CS389L: Automated Logical Reasoning

Omega Test

Işıl Dillig

Theory of Integers

- Earlier, we talked about the theory of integers $T_{\mathbb{Z}}$
- Signature of $T_{\mathbb{Z}}$:

  $\Sigma_{\mathbb{Z}}: \{\ldots, -2, -1, 0, 1, 2, \ldots, -3\cdot, -2\cdot, 2\cdot, 3\cdot, \ldots, +, -, =, >\}$
- This theory also called linear arithmetic over integers
- Since equal in expressive power to Presburger arithmetic, people also refer to it as Presburger arithmetic
- Today and next lecture: Look at algorithms for deciding satisfiability in quantifier-free fragment of $T_{\mathbb{Z}}$

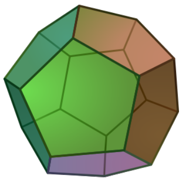
Problem Description

- As in previous two lectures, we'll consider $T_{\mathbb{Z}}$ formulas without disjunctions
- Problem we want to solve: Given an $m \times n$ matrix A with only integer coefficients and a vector $\vec{b}$ in $\mathbb{Z}^n$, does

  $$A\vec{x} \leq \vec{b}$$

  have any integer solutions?
- Problem very similar to that from last lecture, but this time we only accept integer solutions; rational solutions not ok!
- This requirement actually makes problem much harder
- Finding solution over rationals is poly-time, but integer problem is NP-complete even without disjunctions

Geometric Description



- As before the system $A\vec{x} \leq \vec{b}$ defines a polytope
- Last time we asked the question: Is the polytope empty?
- This time, we want to know if polytope contains integer points
- While the polytope is convex, the space formed by all integer points in polytope is not convex
- Unfortunately, non-convexity makes problem much harder to solve

Overview of Techniques

- Two different techniques for solving linear integer inequalities
  1. Elimination-based techniques: Omega Test, Cooper's method
  2. Relaxation-based techniques: Branch-and-bound, Gomory cuts, Cuts-from-Proofs
- Elimination-based techniques eliminate variables one by one until system becomes trivially solvable
- Relaxation-based techniques first drop the integrality requirement and look for a real valued solution
- Then, they iteratively introduce additional constraints to guide search for integer solution
- Called "relaxation-based" because they first solve LP-relaxation

The Omega Test: Historical Perspective

- Omega Test: invented in early 1990's for compiler optimizations
- Particular application: array dependence analysis
- Array dependence analysis: "Can two expressions a[i] and a[j] refer to same element?"
- Can use this information to reorder read and writes from the array and perform operations in parallel

Array Dependence Analysis Example

- Consider the following code snippet:

```
for(i=1; i<= 100; i++) {
  for(j=i; j<= 100; j++)
     a[i, j+1] = a[100, j]
}
```

- Can the expressions `a[i, j+1]` and `a[100,j]` ever refer to same element (not necessarily in same iteration)? No!
- Thus, no array element is both read and written to in the loop
- Hence, we can optimize code by performing assignments in parallel!

Array Dependence Analysis as Integer Constraints

```
for(i=1; i<= 100; i++) {
  for(j=i; j<= 100; j++)
     a[i, j+1] = a[100, j]
}
```

- Can express dependence analysis as linear integer constraints
- Variables w_i and w_j denote array indices when write is performed
- Variables r_i and r_j denote array indices when read is performed
- How do we express that same element is both read and written to? $w_i = r_i \land w_j = r_j$

Array Dependence Analysis as Integer Constraints, cont

```
for(i=1; i<= 100; i++) {
  for(j=i; j<= 100; j++)
     a[i, j+1] = a[100, j]
}
```

- Based on loop start/end conditions, what are constraints on w_i? $1 \leq w_i \leq 100$
- What are constraints on w_j? $w_i < w_j \leq 101$
- What are constraints on r_i? $r_i = 100$
- What are constraints on r_j? $w_i \leq r_j \leq 100 \land r_j = w_j - 1$

Array Dependence Analysis as Integer Constraints, cont

- Thus, an array element may be both read and written in the loop if the conjunction of these constraints is satisfiable:

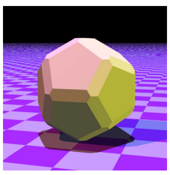
$$w_i = r_i \land w_j = r_j$$
$$1 \leq w_i \leq 100 \land w_i < w_j \leq 101$$
$$r_i = 100 \land w_i \leq r_j \leq 100 \land r_j = w_j - 1$$

- Since array indices can't be real numbers, only interested in integer solution
- Since this constraint has no integer solutions, there is no dependence between array reads and writes
- Thus, all writes can be done in parallel

Applications of Theory of Integers

- Array dependence analysis one application of decision procedure for theory of integers
- Omega Test was initially invented to do better job with array dependence analysis
- Many other applications in software verification, compiler optimizations, operations research, ...

Omega Test: Main Idea



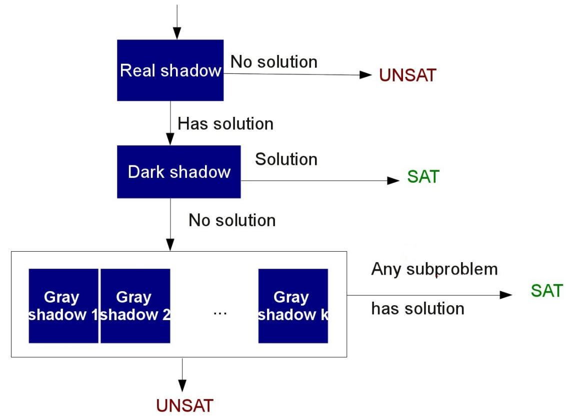
- Main idea: Eliminate variables one by one from the initial system $A\vec{x} \leq \vec{b}$
- Geometrically, eliminating a variable corresponds to computing a projection of a polytope in n-dimensional space to an $n-1$-dimensional space
- Since the polytope has one less dimension at each step, resulting problem is easier to solve than the previous one

Projections in Omega Test

Omega test computes three kind of projections, called shadows:

1. Real Shadow
   - Overapproximates satisfiability over integers
   - If real shadow has no solutions, neither does original problem
2. Dark Shadow
   - Underapproximates satisfiability over integers
   - If dark shadow has solution, original problem has solution
3. Gray Shadows
   - These correspond to areas between real and dark shadow that might contain integer points
   - Omega test constructs multiple gray shadows

Omega Test Work Flow



The Real Shadow

- When constructing the real shadow, we ignore requirement that solution must be integer
- Thus, resulting projection overapproximates satisfiability of original problem
- To construct real shadow, we use the Fourier-Motzkin variable elimination technique

Fourier-Motzkin Variable Elimination

- Suppose we want to eliminate variable x_n from $A\vec{x} \leq \vec{b}$
- Consider an inequality $\sum_{j=1}^{n} a_{ij}x_j \leq b_i$
- This can be rewritten as $a_{in}x_n \leq b_i - \sum_{j=1}^{n-1} a_{ij}x_j$
- If a_{in} is positive, this yields an upper bound on x_n:

$$x_n \leq \frac{b_i}{a_{in}} - \sum_{j=1}^{n-1} \frac{a_{ij}}{a_{in}} x_j$$

- If a_{in} is negative, this yields lower bound on x_n:

$$x_n \geq \frac{b_i}{a_{in}} - \sum_{j=1}^{n-1} \frac{a_{ij}}{a_{in}} x_j$$

Fourier-Motzkin Variable Elimination, cont.

- Thus, if we have $A\vec{x} \leq \vec{b}$ has two rows i and k with positive and negative coefficients for x_n, this yields the inequality:

$$\frac{b_k}{a_{kn}} - \sum_{j=1}^{n-1} \frac{a_{kj}}{a_{kn}} x_j \leq x_n \leq \frac{b_i}{a_{in}} - \sum_{j=1}^{n-1} \frac{a_{ij}}{a_{in}} x_j$$

- We eliminate x_n by removing it from the middle of inequality:

$$\frac{b_k}{a_{kn}} - \sum_{j=1}^{n-1} \frac{a_{kj}}{a_{kn}} x_j \leq \frac{b_i}{a_{in}} - \sum_{j=1}^{n-1} \frac{a_{ij}}{a_{in}} x_j$$

- If we do this for every pair of inequalities with positive and negative coefficients for x_n, this yields the real shadow

Fourier-Motzkin Example

- Consider the set of inequalities:

$$x \leq y + 10 \quad y \leq 15 \quad -x + 20 \leq y$$

- Let's compute real shadow on x-axis using Fourier-Motzkin
- Isolate y on one side:

$$(1)\ x - 10 \leq y \quad (2)\ y \leq 15 \quad (3)\ -x + 20 \leq y$$

- From (1) and (2), we get $x - 10 \leq 15$, i.e., $x \leq 25$
- From (2) and (3), we get $-x + 20 \leq 15$, i.e. $x \geq 5$
- Thus, real shadow on x-axis is $5 \leq x \leq 25$

Dark Shadow

- The second projection Omega test constructs is dark shadow
- Dark shadow underapproximates satisfiability
- Suppose we want to eliminate variable x from $A\vec{x} \leq \vec{b}$
- Dark shadow only projects those parts of polytope that are at least one unit thick in the x-dimension
- If dark shadow has integer solution, original polytope must also have integer solution. Why?
- Since polytope is at least one unit thick above the dark shadow in x-dimension, we are guaranteed to have an integer solution for x as well!

Math Behind the Dark Shadow

- As in real shadow, consider a pair of inequalities corresponding to lower and upper bounds on x:

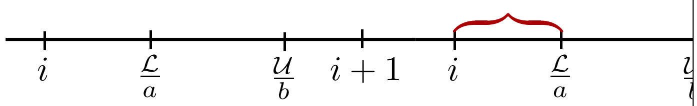
$$\mathcal{L} \leq ax \quad bx \leq \mathcal{U}$$

- These imply:

$$\frac{\mathcal{L}}{a} \leq x \leq \frac{\mathcal{U}}{b}$$

- Now, suppose there is no integer between $\frac{\mathcal{L}}{a}$ and $\frac{\mathcal{U}}{b}$
- Consider first integer i smaller than $\frac{\mathcal{L}}{a}$

Math Behind the Dark Shadow, cont.



- Thus, we have the following inequalities:

$$\frac{\mathcal{L}}{a} - i \geq \frac{1}{a}$$

$$i + 1 - \frac{\mathcal{U}}{b} \geq \frac{1}{b}$$

- If we sum these up, we get:

$$\frac{\mathcal{L}}{a} - \frac{\mathcal{U}}{b} + 1 \geq \frac{1}{a} + \frac{1}{b}$$

Math Behind Dark Shadow, cont

- If we rearrange this equation, we get:

$$b\mathcal{L} - a\mathcal{U} \geq b + a - ab$$

- Finally, multiplying both sides by -1:

$$a\mathcal{U} - b\mathcal{L} \leq ab - a - b \quad (*)$$

- Recall: We derived this equation by assuming that there is no integer solution for x
- That is, we showed "If there is no integer solution for x, then (*) must hold"
- Thus, negation of (*) guarantees there exists integer solution for x!

Math Behind Dark Shadow, cont

- Thus, negation of (*)

$$a\mathcal{U} - b\mathcal{L} > ab - a - b \quad (**)$$

guarantees there is an integer value for x!

- Thus, to construct dark shadow, we remove inequalities containing x and add inequality $(**)$
- Resulting projection is underapproximation because only projects those parts that are at least one unit thick, but there might be an integer solution for x even if it's not unit thick

Dark Shadow Example

$$4y \geq x \quad (1)$$
$$2y \geq 6 - 3x \quad (2)$$
$$3y \leq 7 - x \quad (3)$$

- Using (1), (3), we have $a = 4,\ \mathcal{L} = x$, and $b = 3,\ \mathcal{U} = 7 - x$:

$$4(7 - x) - 3x > 12 - 4 - 3 \quad \Rightarrow \quad x < \frac{23}{7}$$

- Using (2), (3), $a = 2,\ \mathcal{L} = 6 - 3x$, and $b = 3,\ \mathcal{U} = 7 - x$:

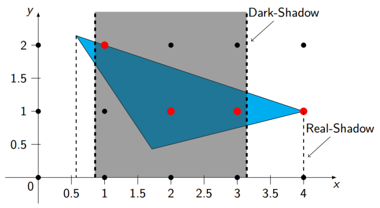
$$2(7 - x) - 3(6 - 3x) > 6 - 3 - 2 \quad \Rightarrow \quad x > \frac{5}{7}$$

- Since $\frac{5}{7} < x < \frac{23}{7}$ has integer solutions, system is satisfiable

Dark Shadow Example, cont.

$$4y \geq x \quad (1)$$
$$2y \geq 6 - 3x \quad (2)$$
$$3y \leq 7 - x \quad (3)$$

- Geometrically:



Gray Shadows

- Recall: Real shadow overapproximates the problem, and dark shadow underapproximates it.
- If real shadow has integer solutions, but dark shadow does not, we still don't know if original problem has integer solutions.
- In this case, Omega test constructs projections called gray shadows
- Gray shadows look for integer solutions outside the dark shadow, but inside the real shadow.

Constructing the Gray Shadow

- Consider again the pair of inequalities:

$$\mathcal{L} \leq ax \qquad bx \leq \mathcal{U}$$

- By construction, any point in the real shadow satisfies:

$$b\mathcal{L} \leq abx \leq a\mathcal{U} \quad (1)$$

- Also, by construction, any point outside dark shadow satisfies:

$$a\mathcal{U} - b\mathcal{L} \leq ab - a - b$$

- We can rewrite above as: $a\mathcal{U} \leq b\mathcal{L} + ab - a - b \quad (2)$
- Combining (1) and (2), we have:

$$b\mathcal{L} \leq abx \leq b\mathcal{L} + ab - a - b$$

Constructing Gray Shadow, cont.

- Thus, any point inside real shadow but outside dark shadow must satisfy:

$$b\mathcal{L} \leq abx \leq ab + b\mathcal{L} - a - b$$

- Dividing by b, points in the gray shadow must satisfy:

$$\mathcal{L} \leq ax \leq \mathcal{L} + \frac{ab - a - b}{b}$$

- Observe: If x is an integer, ax must also be integer
- Furthermore, ax must be equal to

$$\mathcal{L} + i$$

  for some i in the range $[0, \frac{ab-a-b}{b}]$

Constructing Gray Shadow, cont.

- Thus, we construct each gray shadow by adding the equality:

$$ax = \mathcal{L} + i$$

  for each integer i in the range $[0, \frac{ab-a-b}{b}]$

- If any subproblem has integer solution, then so does original problem
- If no subproblem has integer solution, original problem unsatisfiable

Remark about Gray Shadows

- Observe: If there are n integers between 0 and $\frac{ab-a-b}{b}$, Omega test constructs n gray shadows
- Thus, Omega test is very sensitive to coefficients in formula
- The larger a is, the more gray shadows we must consider
- Nightmare for Omega test: Real shadow has solution, dark shadow has no solution, and coefficient a is very large, and problem is unsatisfiable
- In this case, Omega test must solve a very large number of subproblems

Gray Shadow Example

Given the following set of constraints:

$$5y \leq 3x + 3, \qquad 4y \leq 9 - 2x, \qquad 4y \geq 3$$

Real-Shadow:

$$\frac{1}{4} \leq x \leq 3$$

Dark-Shadow:

$$\frac{15}{12} \leq x \leq \frac{15}{8}$$

- Real shadow has solutions, but dark shadow does not; so, need to explore gray shadows...

Example, cont.

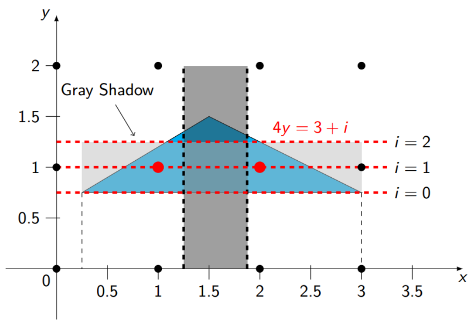
- Derive gray shadow using constraints (1) and (3):

$$5y \leq 3x + 3 \ \ (\text{LB}) \qquad 4y \geq 3 \ (\text{UB})$$

- Here, $\mathcal{L} = 3$, $a = 4$, $\mathcal{U} = 3x + 3$, $b = 5$
- Try $4x = 3 + i$ for $i \in [0, \frac{4 \cdot 5 - 4 - 5}{5}]$ (i.e., $i \in [0, 2]$)
- Satisfiable for $i = 1 \Rightarrow$ Has integer solutions

Example, cont.

Geometrically:



An Optimization

- Omega test uses important optimization to handle equality constraints
- Equality constraints can be expressed as pair of inequalities, but handling equalities directly much more efficient
- Thus, Omega test has special preliminary step where it gets rid of all equality constraints
- Uses interesting coefficient-reducing technique based on symmetric modulo
- Details are in paper posted on class webpage - strongly encouraged to read!

Omega Test Summary

- Omega test is an elimination-based technique for solving linear inequalities over integers
- Constructs three kinds of projections: real shadow, dark shadow, gray shadow
- Problem has no solution if real shadow has no solution
- Problem has solution if dark shadow has solution
- Otherwise, problem has solution iff one of the dark shadows has solution
- Omega test handles equalities specially using the symmetric modulo technique