

Constraints modulo Presburger Arithmetic

Philipp Rümmer
philipp@cs.chalmers.se

6th June 2006

- KeY uses free-variables + constraint techniques (techniques from pure first-order proving)
- Unsolved problem: Combination with arithmetic/theories
- Most theorem provers with built-in theories do not use free variables at all

- One possible solution is shown here:
More powerful constraint language

- Work in progress: I am very interested in comments!

Joint work with Muhammad Ali Shah

Constraints in Sequent Calculus of KeY

$$\frac{\frac{\frac{\vdash X = c, X = d}{\vdash X = c \vee X = d}}{\vdash (X = c \vee X = d) \wedge f(c) = f(X)}}{\vdash \exists x. ((x = c \vee x = d) \wedge f(c) = f(x))}$$

$$\frac{\frac{\frac{\vdash X = c, X = d}{\vdash X = c \vee X = d} \quad \vdash f(c) = f(X)}{\vdash (X = c \vee X = d) \wedge f(c) = f(X)}}{\vdash \exists x. ((x = c \vee x = d) \wedge f(c) = f(x))}$$

- Problem: How to substitute X in the right way?

Constraints in Sequent Calculus of KeY

$$\frac{\frac{\frac{\vdash X = c, X = d}{\vdash X = c \vee X = d} \quad \vdash f(c) = f(X)}{\vdash (X = c \vee X = d) \wedge f(c) = f(X)}}{\vdash \exists x. ((x = c \vee x = d) \wedge f(c) = f(x))}$$

- Problem: How to substitute X in the right way?
- Solution in KeY (developed by Martin Giese):
Collect substitution candidates as **Constraints**
- For pure FOL: Unification constraints (\equiv , \wedge)

Constraints in Sequent Calculus of KeY

$$\frac{\frac{[X \equiv c], [X \equiv d]}{\vdash X = c, X = d}}{\vdash X = c \vee X = d} \quad \frac{[f(c) \equiv f(X)]}{\vdash f(c) = f(X)}$$
$$\frac{\vdash (X = c \vee X = d) \wedge f(c) = f(X)}{\vdash \exists x. ((x = c \vee x = d) \wedge f(c) = f(x))}$$

- Problem: How to substitute X in the right way?
- Solution in KeY (developed by Martin Giese):
Collect substitution candidates as **Constraints**
- For pure FOL: Unification constraints (\equiv, \wedge)

Constraints in Sequent Calculus of KeY

$$\frac{\frac{[X \equiv c], [X \equiv d]}{\vdash X = c, X = d}}{\vdash X = c \vee X = d} \quad \frac{[f(c) \equiv f(X)]}{\vdash f(c) = f(X)}$$
$$\frac{\vdash (X = c \vee X = d) \wedge f(c) = f(X)}{\vdash \exists x. ((x = c \vee x = d) \wedge f(c) = f(x))}$$

- Problem: How to substitute X in the right way?
- Solution in KeY (developed by Martin Giese):
Collect substitution candidates as **Constraints**
- For pure FOL: Unification constraints (\equiv , \wedge)
- To close proof, compatible constraints have to be found for all branches (conjunction is satisfiable):

$$X \equiv c \wedge f(c) \equiv f(X) \quad X \equiv d \wedge f(c) \equiv f(X)$$

- Constraint propagation, caching to improve efficiency

Problem 1 with Metavariables: Interpreted Symbols

- In program verification/KeY:
“Theories” are involved (arithmetic, etc.)
- Terms expressing same value can have different shape

$$c + X \equiv d + c \quad \text{theory of arithmetic}$$

- Either ... Try to normalise terms before unification
(partially done by strategies now available in KeY)
- Or ... Unification has to know about interpreted symbols
→ Unification modulo, done here

- Unification constraints are insufficient:

$$\frac{\frac{\vdash X > 0 \quad \vdash X < 10}{\vdash X > 0 \wedge X < 10}}{\vdash \exists x. (x > 0 \wedge x < 10)}$$

Problem 2 with Metavariables: Inequations

- Unification constraints are insufficient:

$$\frac{\frac{[X \equiv 1], [X \equiv 2], \dots}{\vdash X > 0} \quad \frac{[X \equiv 9], [X \equiv 8], \dots}{\vdash X < 10}}{\vdash X > 0 \wedge X < 10}}{\vdash \exists x. (x > 0 \wedge x < 10)}$$

Problem 2 with Metavariables: Inequations

- Unification constraints are insufficient:

$$\frac{\frac{[X \equiv 1], [X \equiv 2], \dots}{\vdash X > 0} \quad \frac{[X \equiv 9], [X \equiv 8], \dots}{\vdash X < 10}}{\vdash X > 0 \wedge X < 10}}{\vdash \exists x. (x > 0 \wedge x < 10)}$$

- Probably: Backtracking would work better here

A Different Solution . . .
Improve the Constraint Language

“Linear Unification Constraints” (LUC)

- Unification modulo theory (here: Presburger Arithmetic)
(Related to Deduction Modulo)
- Constraints with further predicates (\neq , $<$, \leq)

“Linear Unification Constraints” (LUC)

- Unification modulo theory (here: Presburger Arithmetic)
(Related to Deduction Modulo)
- Constraints with further predicates (\neq , $<$, \leq)

$$\langle \text{LUC} \rangle ::= \langle \text{atom} \rangle \left(\wedge \langle \text{atom} \rangle \right)^*$$

$$\begin{aligned} \langle \text{atom} \rangle ::= & \langle \text{term} \rangle = \langle \text{term} \rangle \mid \\ & \langle \text{intTerm} \rangle < \langle \text{intTerm} \rangle \mid \\ & \langle \text{intTerm} \rangle \leq \langle \text{intTerm} \rangle \mid \\ & \langle \text{intTerm} \rangle \neq \langle \text{intTerm} \rangle \end{aligned}$$

Examples of Linear Unification Constraints

$$C_1 := X = 5 \wedge 2 = Y + 1$$

$$C_3 := f(1 + 2) = f(3)$$

$$C_5 := X < f(X)$$

$$C_6 := f(X) < f(Y)$$

$$C_7 := f(X) \leq f(Y) \wedge X \neq Y$$

Examples of Linear Unification Constraints

$$C_1 := X = 5 \wedge 2 = Y + 1$$

$$C_3 := f(1 + 2) = f(3)$$

$$C_5 := X < f(X)$$

$$C_6 := f(X) < f(Y)$$

$$C_7 := f(X) \leq f(Y) \wedge X \neq Y$$

Which of the constraints are satisfiable?

Examples of Linear Unification Constraints

$$C_1 := X = 5 \wedge 2 = Y + 1$$

$$C_3 := f(1 + 2) = f(3)$$

$$C_5 := X < f(X)$$

$$C_6 := f(X) < f(Y)$$

$$C_7 := f(X) \leq f(Y) \wedge X \neq Y$$

Which of the constraints are satisfiable?

Definition

An LUC C is called *satisfiable* if

- for each structure
(meaning of uninterpreted function symbols)
- there is a variable assignment

such that C is evaluated to tt.

Examples of Linear Unification Constraints

$$C_1 := X = 5 \wedge 2 = Y + 1 \quad \text{Satisfiable}$$

$$C_3 := f(1 + 2) = f(3)$$

$$C_5 := X < f(X)$$

$$C_6 := f(X) < f(Y)$$

$$C_7 := f(X) \leq f(Y) \wedge X \neq Y$$

Which of the constraints are satisfiable?

Definition

An LUC C is called *satisfiable* if

- for each structure
(meaning of uninterpreted function symbols)
- there is a variable assignment

such that C is evaluated to tt.

Examples of Linear Unification Constraints

$$C_1 := X = 5 \wedge 2 = Y + 1 \quad \text{Satisfiable}$$

$$C_3 := f(1 + 2) = f(3) \quad \text{Satisfiable}$$

$$C_5 := X < f(X)$$

$$C_6 := f(X) < f(Y)$$

$$C_7 := f(X) \leq f(Y) \wedge X \neq Y$$

Which of the constraints are satisfiable?

Definition

An LUC C is called *satisfiable* if

- for each structure
(meaning of uninterpreted function symbols)
- there is a variable assignment

such that C is evaluated to tt.

Examples of Linear Unification Constraints

$$C_1 := X = 5 \wedge 2 = Y + 1$$

Satisfiable

$$C_3 := f(1 + 2) = f(3)$$

Satisfiable

$$C_5 := X < f(X)$$

Choose $f = \lambda x.x$

$$C_6 := f(X) < f(Y)$$

$$C_7 := f(X) \leq f(Y) \wedge X \neq Y$$

Which of the constraints are satisfiable?

Definition

An LUC C is called *satisfiable* if

- for each structure
(meaning of uninterpreted function symbols)
- there is a variable assignment

such that C is evaluated to tt.

Examples of Linear Unification Constraints

$C_1 := X = 5 \wedge 2 = Y + 1$	Satisfiable
$C_3 := f(1 + 2) = f(3)$	Satisfiable
$C_5 := X < f(X)$	Choose $f = \lambda x.x$
$C_6 := f(X) < f(Y)$	Choose $f = \lambda x.c$
$C_7 := f(X) \leq f(Y) \wedge X \neq Y$	

Which of the constraints are satisfiable?

Definition

An LUC C is called *satisfiable* if

- for each structure
(meaning of uninterpreted function symbols)
- there is a variable assignment

such that C is evaluated to tt.

Examples of Linear Unification Constraints

$C_1 := X = 5 \wedge 2 = Y + 1$	Satisfiable
$C_3 := f(1 + 2) = f(3)$	Satisfiable
$C_5 := X < f(X)$	Choose $f = \lambda x.x$
$C_6 := f(X) < f(Y)$	Choose $f = \lambda x.c$
$C_7 := f(X) \leq f(Y) \wedge X \neq Y$	Satisfiable

Which of the constraints are satisfiable?

Definition

An LUC C is called *satisfiable* if

- for each structure
(meaning of uninterpreted function symbols)
- there is a variable assignment

such that C is evaluated to tt.

- More constraints are satisfiable:

$$\frac{\frac{[f(X + 1) \equiv f(c)]}{\vdash f(X + 1) = f(c)}}{\vdash \exists x. f(x + 1) = f(c)}$$

- Theories: Uninterpreted functions + Arithmetic

Example: Closing Proofs using LUCs

- Constraints can be created for inequations:

$$\frac{\frac{[0 < X]}{\vdash X > 0} \quad \frac{[X < 10]}{\vdash X < 10}}{\vdash X > 0 \wedge X < 10}}{\vdash \exists x. (x > 0 \wedge x < 10)}$$

- Satisfiable constraint:

$$0 < X \wedge X < 10$$

Geometrically: How Solutions are Enumerated

- LUCs work best for linear arithmetic
- Non-linear situation:

$$\frac{\vdash Y \geq X^2}{\vdots}$$

Geometrically: How Solutions are Enumerated

- LUCs work best for linear arithmetic
- Non-linear situation:

$$\frac{[X \equiv 0 \wedge Y \equiv 0], [X \equiv -1 \wedge Y \equiv 1], [X \equiv 0 \wedge Y \equiv 1], \dots}{\vdash Y \geq X^2}$$

⋮

- With unification constraints:
Solutions of sequents as enumeration of points

Geometrically: How Solutions are Enumerated

- LUCs work best for linear arithmetic
- Non-linear situation:

$$\frac{[X = 0 \wedge Y \geq 0], [-1 \leq X \leq 1 \wedge Y \geq 1], \dots}{\frac{\vdash Y \geq X^2}{\vdots}}$$

- With unification constraints:
Solutions of sequents as enumeration of points
- With LUCs:
Solutions represented as disjunction of convex polytopes

Geometrically: How Solutions are Enumerated

- LUCs work best for linear arithmetic
- Non-linear situation:

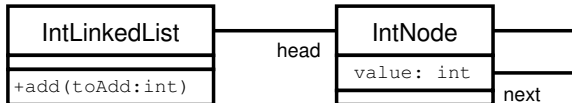
$$\frac{[X = 0 \wedge Y \geq 0], [-1 \leq X \leq 1 \wedge Y \geq 1], \dots}{\vdash Y \geq X^2}$$

⋮

- With unification constraints:
Solutions of sequents as enumeration of points
- With LUCs:
Solutions represented as disjunction of convex polytopes
- Both enumerations are infinite

Example: Disproving with LUCs

- Experimental implementation (incomplete)
- Counterexample for buggy program operating on lists:

$$HEAD = cons(HEAD_0, \dots)$$
$$NEXT = cons(NEXT_0, \dots)$$
$$VALUE = cons(VALUE_0, \dots)$$
$$KINTLINKEDLISTS > 0$$
$$KINTNODES > 0$$
$$OINDEX = 0$$
$$HEAD_0 = 0$$
$$NEXT_0 \geq KINTNODES \quad (= null)$$
$$VALUE_0 \neq TOADD$$


- Experiments are promising, but:
- Unknown so far: Decidability of satisfiability
- Complexity?
- Satisfiability tests that work well in practice?

Future Directions:

- Understand better how LUCs can be integrated in provers
- Investigate subsumption and normalisation issues for LUCs
- Further theories, for instance: lists, sets
- Master's thesis of Henrik Johansson will be continuation of work shown here