Something that . . . that May be Interesting

Andy King

With input from Jacob Howe, Axel Simon, Kathrin Uriu and Joachim von zur Gathen (Frankfurt)

Outline of this talk

Entailment checking for linear systems

- Entailment checking in abstract interpretation
- Syntactic entailment checking

The satisfiability checking approach

- The Fourier-Motzkin procedure
- Revised Fourier-Motzkin procedure

A finite domain approach

- Range consistency constraint methods
- Priming the domain constraints

- Consider $S_1 = \{3w - 2y \leq 1, x - y \leq 1, y - z \leq 0\}$ and $S_2 = \{x - z \leq 1, 3w - 2y \leq 3\}$.
- We say that S_1 entails S_2, written $S_1 \models S_2$, iff $\{\langle w, x, y, z\rangle \in \mathbb{Z}^4 \mid S_1 \text{ holds}\} \subseteq \{\langle w, x, y, z\rangle \in \mathbb{Z}^4 \mid S_2 \text{ holds}\}$.
- Thus $S_1 \models S_2$ corresponds to the $\subseteq$ relation on the sets of *integer* vectors represented by S_1 and S_2;
- Given S_1 and S_2, the entailment check is a *complete*, binary decision procedure that returns:
  - "yes" when $S_1 \models S_2$;
  - "no" when $S_1 \not\models S_2$

  In fact $S_1 \models S_2$ iff $S_1 \models \{x - z \leq 1\}$ and $S_1 \models \{3w - 2y \leq 1\}$ and it is sufficient to suppose S_2 contains a *single* inequality.

Time for a demo!

Entailment checking in abstract interpretation

Entailment checking is *necessary* for checking stability when there is no underlying canonical representation.
Fast entailment checking is *desirable* for:

- filtering operations such as the convex hull S of S_1 and S_2;

Entailment checking in abstract interpretation

Entailment checking is *necessary* for checking stability when there is no underlying canonical representation.
Fast entailment checking is *desirable* for:

- filtering operations such as the convex hull S of S_1 and S_2;
- if $S_1 = \{0 \leq x \leq 0, 0 \leq y \leq 0\}$ and $S_2 = \{0 \leq x, x+1 \leq y \leq x+1\}$ then $\mathit{hull}(S_1, S_2) = \{0 \leq x, x \leq y \leq x+1\}$;

Entailment checking in abstract interpretation

Entailment checking is *necessary* for checking stability when there is no underlying canonical representation.
Fast entailment checking is *desirable* for:

- filtering operations such as the convex hull S of S_1 and S_2;
- if $S_1 = \{0 \leq x \leq 0, 0 \leq y \leq 0\}$ and $S_2 = \{0 \leq x, x+1 \leq y \leq x+1\}$ then $\mathit{hull}(S_1, S_2) = \{0 \leq x, x \leq y \leq x+1\}$;
- if $S_1 \models S_2$ then $\mathit{hull}(S_1, S_2) = S_2$;

Entailment checking in abstract interpretation

Entailment checking is *necessary* for checking stability when there is no underlying canonical representation.
Fast entailment checking is *desirable* for:

- filtering operations such as the convex hull S of S_1 and S_2;
- if $S_1 = \{0 \leq x \leq 0, 0 \leq y \leq 0\}$ and $S_2 = \{0 \leq x, x+1 \leq y \leq x+1\}$ then $\mathit{hull}(S_1, S_2) = \{0 \leq x, x \leq y \leq x+1\}$;
- if $S_1 \models S_2$ then $\mathit{hull}(S_1, S_2) = S_2$;
- compressing a system such as $S_2 \cup \{y \leq 2x+1\}$;

Entailment checking in abstract interpretation

Entailment checking is *necessary* for checking stability when there is no underlying canonical representation.
Fast entailment checking is *desirable* for:

- filtering operations such as the convex hull S of S_1 and S_2;
- if $S_1 = \{0 \leq x \leq 0, 0 \leq y \leq 0\}$ and $S_2 = \{0 \leq x, x+1 \leq y \leq x+1\}$ then $\mathit{hull}(S_1, S_2) = \{0 \leq x, x \leq y \leq x+1\}$;
- if $S_1 \models S_2$ then $\mathit{hull}(S_1, S_2) = S_2$;
- compressing a system such as $S_2 \cup \{y \leq 2x+1\}$;
- since $S_2 \models \{y \leq 2x+1\}$ it follows that $y \leq 2x+1$ is redundant which is important for efficiency (space and time) and for the presentation of the final results

Implementation toolkit

- Want to avoid the use of arrays; list manipulation is preferable;
- Want to avoid the use of foreign language interface; will need to profile and tune the code;
- Can exploit:
  - unification,
  - lightweight constraints,
  - asynchronous concurrency.

Syntactic versus semantic entailment checking

- Observe
  $S_1 = \{3w - 2y \leq 1, x - y \leq 1, y - z \leq 0\} \models \{3w - 2y \leq 3\}$
  since $3w - 2y \leq c \in S_1$ and $c \leq 3$.
- Syntactic entailment checking can probably be used to construct *incomplete* decision procedure:
  - "yes" – for $S_1 \models S_2$;
  - "no" – for $S_1 \not\models S_2$;
  - "dunno" – which could trigger a complete decision procedure
- Not clear whether syntactic filtering is useful; requires benchmarking experiments.

The satisfiability checking approach

- Consider $S_1 \models \{x - z \leq 1\}$.
- Negate the (possibly entailed) inequality:
  $\neg(x - z \leq 1) = (x - z > 1) = (x - z \geq 2) = (z - x \leq -2)$;
- Check whether $S_1 \cup \{z - x \leq -2\}$ is satisfiable, say, using Fourier-Motzkin variable elimination.

The Fourier-Motzkin procedure

$$\left\{\begin{array}{r} \underline{3w-2y\leq 1,} \\ \underline{x-y\leq 1,} \\ \underline{y-z\leq 0,} \\ z-x\leq -2 \end{array}\right\} \underset{\rightarrow}{\text{elim}}\ \text{y} \left\{\begin{array}{r} 3w-2z\leq 1, \\ x-z\leq 1, \\ z-x\leq -2 \end{array}\right\}$$

Because $\left.\begin{array}{r} 3w-2y\leq 1 \\ y-z\leq 0 \end{array}\right\} = \left.\begin{array}{r} 3w-2y\leq 1 \\ 2y-2z\leq 0 \end{array}\right\} = 3w-2z\leq 1$

$$\left.\begin{array}{r} x-y\leq 1 \\ y-z\leq 0 \end{array}\right\} = x-z\leq 1$$

$$\left\{\begin{array}{r} \underline{3w-2z\leq 1,} \\ x-z\leq 1, \\ z-x\leq -2 \end{array}\right\} \underset{\rightarrow}{\text{elim}}\ \text{w} \left\{\underline{\begin{array}{r} x-z\leq 1, \\ z-x\leq -2 \end{array}}\right\} \underset{\rightarrow}{\text{elim}}\ \text{x} \left\{0\leq -1\right\}$$

Simple list operations but each elimination can increase the number of inequalities $O(n^2)$ so this tactic is $O(n^{2^{2^{2^2}}})$ for 4 variables.

An afterthought on Fourier-Motzkin

Recall $S_1 = \{3w - 2y \leq 1, x - y \leq 1, y - z \leq 0\} \models \{x - z \leq 1\}$ iff

- $x - z \leq c$ is a linear combination of S_1 for some $c \leq 1$;

An afterthought on Fourier-Motzkin

Recall $S_1 = \{3w - 2y \leq 1, x - y \leq 1, y - z \leq 0\} \models \{x - z \leq 1\}$ iff

- $x - z \leq c$ is a linear combination of S_1 for some $c \leq 1$;
- there exists non-negative multipliers $\lambda_1, \lambda_2, \lambda_3 \in \mathbb{Q}$ such that $\lambda_1(3w - 2y) + \lambda_2(x - y) + \lambda_3(y - z) = x - z$ and $1.\lambda_1 + 1.\lambda_2 + 0.\lambda_3 = c \leq 1$;

An afterthought on Fourier-Motzkin

Recall $S_1 = \{3w - 2y \leq 1, x - y \leq 1, y - z \leq 0\} \models \{x - z \leq 1\}$ iff

- $x - z \leq c$ is a linear combination of S_1 for some $c \leq 1$;
- there exists non-negative multipliers $\lambda_1, \lambda_2, \lambda_3 \in \mathbb{Q}$ such that $\lambda_1(3w - 2y) + \lambda_2(x - y) + \lambda_3(y - z) = x - z$ and $1.\lambda_1 + 1.\lambda_2 + 0.\lambda_3 = c \leq 1$;
- the following system is satisfiable:

$$S = \left\{ \begin{array}{r} (3\lambda_1) = 0, \\ \lambda_2 = 1, \\ (-2\lambda_1) + (-1\lambda_2) + \lambda_3 = 0, \\ (-1\lambda_3) = -1, \\ \lambda_1 + \lambda_2 \leq 1, \\ -\lambda_1 \leq 0, -\lambda_2 \leq 0, -\lambda_3 \leq 0 \end{array} \right\}$$

An afterthought on Fourier-Motzkin (continued)

$$S = \left\{ \begin{array}{r} \lambda_1 = 0, \\ \lambda_2 = 1, \\ (-2\lambda_1) + (-1\lambda_2) + \lambda_3 = 0, \\ \lambda_3 = 1, \\ \lambda_1 + \lambda_2 \leq 1, \\ -\lambda_1 \leq 0, -\lambda_2 \leq 0, -\lambda_3 \leq 0 \end{array} \right\} \underrightarrow{\text{elim}} \lambda_1 \left\{ \begin{array}{r} \lambda_2 = 1, \\ (-1\lambda_2) + \lambda_3 = 0, \\ \lambda_3 = 1, \\ \lambda_2 \leq 1, \\ -\lambda_2 \leq 0, -\lambda_3 \leq 0 \end{array} \right\}$$

$$\left\{ \begin{array}{r} \lambda_2 = 1, \\ (-1\lambda_2) + \lambda_3 = 0, \\ \lambda_3 = 1, \\ \lambda_2 \leq 1, \\ -\lambda_2 \leq 0, -\lambda_3 \leq 0 \end{array} \right\} \underrightarrow{\text{elim}} \lambda_2 \left\{ \begin{array}{c} \lambda_3 = 1, \\ -\lambda_3 \leq 0 \end{array} \right\} \underrightarrow{\text{elim}} \lambda_3 \left\{ -1 \leq 0 \right\}$$

Single substitution is $O(nm)$ so whole entailment procedure seems to be $O(n^2 m)$ where $n = |var(S_1)|$ and $m = |S_1|$. Also $n \leq m$.

Questions, questions...

- what is this entailment checking reduction strategy called?

Questions, questions...

- what is this entailment checking reduction strategy called?
- is entailment checking $\Omega(n^2 m)$?

Questions, questions...

- what is this entailment checking reduction strategy called?
- is entailment checking $\Omega(n^2 m)$?
- is it possible to realise a substitution such as $\lambda_3 = 1$ with unification?

Questions, questions...

- what is this entailment checking reduction strategy called?
- is entailment checking $\Omega(n^2 m)$?
- is it possible to realise a substitution such as $\lambda_3 = 1$ with unification?
- how does the complexity of the entailment check depend on the *structure* of S_1?

Detecting unsatisfiability with range consistency

```
?- 3*W - 2*Y #=< 1,
   X - Y #=< 1,
   Y - Z #=< 0,
   Z - X #=< -2.

W in inf..sup,
Y in inf..sup,
X in inf..sup,
Z in inf..sup

yes
```

```
?- domain([X,W,Y,Z], -1000, 1000),
   3*W - 2*Y #=< 1,
   X - Y #=< 1,
   Y - Z #=< 0,
   Z - X #=< -2.

no
```

SICStus detects inconsistency in $12004 \ll 1000^4$ domain pruning steps (without employing search).

Enforcing range consistency with indexical daemons

- the constraint `domain([X,W,Y,Z], -1000, 1000)` initialises $\mathtt{X}_{min} = \text{-}1000$, $\mathtt{X}_{max} = 1000$, ..., $\mathtt{Z}_{min} = \text{-}1000$, $\mathtt{Z}_{max} = 1000$.

- a dedicated (fair) scheduler passes control between daemons;
- termination is guaranteed because the domains are finite.

Enforcing range consistency with indexical daemons

- the constraint `domain([X,W,Y,Z], -1000, 1000)` initialises $\mathtt{X}_{min}$ = -1000, $\mathtt{X}_{max}$ = 1000, ..., $\mathtt{Z}_{min}$ = -1000, $\mathtt{Z}_{max}$ = 1000.
- the constraint `X - Y #=< 1` is compiled to:
  - $\mathtt{X}_{max} := \min(\mathtt{Y}_{max} + 1, \mathtt{X}_{max})$;
  - $\mathtt{Y}_{min} := \max(\mathtt{X}_{min} - 1, \mathtt{Y}_{min})$

- a dedicated (fair) scheduler passes control between daemons;
- termination is guaranteed because the domains are finite.

Enforcing range consistency with indexical daemons

- the constraint `domain([X,W,Y,Z], -1000, 1000)` initialises $\mathtt{X}_{min} = \text{-}1000$, $\mathtt{X}_{max} = 1000$, ..., $\mathtt{Z}_{min} = \text{-}1000$, $\mathtt{Z}_{max} = 1000$.
- the constraint `X - Y #=< 1` is compiled to:
  - $\mathtt{X}_{max} := \min(\mathtt{Y}_{max} + 1, \mathtt{X}_{max})$;
  - $\mathtt{Y}_{min} := \max(\mathtt{X}_{min} - 1, \mathtt{Y}_{min})$
- the constraint `3*W - 2*Y #=< 1` is compiled to:
  - $\mathtt{W}_{max} := \min(\lfloor(1 + 2 * \mathtt{Y}_{max})/3\rfloor, \mathtt{W}_{max})$;
  - $\mathtt{Y}_{min} := \max(\lceil(3 * \mathtt{W}_{min} - 1)/2\rceil, \mathtt{Y}_{min})$
- a dedicated (fair) scheduler passes control between daemons;
- termination is guaranteed because the domains are finite.

A bound on integer solutions

Checking $\{3w - 2y \leq 1, x - y \leq 1, y - z \leq 0, z - x \leq -2\}$ for satisfiability is equivalent to solving $C\vec{x} \geq \vec{d}$ where

$$C = \begin{bmatrix} -3 & 0 & 2 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & -1 & 0 & 1 \\ 0 & 1 & 0 & -1 \end{bmatrix} \quad \vec{x} = \begin{bmatrix} w \\ x \\ y \\ z \end{bmatrix} \quad \vec{d} = \begin{bmatrix} -1 \\ -1 \\ 0 \\ 2 \end{bmatrix}$$

Von zur Gathen and Sieveking, AMS (72)1 1978, showed that if an integer solution exists for $C\vec{x} \geq \vec{d}$, then one exists such that $|w| \leq (n+1)M, \ldots, |z| \leq (n+1)M$ where

- n is the number of variables;
- M is an upper bound on the absolute value of the 1×1-, 2×2-, 3×3- and 4×4-sub-determinants of $[C\vec{d}]$.

An upper bound on the upper bound M

Consider the augmented matrix and two sub-matrices A and B:

$$[C\vec{d}] = \begin{bmatrix} -3 & 0 & 2 & 0 & -1 \\ 0 & -1 & 1 & 0 & -1 \\ 0 & -1 & 0 & 1 & 0 \\ 0 & 1 & 0 & -1 & 2 \end{bmatrix} \quad A = \begin{bmatrix} 1 & 0 \\ -1 & 2 \end{bmatrix} \quad B = \begin{bmatrix} -3 & 0 & 2 \\ 0 & -1 & 1 \\ 0 & -1 & 0 \end{bmatrix}$$

Then

$$det(A) = 2 \qquad \begin{aligned} \|\langle 1, 0\rangle\| &= \sqrt{1^2 + 0^2} = 1 \\ \|\langle -1, 2\rangle\| &= \sqrt{(-1)^2 + 2^2} = \sqrt{5} \end{aligned} \qquad det(A) \leq \lfloor 1.\sqrt{5} \rfloor = 2$$

$$det(B) = 3 \qquad \begin{aligned} \|\langle -3, 0, 2\rangle\| &= \sqrt{13} \\ \|\langle 0, -1, 1\rangle\| &= \sqrt{2} \\ \|\langle 0, -1, 0\rangle\| &= 1 \end{aligned} \qquad det(B) \leq \lfloor \sqrt{26} \rfloor = 5$$

Questions, questions...

- Can a good upper bound on M be found in $O(nm)$ where again $n = var(S_1)$ and $m = |S_1|$?
- How does the structure of S_1 effect the pruning steps?