Script **generated by TTT**

Title: Seidl: Programmoptimierung (20.01.2016)

Date: Wed Jan 20 10:20:38 CET 2016

Duration: 90:35 min

Pages: 45

Idea

- We successively remove variables. Thereby we omit division ...
- If x only occurs with coefficient ± 1, we apply Fourier-Motzkin elimination.
- Otherwise, we provide a bound for a positive multiple of x ...

Consider, e.g., (1) and (6) :

$$\begin{array}{rcl} 6 \cdot x_1 & \leq & 17 + 2x_2 \\ 9 - x_2 & \leq & 4 \cdot x_1 \end{array}$$

W.l.o.g., we only consider strict in-equations:

$$\begin{array}{rcl} 6 \cdot x_1 & < & 18 + 2x_2 \\ 8 - x_2 & < & 4 \cdot x_1 \end{array}$$

... where we always divide by gcds:

$$\begin{array}{rcl} 3 \cdot x_1 & < & 9 + x_2 \\ 8 - x_2 & < & 4 \cdot x_1 \end{array}$$

This implies:

$$3 \cdot (8 - x_2) \quad < \quad 4 \cdot (9 + x_2)$$

W.l.o.g., we only consider strict in-equations:

$$\begin{array}{rcl} 6 \cdot x_1 & < & 18 + 2x_2 \\ 8 - x_2 & < & 4 \cdot x_1 \end{array}$$

... where we always divide by gcds:

$$\begin{array}{rcl} 3 \cdot x_1 & < & 9 + x_2 \\ 8 - x_2 & < & 4 \cdot x_1 \end{array}$$

This implies:

$$3 \cdot (8 - x_2) \quad < \quad 4 \cdot (9 + x_2)$$

715

We thereby obtain:

- If one derived in-equation is unsatisfiable, then also the overall system.
- If all derived in-equations are satisfiable, then there is a solution which, however, need not be integer.
- An integer solution is guaranteed to exist if there is sufficient separation between lower and upper bound ...
- Assume $\alpha < a \cdot x \qquad b \cdot x < \beta$.

  Then it should hold that:

$$b \cdot \alpha < a \cdot \beta$$

and moreover:

$$\boxed{a \cdot b} < a \cdot \beta - b \cdot \alpha$$

W.l.o.g., we only consider strict in-equations:

$$\begin{array}{rcl} 6 \cdot x_1 & < & 18 + 2x_2 \\ 8 - x_2 & < & 4 \cdot x_1 \end{array}$$

... where we always divide by gcds:

$$\begin{array}{rcl} 3 \cdot x_1 & < & 9 + x_2 \\ 8 - x_2 & < & 4 \cdot x_1 \end{array}$$

This implies:

$$3 \cdot (8 - x_2) \quad < \quad 4 \cdot (9 + x_2)$$

We thereby obtain:

- If one derived in-equation is unsatisfiable, then also the overall system.
- If all derived in-equations are satisfiable, then there is a solution which, however, need not be integer.
- An integer solution is guaranteed to exist if there is sufficient separation between lower and upper bound ...
- Assume $\alpha < a \cdot x \qquad b \cdot x < \beta$.

  Then it should hold that:

$$b \cdot \alpha < a \cdot \beta$$

and moreover:

$$\boxed{\boxed{a \cdot b} < a \cdot \beta - b \cdot \alpha}$$

716

... in the Example:

$$12 \quad < \quad 4 \cdot (9 + x_2) - 3 \cdot (8 - x_2)$$

or:

$$12 \quad < \quad 12 + 7x_2$$

or:

$$0 \quad < \quad x_2$$

In the example, also these strengthened in-equations are satisfiable

$\Longrightarrow$ the system has a solution over $\mathbb{Z}$.

Discussion

- If the strengthened in-equations are satisfiable, then also the original system. The reverse implication may be wrong !
- In the case where upper and lower bound are not sufficiently separated, we have:

$$a \cdot \beta \leq b \cdot \alpha + \boxed{a \cdot b}$$

or:

$$\cancel{b} \cdot \alpha < a\cancel{b} \cdot x < \cancel{b} \cdot \alpha + \boxed{a \cdot \cancel{b}}$$

Division with b yields:

$$\alpha < a \cdot x < \alpha + \boxed{a}$$

$\Longrightarrow$ $\boxed{\alpha + i = a \cdot x}$ for some $i \in \{1, \ldots, a-1\}$!!!

Discussion (cont.)

→ Fourier-Motzkin Elimination is not the best method for rational systems of in-equations.

→ The Omega test is necessarily exponential.

If the system is solvable, the test generally terminates rapidly.

It may have problems with unsolvable systems.

→ Also for ILP, there are other/smarter algorithms ...

→ For programming language problems, however, it seems to behave quite well.

4. Generalization to a Logic

Disjunction

$$\begin{array}{lll} (x - 2y = 15 & \wedge & x + y = 7) \quad \vee \\ (x + y = 6 & \wedge & 3x + z = -8) \end{array}$$

Quantors:

$$\exists\, x : \quad z - 2x = 42 \quad \wedge \quad z + x = 19$$

4. Generalization to a Logic

Disjunction:

$$\begin{array}{lll} (x - 2y = 15 & \wedge & x + y = 7) \quad \vee \\ (x + y = 6 & \wedge & 3x + z = -8) \end{array}$$

Quantors:

$$\exists\, x : \quad z - 2x = 42 \quad \wedge \quad z + x = 19$$

$\Longrightarrow$ Presburger Arithmetic

Mojzesz Presburger, 1904–1943 (?)

Presburger Arithmetic = full arithmetic without multiplication

Arithmetic : highly undecidable even incomplete

Presburger Arithmetic = full arithmetic without multiplication

Arithmetic : highly undecidable even incomplete

$\Longrightarrow$ Hilbert's 10th Problem

$\Longrightarrow$ Gödel's Theorem

Presburger Formulas over $\mathbb{N}$:

$$\begin{aligned}\phi \quad ::= \quad & x + y = z \quad | \quad x = n \quad | \\ & \phi_1 \wedge \phi_2 \quad | \quad \neg\phi \quad | \\ & \exists\, x : \ \phi\end{aligned}$$

$$x \geqslant y$$

$$\Longleftrightarrow \exists z .\ y + z = x$$

Idea: Code the values of the variables as Words

213	t	1	0	1	0	1	0	1	1
42	z	0	1	0	1	0	1	0	0
89	y	1	0	0	1	1	0	1	0
17	x	1	0	0	0	1	0	0	0

Presburger Formulas over $\mathbb{N}$:

$$
\begin{array}{lll}
\phi & ::= & x + y = z \quad | \quad x = n \quad | \\
 & & \phi_1 \wedge \phi_2 \quad | \quad \neg\phi \quad | \\
 & & \exists\, x : \ \phi
\end{array}
$$

Goal: PSAT

Find values for the free variables in $\mathbb{N}$ such that ϕ holds ...

Idea: Code the values of the variables as Words

213	t	1	0	1	0	1	0	1	1	0
42	z	0	1	0	1	0	1	0	0	0
89	y	1	0	0	1	1	0	1	0	0
17	x	1	0	0	0	1	0	0	0	0

Observation

The set of satisfying variable assignments is regular !

Observation

The set of satisfying variable assignments is regular !

$\phi_1 \wedge \phi_2$	$\Longrightarrow$	$\mathcal{L}(\phi_1) \cap \mathcal{L}(\phi_2)$	(Intersection)
$\neg\phi$	$\Longrightarrow$	$\overline{\mathcal{L}(\phi)}$	(Complement)
$\boxed{\exists\, x : \;\; \phi}$	$\Longrightarrow$	$\pi_x(\mathcal{L}(\phi))$	(Projection)

$$\pi_x\left(\begin{bmatrix}0\\1\\1\\0\end{bmatrix}\right) = \begin{bmatrix}0\\0\\1\\0\end{bmatrix}$$

Projecting away the x-component:

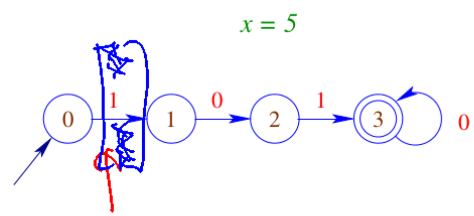
213	t	1	0	1	0	1	0	1	1
42	z	0	1	0	1	0	1	0	0
89	y	1	0	0	1	1	0	1	0

Caveat

- Our representation of numbers is not unique: 011101 should be accepted iff every word from $011101 \cdot 0^*$ is accepted!
- This property is preserved by union, intersection and complement.
- It is lost by projection !!!

$\Longrightarrow$ The automaton for projection must be enriched such that the property is re-established !!

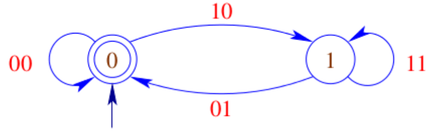
Automata for Basic Predicates



Automata for Basic Predicates

$x + y = z$

$x{+}x = y$

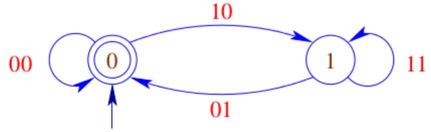


$3x = 5$

$\exists x_2 \exists x_1 \; x + x = x_1 \land$
$x_1 + x = x_2 \land$
$x_2 = 5$

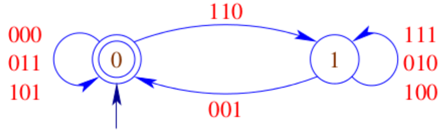
Automata for Basic Predicates

$x{+}x = y$



Automata for Basic Predicates

$x{+}y = z$



Results

Ferrante, Rackoff,1973 : $\text{PSAT} \leq \text{DSPACE}(2^{2^{c \cdot n}})$

Results

Ferrante, Rackoff,1973 : $\text{PSAT} \leq \text{DSPACE}(2^{2^{c \cdot n}})$

Fischer, Rabin,1974 : $\text{PSAT} \geq \text{NTIME}(2^{2^{c \cdot n}})$

3.3 Improving the Memory Layout

Goal

- Better utilization of caches

  $\Longrightarrow$ reduction of the number of cache misses
- Reduction of allocation/de-allocation costs

  $\Longrightarrow$ replacing heap allocation by stack allocation

  $\Longrightarrow$ support to free superfluous heap objects
- Reduction of access costs

  $\Longrightarrow$ short-circuiting indirection chains (Unboxing)

3.3 Improving the Memory Layout

Goal

- Better utilization of caches

  $\Longrightarrow$ reduction of the number of cache misses
- Reduction of allocation/de-allocation costs

  $\Longrightarrow$ replacing heap allocation by stack allocation

  $\Longrightarrow$ support to free superfluous heap objects
- Reduction of access costs

  $\Longrightarrow$ short-circuiting indirection chains (Unboxing)

748

1. Cache Optimization

Idea: local memory access

- Loading from memory fetches not just one byte but fills a complete cache line.
- Access to neighbored cells become cheaper.
- If all data of an inner loop fits into the cache, the iteration becomes maximally memory-efficient ...

1. Cache Optimization

Idea: local memory access

- Loading from memory fetches not just one byte but fills a complete cache line.
- Access to neighbored cells become cheaper.
- If all data of an inner loop fits into the cache, the iteration becomes maximally memory-efficient ...

Possible Solutions

→ Reorganize the data accesses !

→ Reorganize the data !

Such optimizations can be made fully automatic only for arrays.

Example

$$
\begin{array}{l}
\text{for } (j = 1; j < n; j{+}{+}) \\
\quad \text{for } (i = 1; i < m; i{+}{+}) \\
\quad\quad a[i][j] = a[i-1][j-1] + a[i][j];
\end{array}
$$

Possible Solutions

→ Reorganize the data accesses !

→ Reorganize the data !

Such optimizations can be made fully automatic only for arrays.

Example

$$
\begin{array}{l}
\text{for } (j = 1; j < n; j{+}{+}) \\
\quad \text{for } (i = 1; i < m; i{+}{+}) \\
\quad\quad a[i][j] = a[i-1][j-1] + a[i][j];
\end{array}
$$

$\Longrightarrow$ At first, always iterate over the rows!

$\Longrightarrow$ Exchange the ordering of the iterations:

for $(i = 1; i < m; i{+}{+})$
  for $(j = 1; j < n; j{+}{+})$
    $a[i][j] = a[i-1][j-1] + a[i][j];$

When is this permitted???

Possible Solutions

$\rightarrow$ Reorganize the data accesses !

$\rightarrow$ Reorganize the data !

Such optimizations can be made fully automatic only for arrays.

Example

for $(j = 1; j < n; j{+}{+})$
  for $(i = 1; i < m; i{+}{+})$
    $a[i][j] = a[i-1][j-1] + a[i][j];$

$\Longrightarrow$ At first, always iterate over the rows!

$\Longrightarrow$ Exchange the ordering of the iterations:

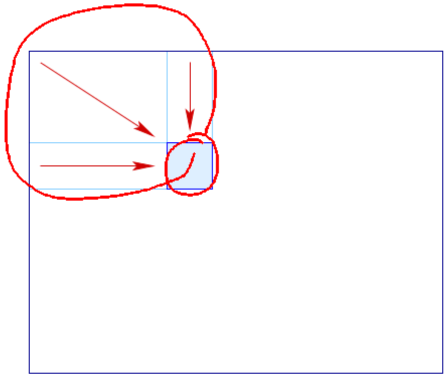
for $(i = 1; i < m; i{+}{+})$
  for $(j = 1; j < n; j{+}{+})$
    $a[i][j] = a[i-1][j-1] + a[i][j];$

When is this permitted???

751

Iteration Scheme: after:

Iteration Scheme: allowed dependencies:



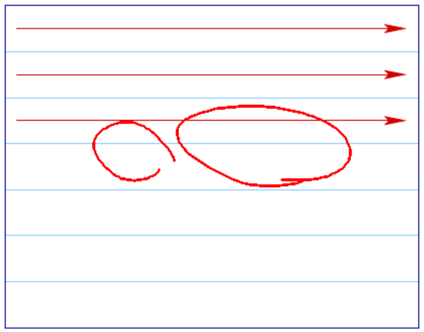
In our case, we must check that the following equation systems have no solution:

Write		Read
(i_1, j_1)	$=$	$(i_2 - 1, j_2 - 1)$
i_1	$\leq$	i_2
j_2	$\leq$	j_1
(i_1, j_1)	$=$	$(i_2 - 1, j_2 - 1)$
i_2	$\leq$	i_1
j_1	$\leq$	j_2

The first implies: $j_2 \leq j_2 - 1$ Hurra!

The second implies: $i_2 \leq i_2 - 1$ Hurra!

Iteration Scheme: after:



In our case, we must check that the following equation systems have no solution:

Write		Read
(i_1, j_1)	$=$	$(i_2 - 1, j_2 - 1)$
i_1	$\leq$	i_2
j_2	$\leq$	j_1
(i_1, j_1)	$=$	$(i_2 - 1, j_2 - 1)$
i_2	$\leq$	i_1
j_1	$\leq$	j_2

The first implies: $j_2 \leq j_2 - 1$ Hurra!

The second implies: $i_2 \leq i_2 - 1$ Hurra!

Example: Matrix-Matrix Multiplication

$$
\begin{array}{l}
\text{for } (i = 0; i < N; i{+}{+}) \\
\quad \text{for } (j = 0; j < M; j{+}{+}) \\
\quad\quad \text{for } (k = 0; k < K; k{+}{+}) \\
\quad\quad\quad c[i][j] = c[i][j] + a[i][k] \cdot b[k][j];
\end{array}
$$

Over $b[][]$ the iteration is columnwise.

Example: Matrix-Matrix Multiplication

$$
\begin{array}{l}
\text{for } (i = 0; i < N; i{+}{+}) \\
\quad \text{for } (j = 0; j < M; j{+}{+}) \\
\quad\quad \text{for } (k = 0; k < K; k{+}{+}) \\
\quad\quad\quad c[i][j] = c[i][j] + a[i][k] \cdot b[k][j];
\end{array}
$$

Over $b[][]$ the iteration is columnwise.

756