Leveraging Linear and Mixed Integer Programming for SMT

Tim King[1] Clark Barrett[1] Cesare Tinelli[2]

[1]New York University

[2]The University of Iowa

July 18, 2014

Big Ideas

- Call a floating point LP/MIP solver (GLPK) from CVC4
- Focus on hard problems
- Technique 1: Reseed a Simplex solver
- Technique 2: Replay an MIP proof
- Great on some families and not so great on others

Table of Contents

Decision Procedure for QF_LRA

Quantifier Free Linear Real Arithmetic

Is there a satisfying assignment, $a : \mathcal{X} \to \mathbb{R}$, that makes,

$$\begin{array}{rcrcl} x & + & y & \geq & 1 \\ x & - & y & \geq & 0 \\ 4x & - & y & \leq & 2 \end{array}$$

evaluate to true?

Decision Procedure for QF_LRA

Quantifier Free Linear Real Arithmetic

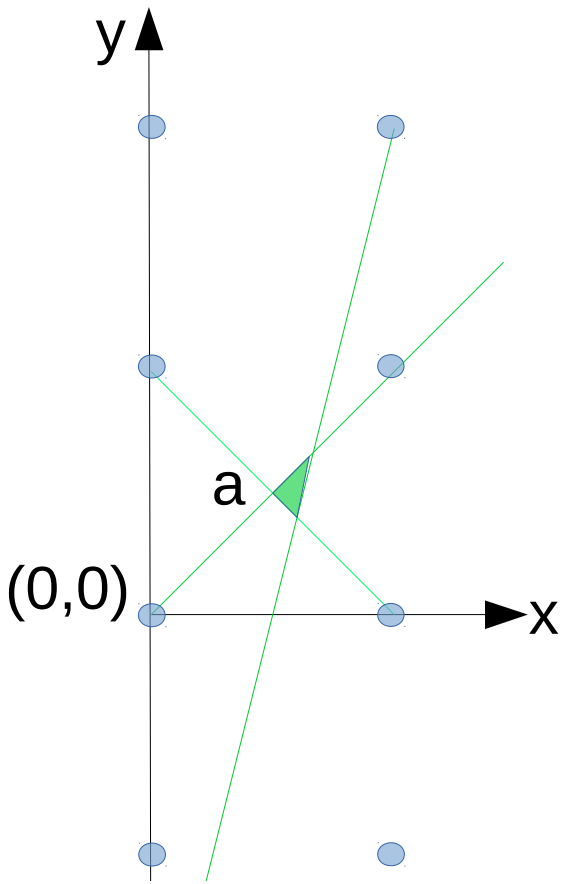
Is there a satisfying assignment, $a : \mathcal{X} \to \mathbb{R}$, that makes,

$$\begin{array}{rcrcl} x & + & y & \geq & 1 \\ x & - & y & \geq & 0 \\ 4x & - & y & \leq & 2 \end{array}$$

evaluate to true?

$$\begin{bmatrix} a_x \\ a_y \end{bmatrix} = \begin{bmatrix} \frac{1}{2} \\ \frac{1}{2} \end{bmatrix}$$

Visually



$$\begin{aligned} x &+ y \geq 1 \\ x &- y \geq 0 \\ 4x &- y \leq 2 \end{aligned}$$

$$\begin{bmatrix} a_x \\ a_y \end{bmatrix} = \begin{bmatrix} \frac{1}{2} \\ \frac{1}{2} \end{bmatrix}$$

Preprocessing

- Introduce a fresh s_i for each $\sum T_{i,j} \cdot x_j$
- Literals are of the form:

$$\bigwedge \left(s_i = \sum_{x_j} T_{i,j} \cdot x_j \right) \wedge \bigwedge l_i \leq x_i \leq u_i$$

  and s_i appears in exactly 1 equality.

- Collect into:

$$T\vec{x} = 0 \qquad \vec{l} \leq \vec{x} \leq \vec{u}$$

Preprocessing

- Introduce a fresh s_i for each $\sum T_{i,j} \cdot x_j$
- Literals are of the form:

$$\bigwedge \left(s_i = \sum_{x_j} T_{i,j} \cdot x_j \right) \wedge \bigwedge l_i \leq x_i \leq u_i$$

  and s_i appears in exactly 1 equality.

- Collect into:

$$T\vec{x} = 0 \qquad \vec{l} \leq \vec{x} \leq \vec{u}$$

Preprocessing

- Introduce a fresh s_i for each $\sum T_{i,j} \cdot x_j$
- Literals are of the form:

$$\bigwedge \left(s_i = \sum_{x_j} T_{i,j} \cdot x_j \right) \wedge \bigwedge l_i \leq x_i \leq u_i$$

  and s_i appears in exactly 1 equality.

- Collect into:

$$T\vec{x} = 0 \qquad \vec{l} \leq \vec{x} \leq \vec{u}$$

Basic, Nonbasic, & Tableau

- Every row in T is solved for a variable x_i

$$x_i = \sum_{x_j \in \mathcal{N}} T_{i,j} x_j$$

- Not solved for variables are **nonbasic** ($x_j \in \mathcal{N}$)
- Set of solved for variables are **basic** ($x_i \in \mathcal{B}$)

Pivoting the Tableau & Updating the Assignment

- **Pivoting** x_i for x_j solve x_i's row for x_j and "substitute" out x_j from the other rows

$$x_i = T_{i,j}x_j + \sum_{x_k \in \mathcal{N}} T_{i,k}x_k \quad \Longrightarrow \quad x_j = \frac{1}{T_{i,j}}x_i + \sum_{x_k \in \mathcal{N}} \frac{T_{i,k}}{T_{i,j}}x_k$$

- Invariant: $T\vec{a} = 0$

- **Update** the assignment of nonbasic x_j to α
  if we also update assignment of the dependent basic variables

Tableau Example

$$\begin{array}{rcrcl} x & + & y & \geq & 1 \\ x & - & y & \geq & 0 \\ 4x & - & y & \leq & 2 \end{array}$$

Tableau Example

$$T\vec{\mathcal{X}} = 0 \qquad \text{is equivalent to} \qquad \begin{array}{rcrcr} s_1 & = & x & + & y \\ s_2 & = & x & - & y \\ s_3 & = & 4x & + & y \end{array}$$

$$s_1 \geq 1 \wedge s_2 \geq 0 \wedge s_3 \leq 2$$

$$\mathcal{B} = \{s_1, s_2, s_3\}, \mathcal{N} = \{x, y\}$$

Result of applying Simplex

1. Starting from $a_x = a_y = 0$.
2. Pivot s_1 with y. Update a_{s_1} to 1
3. Pivot s_2 with x. Update a_{s_2} to 0

$$\begin{aligned} y &= \tfrac{1}{2}s_1 - \tfrac{1}{2}s_2 \\ x &= \tfrac{1}{2}s_1 + \tfrac{1}{2}s_2 \\ s_3 &= \tfrac{3}{2}s_1 + \tfrac{5}{2}s_2 \end{aligned} \qquad \begin{bmatrix} a_x \\ a_y \\ a_{s_1} \\ a_{s_2} \\ a_{s_3} \end{bmatrix} = \begin{bmatrix} \frac{1}{2} \\ \frac{1}{2} \\ 1 \\ 0 \\ \frac{3}{2} \end{bmatrix}$$

Simplex for DPLL(T) [DdM06]

procedure SimplexDPLL

while $x_i \in \mathcal{B}$ s.t. $a_i > u_i$ or ... **do**

select some $x_i = \sum T_{i,j} \cdot x_j$ s.t. $a_i > u_i$

if $\sum T_{i,j} \cdot x_j$ is at a minimum under a **then**

return a row conflict

else

select some x_j in $\sum T_{i,j} \cdot x_j$

Pivot x_i with x_j

Update assignment of x_i to u_i

Simplex for DPLL(T) [DdM06]

procedure SIMPLEXDPLL

while $x_i \in \mathcal{B}$ s.t. $a_i > u_i$ or ... **do**

select some $x_i = \sum T_{i,j} \cdot x_j$ s.t. $a_i > u_i$

if $\sum T_{i,j} \cdot x_j$ is at a minimum under a **then**

return a row conflict

else

select some x_j in $\sum T_{i,j} \cdot x_j$

PIVOT x_i with x_j ▷ $O(|T|)$

Update assignment of x_i to u_i

Simplex for DPLL(T) : Key Observations

- Assuming $a_i > u_i$, if

$$\forall T_{i,j} > 0.\ a_j = l_j \quad \text{and} \quad \forall T_{i,j} < 0.\ a_j = u_j$$

  then the bounds on the variables on the row are in conflict

$$\{x_j \geq l_j | T_{i,j} > 0\} \cup \{x_j \geq u_i | T_{i,j} < 0\} \cup \{x_i \leq u_i\}$$

- Simplex "likes" assignments that are against bounds
- Pivoting is expensive
- 90% of checks need 0 or 1 pivots [KBD13]

Table of Contents

General Approach

- Call an external off-the-shelf **untrusted** Simplex LP solver
- Reseed the state of the exact precision solver
- Only when it is likely to help
- Implemented with GLPK

Reseeding the Simplex State

If the real relaxation is hard, try the following:

1. Construct an approximate problem from exact

$$T\vec{\mathcal{X}} = 0, \vec{l} \leq \vec{\mathcal{X}} \leq \vec{u} \quad \Longrightarrow \quad \widetilde{T}\vec{\mathcal{X}} = 0, \vec{\widetilde{l}} \leq \vec{\mathcal{X}} \leq \vec{\widetilde{u}}$$

2. Call *untrusted floating point* Simplex solver on $\widetilde{T}, \widetilde{l}, \widetilde{u}$
3. Get back an approximate $\widetilde{a}$ and $\widetilde{\mathcal{B}}$
4. Convert floating point $\widetilde{a}$ into $a^{massage}$ $(\mathcal{X} \to \mathbb{Q})$
5. $\textsc{Reseed}(a^{massage}, \widetilde{\mathcal{B}})$ to get a new a and T
6. Call exact precision Simplex

Massaging Assignments

- Suppose we directly attempted to use $\widetilde{a}$.
- Each row must satisfy:

$$a_i = \sum T_{i,j} a_j$$

- Many variables have assignments near the bounds
- Many slack variables are entailed to be 0 (in practice)
- Get in a Simplex "friendly" state

Massaging Assignments

Floats to Rationals

$r \leftarrow \textsc{DioApprox}(\widetilde{a}_i, D)$

if $|r - a_i| \leq \epsilon$ **then** $r \leftarrow a_i$

if $x \in \mathcal{X}_{\mathbb{Z}}$ and $|r - \lfloor r \rceil| \leq \epsilon$ **then** $r \leftarrow \lfloor r \rceil$

if $r > u_i$ or $|r - u_i| \leq \epsilon$ **then** $r \leftarrow u_i$

else if $r < l_i$ or $|r - l_i| \leq \epsilon$ **then** $r \leftarrow l_i$

$a_i^{massage} \leftarrow r$

Reseeding Simplex ($a^{massage}, \widetilde{\mathcal{B}}$)

1. Update a_j to $a_j^{massage}$ for all $x_j \in \mathcal{N}$
2. $\mathcal{B}' \leftarrow \mathcal{N} \cap \widetilde{\mathcal{B}}$
3. If T has a row conflict, return **Unsat**
4. If all variables satisfy their bounds, return (**Sat**)
5. If $\neg \left(\exists i\ k.\ x_k \in \mathcal{B}' \wedge x_i \notin \widetilde{\mathcal{B}} \wedge T_{i,k} \neq 0 \right)$, return **Unknown**
   $\widetilde{\mathcal{B}}$ is not valid basis
6. Otherwise, PIVOT x_i with x_j, and update a_i to $a_i^{massage}$
7. If $\mathcal{B}' \neq \varnothing$, goto (3)
8. Otherwise, **Unknown** (call Simplex)

Reseeding Simplex

Related work

- More robust with Sum-of-Infeasibilities Simplex [KBD13]
- **ForcedPivot procedure via Simplex [CBdOM12, Mon09]**
- Check each conflict used in resolution at the end [FNORC08]

Table of Contents

From QF_LRA to QF_LIRA

- Partition variables $\mathcal{X}$ into $\mathcal{X}_{\mathbb{R}} \cup \mathcal{X}_{\mathbb{Z}}$
- a is **integer-compatible** if $\forall x_i \in \mathcal{X}_{\mathbb{Z}}$, then $a_i \in \mathbb{Z}$

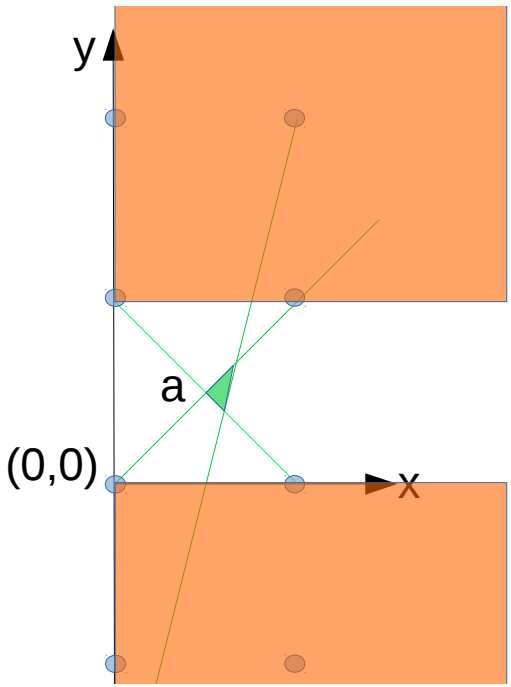
Branches and Cuts

- Branch: $x_i \leq \lfloor \alpha \rfloor \vee x_i \geq \lceil \alpha \rceil$ if $x_i \in \mathcal{X}_{\mathbb{Z}}$
- Cut: $\sum c_i x_j \geq d$ such that
  - $\{l_i\} \models_{\mathbb{RZ}} \sum c_j x_j \geq d$
  - $\{l_i\} \not\models_{\mathbb{R}} \sum c_j x_j \geq d$
  - $\{x_j = a_j\} \not\models \sum c_j x_j \geq d$ (*)

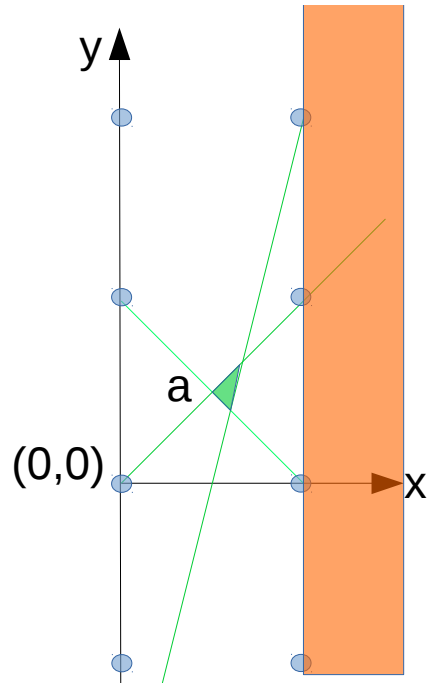
Branches and Cuts

Visually

Branch: $y \geq 1 \vee y \leq 0$



Cut: $\{\cdots\} \models_{\mathbb{RZ}} x \geq 1$



Branch-and-cut Solvers

Most SMT solvers and many MIP solvers

1. Treat all of $\mathcal{X}$ as if they were $\mathcal{X}_{\mathbb{R}}$
2. Solve this $\mathbb{R}$-*relaxation*
3. If unsat, return $\mathbb{R}$-conflict[s]
4. If $\mathbb{R}$-relaxation is (**Sat** a) and a is $\mathbb{Z}$-compatible, return a
5. [Heuristically] try to derive a cut.
   If successful, add the cut $\sum c_j x_j \geq d$, and goto (1)
6. Branch on some $x_i \in \mathcal{X}_{\mathbb{Z}}$ with $a_i \notin \mathbb{Z}$

Branch-and-cut Solvers

Most SMT solvers and many MIP solvers

1. Treat all of $\mathcal{X}$ as if they were $\mathcal{X}_{\mathbb{R}}$
2. Solve this $\mathbb{R}$-*relaxation*
3. If unsat, return $\mathbb{R}$-conflict[s]
4. If $\mathbb{R}$-relaxation is (**Sat** a) and a is $\mathbb{Z}$-compatible, return a
5. [Heuristically] try to derive a cut.
   If successful, add the cut $\sum c_j x_j \geq d$, and goto (1)
6. Branch on some $x_i \in \mathcal{X}_{\mathbb{Z}}$ with $a_i \notin \mathbb{Z}$
   Splitting-on-Demand in SMT

Answers for QF_LIA and QF_LIRA

- $\mathbb{R}$-infeasible
- $\mathbb{R}$-feasible and $\mathbb{Z}$-feasible
- $\mathbb{R}$-feasible and $\mathbb{Z}$-infeasible

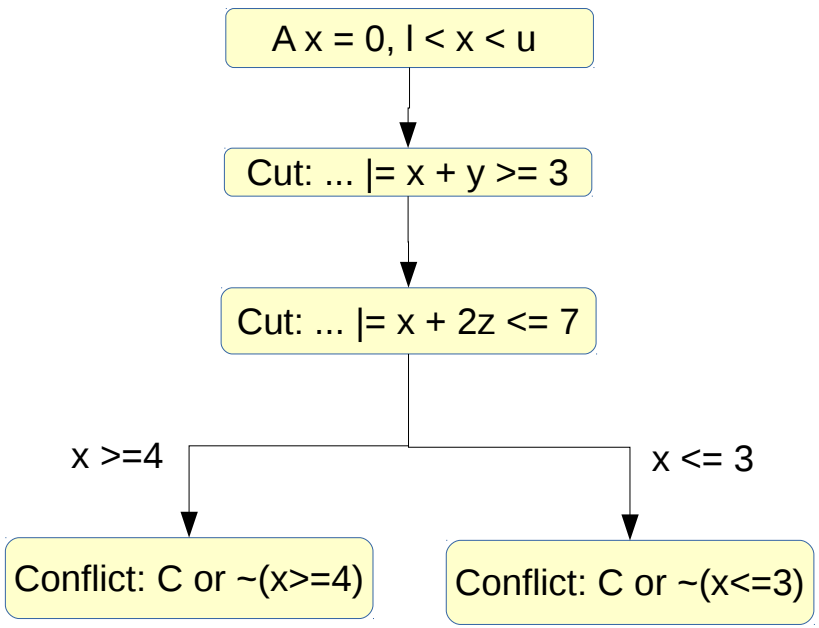
Answers for QF_LIA and QF_LIRA

- $\mathbb{R}$-infeasible
- $\mathbb{R}$-feasible and $\mathbb{Z}$-feasible
  Same reseeding trick as $\mathbb{R}$-feasible
- $\mathbb{R}$-feasible and $\mathbb{Z}$-infeasible

Answers for QF_LIA and QF_LIRA

- $\mathbb{R}$-infeasible
- $\mathbb{R}$-feasible and $\mathbb{Z}$-feasible
  Same reseeding trick as $\mathbb{R}$-feasible
- $\mathbb{R}$-feasible and $\mathbb{Z}$-infeasible

Infeasible Branch-and-Cut Executions



- Leaves are conflicts
- Internal nodes are branches

$$x_i \leq \lfloor \alpha \rfloor \vee x_i \geq \lceil \alpha \rceil \quad \text{if } x_i \in \mathcal{X}_{\mathbb{Z}}$$

- Nodes have cuts

$$\{l_i\} \models_{\mathbb{RZ}} \sum c_j x_j \geq d$$

Replaying the MIP Execution

- Minimizes changes to the MIP solver's search

Replaying the MIP Execution

- Minimizes changes to the MIP solver's search
- Instrument GLPK to print hints about:
  branch, unsat leaves, and derivations of cutting planes

Replaying the MIP Execution

- Minimizes changes to the MIP solver's search
- Instrument GLPK to print hints about:
  branch, unsat leaves, and derivations of cutting planes
- Repeat "the big steps" in the SMT solver

Replaying the MIP Execution

- Minimizes changes to the MIP solver's search
- Instrument GLPK to print hints about:
  branch, unsat leaves, and derivations of cutting planes
- Repeat "the big steps" in the SMT solver
- Reconstruct the Resolution+Cutting Planes proof
- Resolution removes branching literals

Replaying the MIP Execution

- Minimizes changes to the MIP solver's search
- Instrument GLPK to print hints about:
  branch, unsat leaves, and derivations of cutting planes
- Repeat "the big steps" in the SMT solver
- Reconstruct the Resolution+Cutting Planes proof
- Resolution removes branching literals
- Any failure can be safely dropped
- Success is a conflict

Cutting Planes

- Hint is used to instantiate a cutting plane procedure
- Proof must tightly match to get the "same" cut
- White-box knowledge and detailed hints
- Support for Gomory (easy) and MK-MIR (hard) cuts

Table of Contents

SMT Solver Comparison

QF_LRA

			+MIP		CVC4		yices2		mathsat5		Z3	
set	# inst.	# sel.	solved	time (s)	solved	time (s)	solved	time (s)	solved	time (s)	solved	time (s)
QF_LRA	634	634	627	6199	618	7721	620	5265	612	10814	615	5696
latendresse	18	18	18	129	10	44	12	85	10	99	0	0
miplib	42	37	30	1530	21	3037	23	2730	17	5682	18	2435
DTP-*	91	4	4	4	4	4	4	0	4	2	4	1
total	-	41	34	1534	25	3041	27	2330	21	5684	22	2436

(AR) = Applied any Replay technique, **K** = 1000

SMT Solver Comparison

QF_LIA ¬-conjunctive

			+MIP		CVC4		mathsat5		Z3		altergo	
set	# inst.	# sel.	solved	time (s)	solved	time (s)	solved	time (s)	solved	time (s)	solved	time (s)
everything												
QF_LIA	5882	5882	5738	97**K**	5540	117**K**	5697	88**K**	5513	94**K**	5188	264**K**
conjuncts	1303	1303	1249	11**K**	1068	31**K**	1154	33**K**	1039	19**K**	1232	2055
(AR) ¬ conjuntive												
convert	319	282	208	9646	193	9343	274	1876	282	118	166	272
bofill-*	652	460	460	5401	458	4490	460	1519	460	2060	67	55
CIRC	51	11	11	0	11	0	11	0	11	0	11	0
calypto	37	37	37	3	37	3	37	6	36	5	35	24
nec-smt	2780	207	207	17**K**	207	18**K**	207	17**K**	201	7209	184	23**K**
wisa	5	1	1	0	1	0	1	1	1	0	1	0
total	-	998	924	32**K**	907	31**K**	990	21**K**	991	9392	464	24**K**

AltErgo is using [BCC+12]

(AR) = Applied any Replay technique, **K** = 1000

SMT Solver Comparison

QF_LIA conjunctive

			+MIP		CVC4		mathsat5		Z3		altergo	
set	# inst.	# sel.	solved	time (s)	solved	time (s)	solved	time (s)	solved	time (s)	solved	time (s)
everything												
QF_LIA	5882	5882	5738	97**K**	5540	117**K**	5697	88**K**	5513	94**K**	5188	264**K**
conjuncts	1303	1303	1249	11**K**	1068	31**K**	1154	33**K**	1039	19**K**	1232	2055
(AR) conjuntive												
dillig	233	189	189	49	157	9823	188	7185	166	1269	189	5
miplib2003	16	8	4	307	4	1283	5	354	5	1089	0	0
prime-cone	37	37	37	2	37	2	37	1	37	2	37	1
slacks	233	188	166	61	93	2003	119	4741	90	1994	188	84
CAV_2009	591	424	424	69	346	10**K**	421	10**K**	354	2759	423	323
cut_lem.	93	74	62	9581	64	6865	45	9472	38	5858	74	267
total	-	920	882	10**K**	701	30**K**	815	31**K**	690	12**K**	911	680

(AR) = Applied any Replay technique, **K** = 1000

Comparison with conjunctive solvers

			+MIP		cutsat		scip		glpk	
set	# inst.	# sel.	solved	time (s)	solved	time (s)	solved	time (s)	solved	time (s)
conjuncts	1303	1303	1249	11130	1018	35330	1255	7164	1173	8895
(AR) conjuntive										
dillig	233	189	189	49	166	5840	189	42	189	3
miplib2003	16	8	4	307	6	146	7	17	6	295
prime-cone	37	37	37	2	37	4	37	1	37	0
slacks	233	188	166	61	96	6324	161	2361	101	11
CAV_2009	591	424	424	69	377	17015	424	105	424	6
cut_lemmas	93	74	62	9581	15	1887	72	1757	71	760
total	-	920	882	10069	697	31216	890	4283	828	1075

(AR) = Applied any Replay technique, $\mathbf{K} = 1000$

QF_LIA Reseed and Replay success rates

			Sat replay		**Unsat** replay	
set	# inst.	solve int calls	attempts	successes	attempts	successes
QF_LIA	1806	3873	2559	1058	652	425
convert	208	2130	1356	1	178	3
bofill-scheduling	460	254	245	245	0	0
CIRC	11	85	6	5	79	77
calypto	37	375	77	23	293	278
wisa	1	1	1	1	0	0
dillig	189	228	225	185	3	2
miplib2003	4	10	3	3	5	4
prime-cone	37	37	19	19	18	18
slacks	166	195	168	162	3	3
CAV_2009	424	469	459	414	8	7
cut_lemmas	62	89	0	0	65	33

Only includes solved instances

What happened on the `convert` family?

- MIP solver is wrong about feasibility too often
- Variables are in bounds up to a "dual gap"
  - Intuitively: Let a_i violate u_i by a litle where little is scaled by the size of the numbers
  - Numerically stability of floating points
- Gap is too large for `QF_LIA` bit-extracts for $\sim m + n > 40$

$$x = 2^m y + z \wedge z \in [0, 2^m), y \in [0, 2^n), x \in [0, 2^{m+n})$$

- Decreasing the maximum gap leads to cycling in practice
- Need bigger floating point numbers if MIP solver is to work

Table of Contents

Future Work

- What else can we do with an MIP solver?
- Different heuristics for cuts?
- Logging and replaying approximate Farkas's lemma instances [NS04]
- k-precision floating Simplex solver for SMT (1-2years?) [CKSW13]

In Summary

- Integrated a floating point LP/MIP solver (GLPK) into CVC4 (Backup. Not the main engine!)

In Summary

- Integrated a floating point LP/MIP solver (GLPK) into CVC4 (Backup. Not the main engine!)
- Reseeding Simplex (1 week to implement[*])
  - Gives candidate models and gives real relaxation conflicts
  - Massaging floating points is really important

In Summary

- Integrated a floating point LP/MIP solver (GLPK) into CVC4 (Backup. Not the main engine!)
- Reseeding Simplex (1 week to implement[*])
  - Gives candidate models and gives real relaxation conflicts
  - Massaging floating points is really important
- Replaying MIP conflicts (significantly more effort)
  MIP must be white-box and must log proofs!

In Summary

- Integrated a floating point LP/MIP solver (GLPK) into CVC4 (Backup. Not the main engine!)
- Reseeding Simplex (1 week to implement[*])
  - Gives candidate models and gives real relaxation conflicts
  - Massaging floating points is really important
- Replaying MIP conflicts (significantly more effort)
  MIP must be white-box and must log proofs!
- Overall performance is good, but there are known problems

In Summary

- Integrated a floating point LP/MIP solver (GLPK) into CVC4 (Backup. Not the main engine!)
- Reseeding Simplex (1 week to implement[*])
  - Gives candidate models and gives real relaxation conflicts
  - Massaging floating points is really important
- Replaying MIP conflicts (significantly more effort)
  MIP must be white-box and must log proofs!
- Overall performance is good, but there are known problems

Thank you for your attention

References I

François Bobot, Sylvain Conchon, Évelyne Contejean, Mohamed Iguernelala, Assia Mahboubi, Alain Mebsout, and Guillaume Melquiond, *A Simplex-based extension of Fourier-Motzkin for solving linear integer arithmetic*, IJCAR 2012: Proceedings of the 6th International Joint Conference on Automated Reasoning (Manchester, UK) (Bernhard Gramlich, Dale Miller, and Ulrike Sattler, eds.), Lecture Notes in Computer Science, vol. 7364, Springer, June 2012, pp. 67–81.

Diego Caminha Barbosa de Oliveira and David Monniaux, *Experiments on the feasibility of using a floating-point simplex in an SMT solver*, Workshop on Practical Aspects of Automated Reasoning (PAAR), CEUR Workshop Proceedings, 2012.

References II

William Cook, Thorsten Koch, Daniel E. Steffy, and Kati Wolter, *A hybrid branch-and-bound approach for exact rational mixed-integer programming*, Math. Program. Comput. **5** (2013), no. 3, 305–344.

Bruno Dutertre and Leonardo de Moura, *Integrating Simplex with DPLL(T)*, Tech. Report SRI-CSL-06-01, Computer Science Laboratory, SRI International, May 2006.

Germain Faure, Robert Nieuwenhuis, Albert Oliveras, and Enric Rodríguez-Carbonell, *Sat modulo the theory of linear arithmetic: Exact, inexact and commercial solvers*, SAT, 2008, pp. 77–90.

References III

Timothy King, Clark Barrett, and Bruno Dutertre, *Simplex with sum of infeasibilities for SMT*, Proceedings of the 13^{th} International Conference on Formal Methods In Computer-Aided Design (FMCAD '13), Lecture Notes in Computer Science, November 2013, pp. 189–196.

David Monniaux, *On using floating-point computations to help an exact linear arithmetic decision procedure*, Computer-aided verification (CAV), Lecture Notes in Computer Science, no. 5643, Springer-Verlag, 2009, pp. 570–583.

Arnold Neumaier and Oleg Shcherbina, *Safe bounds in linear and mixed-integer linear programming*, Mathematical Programming **99** (2004), no. 2, 283–296.

Appendix

Resolution Phase

The proof reconstruction phase uses the following heuristics:

- All up-branch conflicts are resolved with all down-branch conflicts
  (DP-style)
- Performed eager subsumption checking
  Pays for itself by keeping the set of conflicts small
- Non-chronological backtracks when possible
  (One branch has a conflict not involving its branch literal)