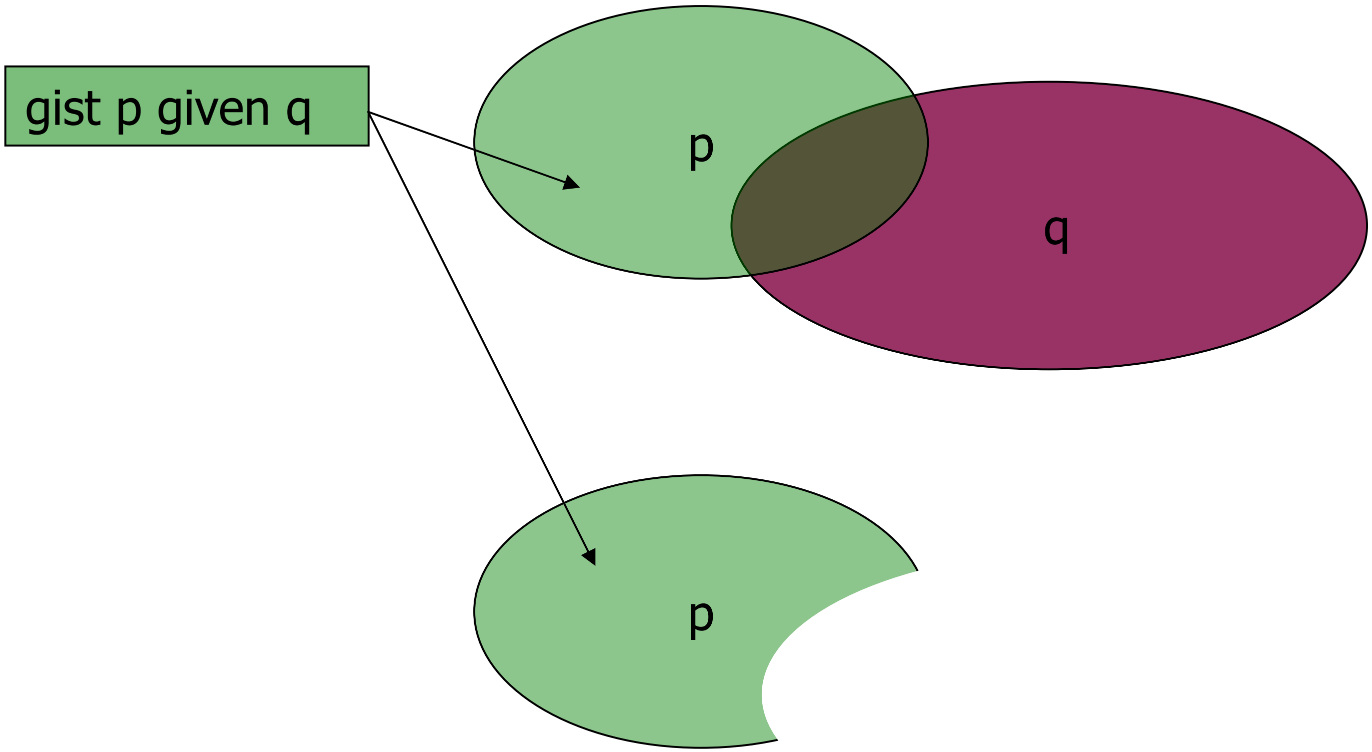
   …and similar

7. Do the same simplification for all group of the parallel constraint
8. Return the simplified system or inconsistency if found

Elimination of redundant constrains using gist

- The gist operation was introduced by W. Pugh, D. Wonnacott in [1]
- *gist p given q* – conjunction of constraint a minimal subset of constraints of *p* such that ((gist p given q) $\wedge$ q) = (p $\wedge$ q)
- Intuitively *gist p given q* the new information contained in p, given that we already known q
- Gist always has not more constraints than an initial system of constraints

Gist p given q



Computing Gist Algorithm

- If q is satisfiable, we could compute gist p given q as follows:

gist p given q =

if p=True return *True*

else let *c* be constraint in *p*

if $p^{c}_{\sim(c)} \wedge q$ is *satisfiable,*

then return $c \wedge (gist\ p^{c}_{True}\ given\ (q \wedge c)$

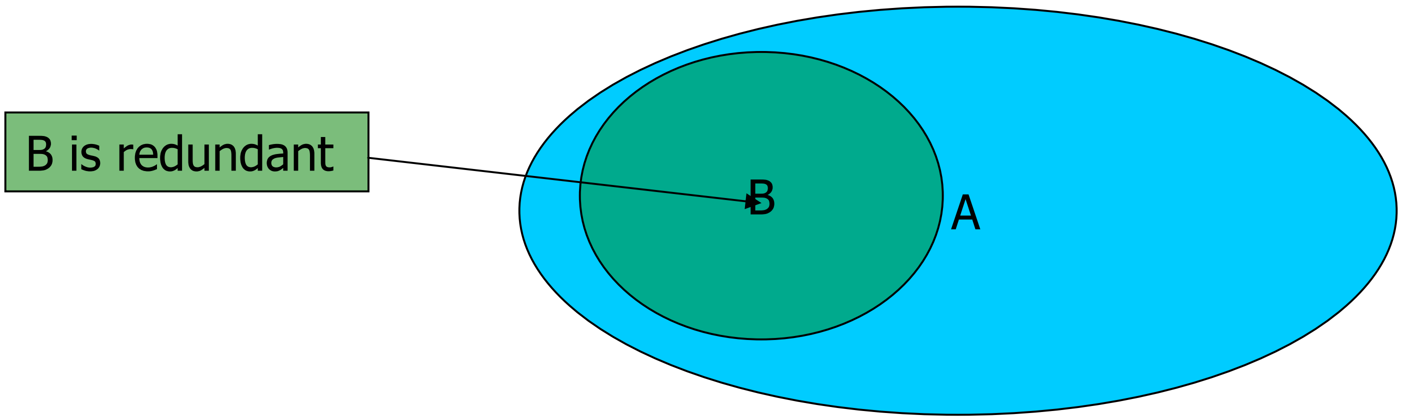
else return $gist\ p^{c}_{True}\ given\ q$

Where p^{oldc}_{newc} is *p* with *oldc* replaced by *newc*

Checking tautologies with Gist

- *Gist p given q = True* $\Leftrightarrow$ $q = (p \wedge q)$ $\Leftrightarrow$ $(q \Rightarrow p)$
- We can simplify disjunction of conjuncts:

  Let $A \vee B \Leftrightarrow A$ *if* $A \Rightarrow B$



Using gist to simplify negations

- $A \wedge (\text{gist } B \text{ given } A) \equiv (A \wedge B)$

  $A \wedge \neg B \equiv A \wedge \neg(A \wedge B)$

  $\equiv A \wedge \neg(A \wedge (\text{gist } B \text{ given } A))$

  $\equiv A \wedge \neg\ (\text{gist } B \text{ given } A)$
- $A \wedge \neg(\text{gist } B \text{ given } A)$ often have fewer clauses than $A \wedge \neg B$, since gist B given A often have fewer clauses than B [2]

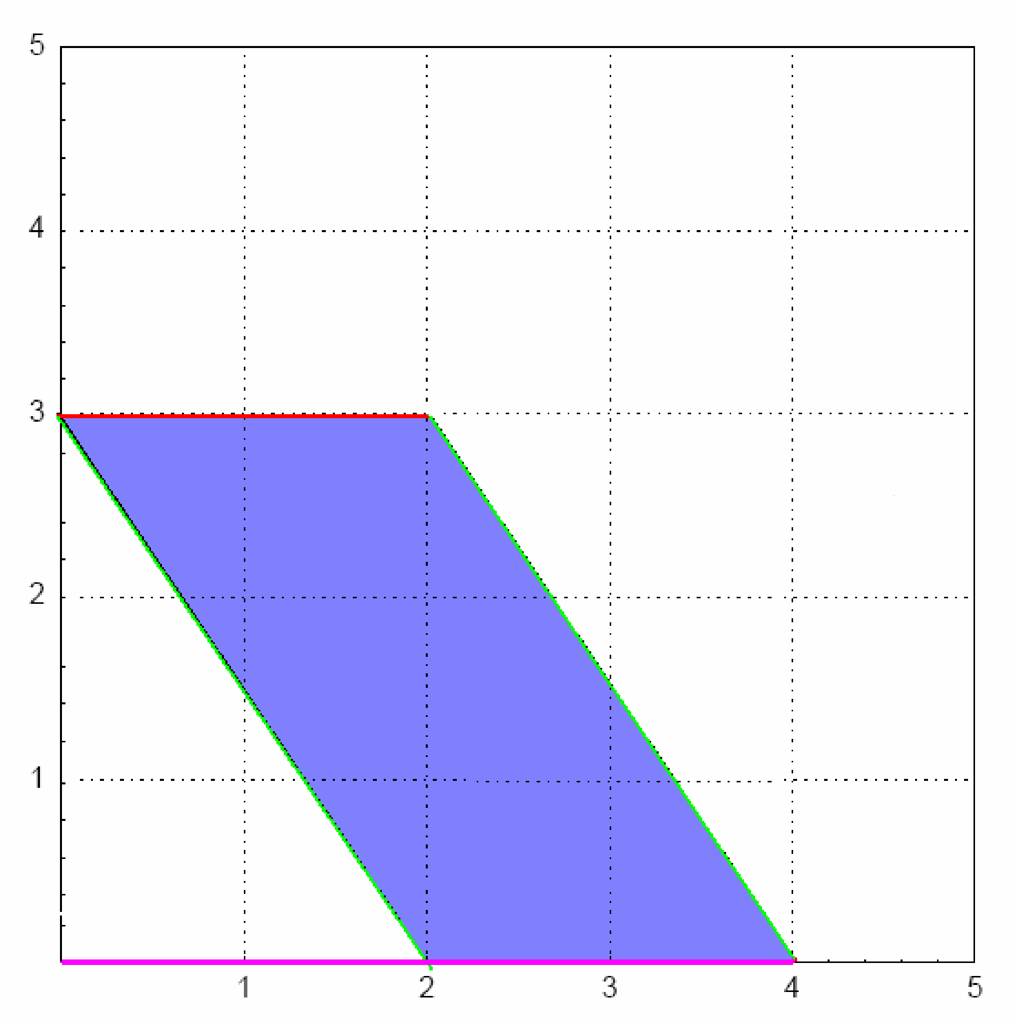
Disequalities handling

- Generic disequality transformation

$$(x_1 \neq a_1) \Leftrightarrow (a_1 < x_1) \vee (x_1 < a_1)$$

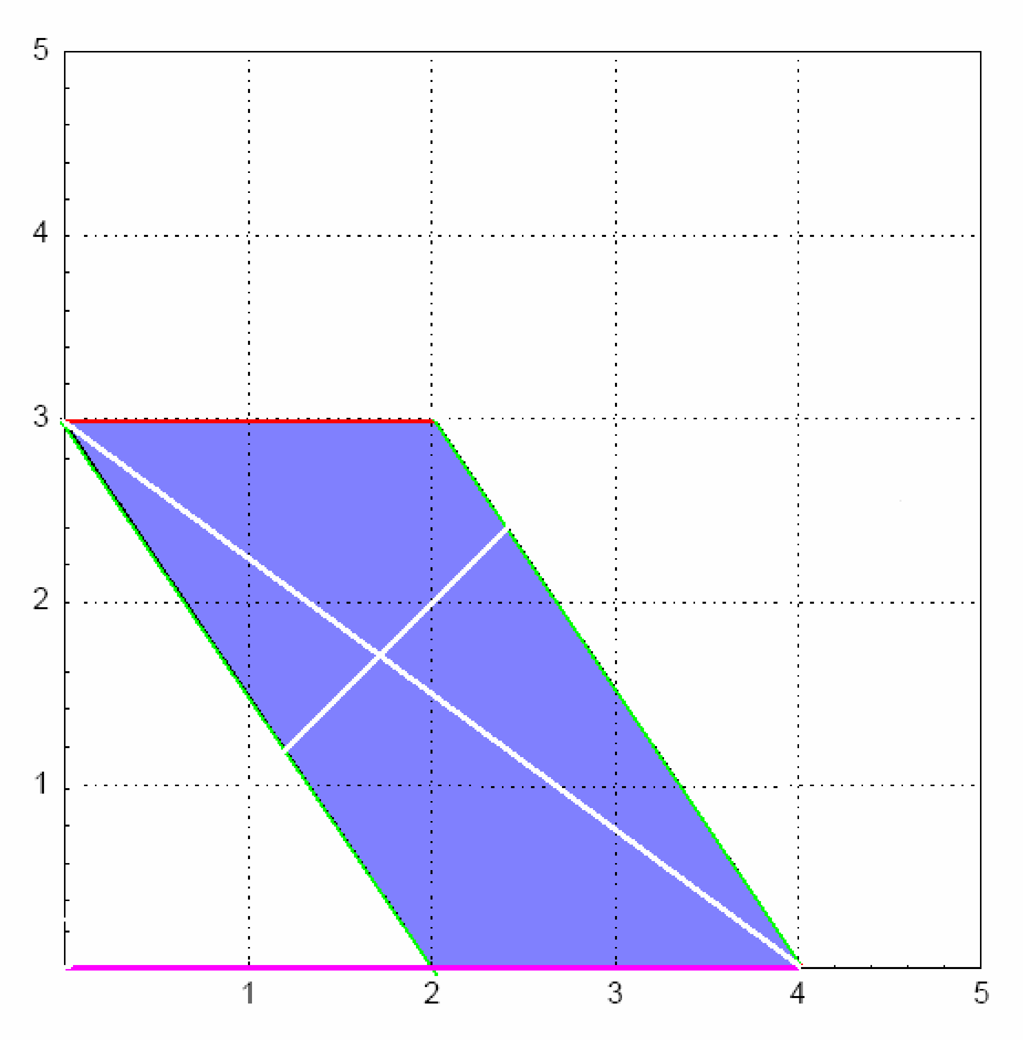
- If this transformation followed by conversion to DNF the size of problem increases exponentially
- Satisfiability test of a conjunction of m inequalities and k disequalities involves 2^k satisfiability tests of conjunctions of m+k inequality constraints

Disequalities (cont.)



$$\begin{cases} y \le 3 \\ y \ge 0 \\ 2y + 3x = 6 \\ 2y + 3x = 12 \end{cases}$$

Disequalities (cont.)



$$\begin{cases} y \le 3 \\ y \ge 0 \\ 2y + 3x = 6 \\ 2y + 3x = 12 \\ \mathbf{x \neq y} \\ \mathbf{3x + 4y \neq 12} \end{cases}$$

Disequalities are independent [8]. Finite number of disequalities cannot eliminate all solutions. For real only.

Disequalities (cont.)

- Main key is *Independence of the System of Disequality Constraints* [8].
- There is no way for a finite number of disequalities to add up together make the system unsatisfiable
- Thus, satisfiability testing of a conjunction m inequalities and k disequalities on real variables can be treated *2k* satisfiability tests of m + 1 inequalities
- The independence property of disequalities allows obtain DNF in polynomial time
- For the integer arithmetic the disequalities are not independent, so other approach may be used [9]

Technologies used

- SUSE Linux 10.0
- GCC 4.0.2 – C++ compiler
- STL, BOOST (smart_pointer, bind, date/time, multi_index_container) libraries
- CxxTest – unit tests framework
- KDevelop - IDE

Example: TTP 3 nodes model

```
Solve (M)(
        Forall (df12,df13,df21,df23,df31,df32, np1,np2,np3, nc1,nc2,nc3,
                p1,p2,p3, c1,c2,c3, d1,d2,d3)
        (
        (np1=p1+c1+d1)&
        (np2=p2+c2+d2)&
        (np3=p3+c3+d3)&
        (df12 = p1+(-1)*p2 + tau1*(c1+(-1)*c2+d1+(-1)*d2))& /* tau1 */
        (df13 = p1+(-1)*p3 + tau1*(c1+(-1)*c3+d1+(-1)*d3))& /* tau1 */
        (df21 = p2+(-1)*p1 + tau2*(c2+(-1)*c1+d2+(-1)*d1))& /* tau2 */
        (df23 = p2+(-1)*p3 + tau2*(c2+(-1)*c3+d2+(-1)*d3))& /* tau2 */
        (df31 = p3+(-1)*p1 + tau3*(c3+(-1)*c1+d3+(-1)*d1))& /* tau3 */
        (df32 = p3+(-1)*p2 + tau3*(c3+(-1)*c2+d3+(-1)*d2))& /* tau3 */
        (nc1= 0.5*(df21+df31))&
        (nc2= 0.5*(df12+df32))&
        (nc3= 0.5*(df23+df13))&
        (( -1 <= d1) & (d1 <=1))&
        (( -1 <= d2) & (d2 <=1))&
        (( -1 <= d3) & (d3 <=1))&
        ((-1)*M<=p1+(-1)*p2)&
        ((-1)*M<=p1+(-1)*p3)&
        ((-1)*M<=p3+(-1)*p2)&
        (p1 +(-1)* p2<=M)&
        (p1 +(-1)* p3<=M)&
        (p3 +(-1)* p2<=M)&
        (2+(-1)* M<=(p1+(-1)*p2+c1+(-1)*c2))&
        (2+(-1)* M<=(p1+(-1)*p3+c1+(-1)*c3))&
        (2+(-1)* M<=(p3+(-1)*p2+c3+(-1)*c2))&
        ((p1+(-1)*p2+c1+(-1)*c2)<=M +(-1)*2)&
        ((p1+(-1)*p3+c1+(-1)*c3)<=M +(-1)*2)&
        ((p3+(-1)*p2+c3+(-1)*c2)<=M +(-1)*2)
         ->
        (2+(-1)*M<=(np1+(-1)*np2+nc1+(-1)*nc2))&
        (2+(-1)*M<=(np1+(-1)*np3+nc1+(-1)*nc3))&
        (2+(-1)*M<=(np3+(-1)*np2+nc3+(-1)*nc2))&
        ((np1+(-1)*np2+nc1+(-1)*nc2)<=M+(-1)*2)&
        ((np1+(-1)*np3+nc1+(-1)*nc3)<=M+(-1)*2)&
        ((np3+(-1)*np2+nc3+(-1)*nc2)<=M+(-1)*2)
));
```

21 variables

12 implicit disequalities

Hard to solve by a generic Fourier-Motrzkin solver

1..60 sec by the partly enhanced Solver. The time depends on the tau_i values.

tau_i: (0..1)

Future Plans

- Implement ALL presented here
- Algorithms enhancement
- Compare speed of the variables projection to the PPL library
- Experiment with the Simplex based methods
- Experiment with the representation of polytope using vectors of points and rays
- Pick up a faster projection algorithm

Conclusions

- Methods for implementing of an efficient solver over real (rational) numbers are presented
- Redundancy elimination and complexity handling techniques are outlined
- The Solver implemented using the presented techniques believed to be efficient enough to handle the real-life verification problems

References

1. W. Pugh, D. Wonnacott Going beyond Integer Programming with the Omega Test to eliminate False Data Dependences. December, 1992
2. W. Pugh, D. Wonnacott Exact Method for Analysis of Value-based Array Data Dependences. December, 1993
3. PPL library: http://www.cs.unipr.it/ppl/
4. Shostak, R. 1981. Deciding Linear Inequalities by Computing Loop Residues. J. ACM 28, 4 (Oct. 1981), 769-779. DOI= http://doi.acm.org/10.1145/322276.322288
5. Shuvendu K. Lahiri, Madanlal Musuvathi An Efficient Nelson-Oppen Decision Procedure for Difference constraints over Rationals
6. Harald Rues, Natarajan Shankar Solving Linear arithmetic Constraints, CSL Technical Report CSL-SRI-04-01, January 2004
7. http://mathworld.wolfram.com/GaussianElimination.html
8. Jean-Louis Lassez, Ken McAloon Independence of Negative Constraints
9. R. Seater D. Wonnacott Efficient Manipulation of Disequalities During Dependence Analysis

Thank you.

Questions?