On using an inexact floating-point LP solver for deciding linear arithmetic in an SMT solver

Frédéric Besson*

Inria, Centre Rennes - Bretagne Atlantique
Campus Universitaire de Beaulieu
FR-35042 Rennes cedex

Abstract Off-the-shelf linear programming (LP) solvers trade soundness for speed: for efficiency, the arithmetic is not exact rational arithmetic but floating-point arithmetic. As a side-effect the results come without any formal guarantee and cannot be directly used for deciding linear arithmetic. In this work we explain how to design a sound procedure for linear arithmetic built upon an inexact floating-point LP solver. Our approach relies on linear programming duality to instruct a black-box off-the-shelf LP solver to output, when the problem is not satisfiable, an untrusted proof certificate. We present a heuristic post-processing of the certificate which accommodates for certain numeric inaccuracies. Upon success it returns a provably correct proof witness that can be independently checked. Our preliminary results are promising. For a benchmark suite extracted from SMT verification problems the floating-point LP solver returns a result for which proof witnesses are successfully and efficiently generated.

1 Introduction

Satisfiability Modulo Theories (SMT) consists in solving formulae belonging to a combination of theories [4] such as the theory of uninterpreted function symbols, arrays, bit-vectors or linear real arithmetic (LRA). In practise linear arithmetic formulae are pervasive. Therefore, to be competitive, SMT solvers need an efficient linear arithmetic decision procedure. For fragments of LRA there exist scalable algorithms. For instance, difference logic (constraints of the form $x - y \leq c$) is solved by variations of the shortest-path algorithm [12,2]. Deciding *full* LRA is more challenging. State-of-the-art SMT solvers, such as Yices and Z3, re-implement from scratch a Simplex solver [6,3] in exact rational arithmetic. In terms of engineering it would be much more cost-effective to simply use off-the-shelf state-of-the-art Simplex implementations.

Dutertre and De Moura [6] explain how to design an efficient Simplex algorithm that fulfills the requirements of SMT solvers. They put the emphasis on distinctive features that are key for efficiency *e.g.,* fast backtracking and theory propagation. Other important features are generation of conflict clauses and

* This work is partially funded by the ANR Decert project.

handling of strict inequalities. In their work there is one obvious and essential property that is only alluded to: *soundness*. From an SMT solver perspective, soundness is of course mandatory and not negotiable. On the contrary, scalable off-the-shelf Simplex implementations, *e.g.,* Glpk[10] and CPLex [8], are inexact because they use floating-point instead of exact rational arithmetic. When the Simplex method is used as a decision procedure for linear arithmetic, this is an issue because even the slightest approximation can be responsible for an unsound result.

Previous works [11,7] propose approaches allowing to leverage inexact floating-point Linear Programming (LP) solvers in an SMT solver. The present paper belongs to this trend. Like previous approaches we consider the LP solver as an untrusted oracle. Its output is therefore post-processed, using exact rational arithmetic, in order to obtain a proof witness.

The contribution of this work is to develop an approach exploiting *linear programming duality*. The benefit of a dual encoding is that it minimises the requirements on LP solvers. More precisely, we present a dual formulation of the original (*primal*) linear problem with the following properties: a) it has no strict inequalities; b) it is incremental; c) it generates conflict clauses; d) it allows for conflict-driven theory propagation. Such properties have engineering consequences and make it easier to interface an SMT solver with existing of-the-shelf exact rational arithmetic LP solvers that might neither be incremental nor support strict inequalities nor generate conflict clauses. The other contribution of the paper is a fast post-processing of the output of an inexact floating-point LP solver allowing to accommodate for numeric inaccuracy. Compared to previous approaches our post-processing is faster and always succeeds for a benchmark suite representative of SMT verifications problems.

The rest of the paper is organised as follows. Section 2 exploits linear duality in order to derive a notion of witness for LRA. Section 3 shows how a witness can be found by solving a single linear problem. Section 4 emphasises that the formulation of the problem is compatible with the requirements of SMT solvers. Section 5 presents an algorithm to reconstruct an exact result from an untrusted oracle obtained from a floating-point LP solver. The approach is validated experimentally in Section 6. Section 7 compares with related work and Section 8 concludes. Notations are fairly standard; for completeness, they are given in Appendix A. Detailed proofs can be found in Appendix B.

2 Witnesses for LRA

Our notion of witnesses for LRA originates from the duality theory of linear programming. The original problem is called the *primal* problem. Primal and dual are such that: if the primal is not satisfiable then the dual is satisfiable; and if the primal is satisfiable then the dual is unsatisfiable. As a result, to prove that a conjunctive LRA formula F is unsatisfiable, it suffices to exhibit a model w of the dual of F. The model w is a *witness* that F is unsatisfiable. In the following, we show how to derive witnesses from Motzkin's transposition

theorem, a duality theorem, which generalises Farkas' lemma in the presence of strict inequalities.

2.1 LRA formulae

Atoms of LRA formulae are linear constraints of the form $a_1x_1 + \ldots + a_nx_n \bowtie b$ where $a_1, \ldots, a_n$ and b are rational constants, $x_1, \ldots, x_n$ are real variables and $\bowtie$ belongs to the set $\{=, \leq, <\}$. (Other operators such as $\geq, >, \neq$ can easily be encoded.) In an SMT solver the SAT engine proposes boolean models of the formula that are fed into a decision procedure for the conjunctive fragment of the theory. Therefore, we only consider conjunctive formulae of LRA.

In the sequel we use matrix notations for reasoning about conjunctive LRA formulae. We write $A \cdot x < a$, $B \cdot x \leq b$, $C \cdot x = c$ for the LRA formula

$$\bigwedge_{i=1}^{m} A_{i,1}x_1 + \cdots + A_{i,j}x_j < a_i \wedge \bigwedge_{i=1}^{n} B_{i,1}x_1 + \cdots + B_{i,j}x_j \leq b_i \wedge \bigwedge_{i=1}^{o} C_{i,1}x_1 + \cdots + C_{i,j}x_j = c_i$$

2.2 Proof witnesses for LRA

In this section we derive a notion of proof witness for LRA from Motzkin's transposition theorem. For a comprehensive presentation and formal proofs of this result we refer the reader to [13, Chapter 7]. The notion of witness we come up with has the advantage of simplifying the interface with an SMT solver (see Section 4).

Lemma 1 (Motzkin's transposition theorem [13, Corollary 7.1k]). *Let $A \in \mathbb{Q}^{p \times n}$ and $B \in \mathbb{Q}^{q \times n}$ be matrices and $a \in \mathbb{Q}^n$ and $b \in \mathbb{Q}^n$ be vectors.*

$$\exists x, A \cdot x < a \wedge B \cdot x \leq b$$

if and only if

$$\forall (y, z), y \geq \mathbf{0} \wedge z \geq \mathbf{0} \Rightarrow \begin{cases} A^t \cdot y + B^t \cdot z = \mathbf{0} \Rightarrow a^t \cdot y + b^t \cdot z \geq 0 \\ A^t \cdot y + B^t \cdot z = \mathbf{0} \wedge \neg(y = \mathbf{0}) \Rightarrow a^t \cdot y + b^t \cdot z > 0 \end{cases}$$

Using Lemma 1 we define a notion of witness (Definition 1) which handles equalities, strict and non-strict inequalities. Equalities could be easily dealt with by rewriting an equality $a_1x_1 + \cdots + a_nx_n = b$ into two non-strict inequalities *i.e.,* $a_1x_1 + \cdots + a_nx_n \leq b$ and $(-a_1)x_1 + \cdots + (-a_n)x_n \leq (-b)$. Such an encoding would have the disadvantage of potentially doubling the size of the problem. Definition 1 keeps the size of the problem unchanged.

Definition 1 (*wit*). *A triple of vectors (y, z, t) is a witness that $A \cdot x < a$ and $B \cdot x \leq b$ and $C \cdot x = c$ is unsatisfiable (written $wit(A, a, B, b, C, c, (y, z, t))$) if the following conditions hold:*

i) $y \geq \mathbf{0}$, $z \geq \mathbf{0}$

ii) $A^t \cdot y + B^t \cdot z + C^t \cdot t = \mathbf{0}$
iii) $a^t \cdot y + b^t \cdot z + c^t \cdot t < 0$ *or* $\neg(y = \mathbf{0}) \wedge a^t \cdot y + b^t \cdot z + c^t \cdot t \leq 0$

Definition 1 provides a sound and complete notion of proof witnesses for conjunctions of formulae with strict and non-strict inequalities. These facts, direct consequence of Lemma 1, are established by Corollary 1 and Corollary 2. Their proofs can be found in Appendix B.

Corollary 1 (Soundness of *wit*). *Let A, B be matrices and a, b be vectors. We have* $\forall(y, z), wit(A, a, B, b, \mathbf{0}, \mathbf{0}, (y, z, \mathbf{0})) \Rightarrow \forall x, \neg(A \cdot x < a \wedge B \cdot x \leq b)$.

Corollary 2 (Completeness of *wit*). *Let A, B be matrices and a, b be vectors. We have* $\forall x, \neg(A \cdot x < a \wedge B \cdot x \leq b) \Rightarrow \exists(y, z), wit(A, a, B, b, \mathbf{0}, \mathbf{0}, (y, z, \mathbf{0}))$.

The soundness and completness results generalise to formulae with equations, strict and non-strict inequalities. The proof relies on the fact that equalities can be transformed into pairs of inequalities and that it is always possible to reconstruct a witness for the initial formula from a witness of the transformed formulae (and *vice-versa*).

Theorem 1. *Let A, B and C be matrices and a, b and c be vectors.*

$$\exists(y, z, t), wit(A, a, B, b, C, c, (y, z, t)) \text{ iff } \forall x, \neg(A \cdot x < a \wedge B \cdot x \leq b \wedge C \cdot x = c)$$

Theorem 1 establishes that Definition 1 provides a sound and complete notion of witness for LRA. The next section is about computing a witness by solving a *linear program*.

3 Witness linear program

We show how proof witnesses that a conjunctive LRA formula is not satisfiable can be computed by solving linear programs in the form accepted by off-the-shelf LP solvers such as Glpk [10] and CPLex [8]. To our knowledge, the formulation is original and has never been exploited by SMT solvers.

3.1 Linear programming

Linear programming consists in optimising a linear function under a conjunction of linear constraints. There exist several forms of linear programs that can be proved equivalent [13]. Available LP solvers usually take problems in the following standard form.

Definition 2. *Let $A \in \mathbb{Q}^{m,n}$, $c \in \mathbb{Q}^n$, $l \in (\mathbb{Q} \cup \{-\infty\})^n$ and $u \in (\mathbb{Q} \cup \{+\infty\})^n$. A linear program is written $max\{c^t \cdot x \mid A \cdot x = \mathbf{0}, l \leq x \leq u\}$ where 1) $c^t \cdot x$ is the objective function to optimise with respect to x; 2) A is the constraint matrix; 3) l is a lower bound for x; 4) u is an upper bound for x. If $l_i = -\infty$ (resp. $u_i = +\infty$) then x_i has no lower bound (resp. no upper bound).*

For our purpose, an LP solver is a black-box procedure for solving linear programs. The specification of an exact LP solver is as follows. If the linear program is not feasible, the LP solver detects it. If the problem is feasible and has an optimal value v, the LP solver returns v and a feasible vector x allowing to reach v *i.e,* $c^t \cdot x = v$. Finally, if the problem is feasible but the objective function has no upper bound, the LP solver reports that the linear program is unbounded.

3.2 Witnesses as linear programs

A witness for a conjunction of linear constraints (Definition 1) cannot be directly found by solving a linear program (Definition 2). The reasons are that the syntactic definition of witnesses i) makes use of strict inequalities and; ii) exhibits a disjunction. Definition 3 remedies these deficiencies and reinterpret a witness as the solution of a linear program.

Definition 3 (Witness Linear Program *wlp*). *Given* $A \cdot x < a$, $B \cdot x \leq b$ *and* $C \cdot x = c$ *the* witness linear program $wlp(A, a, B, b, C, c)$ *is defined by*

$$max \left\{ \mathbf{0}^t \cdot (y/z/t) \;\middle|\; \begin{array}{rll} & & \text{ⓓ} \quad \mathbf{0} \leq y \leq +\boldsymbol{\infty} \\ \text{ⓐ}\ A^t \cdot y + B^t \cdot z + C^t \cdot t & = \mathbf{0} & \text{ⓔ} \quad \mathbf{0} \leq z \leq +\boldsymbol{\infty} \\ \text{ⓑ}\ u + a^t \cdot y + b^t \cdot z + c^t \cdot t & = 0\,, & \text{ⓕ} \quad -\boldsymbol{\infty} \leq t \leq +\boldsymbol{\infty} \\ \text{ⓒ}\ v - u - \mathbf{1}^t \cdot y & = 0 & \text{ⓖ} \quad 0 \leq u \leq +\infty \\ & & \text{ⓗ} \quad 1 \leq v \leq +\infty \end{array} \right\}$$

For this linear program the objective function to optimise is the constant function 0. As a result, the optimisation problem is turned into a feasibility problem. Any other objective function would increase the running time. The constraint matrix and the lower and upper bounds of the linear problem encode the conditions (i), (ii) and (iii) of Definition 1. Equation ⓐ corresponds syntactically to condition (ii). The bound conditions ⓓ and ⓔ directly encode condition (i). The bound condition ⓕ states that the vector t is actually unconstrained. The remaining equations *i.e.,* ⓑ and ⓒ together with the remaining bound conditions *i.e.,* ⓖ and ⓗ are more intricate and encode condition (iii)

$$a^t \cdot y + b^t \cdot z + c^t \cdot t < 0 \text{ or } \neg(y = \mathbf{0}) \wedge a^t \cdot y + b^t \cdot z + c^t \cdot t \leq 0$$

In the linear problem, equation ⓑ and bound condition ⓖ constrain the quantity $a^t \cdot y + b^t \cdot z + c^t \cdot t$ to be smaller **or** equal to zero. From the definition of a witness the equality with zero is only allowed if $\neg y = 0$ *i.e.,* if there is at least one strict inequality for which y_i is strictly positive. This is catered for by the equation ⓒ and bound ⓗ . In particular, the term $\mathbf{1}^t \cdot y$ is strictly positive as soon as there exists $y_i > 0$ for some i.

Lemma 2 and Lemma 3 formalise the above discussion and show how to obtain a witness by solving the linear program *wlp*. Full proofs can be found in Appendix B.

Lemma 2 (Soundness of *wlp*). *If* $wlp(A, a, B, b, C, c)$ *has optimum* $(y/z/t/u/v)$ *then* $wit(A, a, B, b, C, c, (y, z, t))$ *holds.*

Lemma 3 (Completeness of *wlp*). *If* $wit(A, a, B, b, C, c, (y, z, t))$ *then there exists* $\alpha > 0$*,* u *and* v *such that* $wlp(A, a, B, b, C, c)$ *has optimum* $\alpha \cdot (y/z/t/u/v)$*.*

4 Witness linear program for SMT solvers

A SMT-compatible LP solver has to provide a number of extra-features [6] : strict inequalities, conflict clauses, theory propagation and backtracking. The advantage of the *witness linear program* encoding is that it makes almost *any* LP solver compatible with an SMT solver.

Strict inequalities are absent from our dual formulation. Because inexact solvers do not support them this is essential for not incurring an additional loss of precision. The encoding used by previous approaches (see [11, Section 3] and [7, Section 4]) are approximate and have the drawback of potentially changing the status of the problem. In an exact rational, strict inequalities are modeled by introducing a symbolic *infinitesimal parameter* [6, Section 5]. This complication is avoided by our dual formulation.

Conflict clauses are a by-product of the notion of witness. A zero entry of a witness w, say $w_i = 0$, means that the contradiction is independent from the i^{th} formula of the primal problem. In other words, the minimal conflict clause of a witness is obtained by collecting the formulae corresponding to the non-zero entries of the witness. Therefore, the LP solver does not need to support explanations or conflict clauses.

Theory propagation can be implemented on top of a core decision procedure. A more specific *conflict-driven* theory propagation can be obtained by inspection of a witness. If a witness vector allows to exhibit a contradiction *i.e.,* a conflict clause; a sub-vector of a witness can be interpreted as a logic consequence of the input formula *i.e.,* an intermediate clause used to establish the contradiction. Those clauses can be exported by theory propagation.

Example 1. Consider the following conjunction:

$$2x + y \leq 1 \wedge -x - y \leq -2 \wedge y \leq 1 \wedge 2y + z \leq 1$$

The vector $(1, 2, 1, 0)$ is a witness and the conflict clause is $\neg(2x+y \leq 1) \vee \neg(-x-y \leq 2) \vee \neg(y \leq 1)$. Conflict-driven theory propagation could also generate the clauses: $\neg(2x + y \leq 1) \vee \neg(-x - y \leq -2) \vee (-y \leq -3)$, $\neg(2x + y \leq 1) \vee \neg(y \leq 1) \vee (2x + 2y \leq 2)$ and $\neg(-x - y \leq -2) \vee \neg(y \leq 1) \vee (-2x - y \leq -3)$.

Backtracking is implemented by updating a bound condition of the linear program. Our dual approach is therefore incremental in the sense of Dutertre and de Moura [6, Section 3]. Removing the ith formula of the primal problem has the simple effect of updating the bound condition of the ith variable of the *wlp* such that $0 \leq x_i \leq 0$. The reverse operation consists in updating the bound

condition so that x_i is unbounded for equalities and positive for inequalities. To be incremental the Simplex method must return a final tableau obtained by pivoting the initial constraint matrix. This precludes certain pre-solving optimisations that would exploit a bound condition, say $0 \leq x \leq 0$, to simplify the constraint matrix. Pre-solving can usually be disabled. This is for instance the case for Glpk [10], the Simplex solver we use in our experiments.

5 From untrusted oracles to proof witnesses

In Section 3 we have shown that finding a witness could be done by solving the *witness linear program wlp*. In this section we propose an optimistic algorithm for finding a witness by post-processing the result of an inexact floating-point LP solver. In the following, we detail the algorithm given in Figure 1.

Require: $A \cdot x < a, B \cdot x \leq b, C \cdot c = c$
1: $lp \leftarrow wlp(A, a, B, b, C, c) = max\{\mathbf{0} \mid M \cdot y = \mathbf{0}, l \leq y \leq u\}$
2: $LPsolve(lp)$
3: **if** $status(lp) = not\_feasible$ **then return unknown** //probably sat
4: **if** $status(lp) = error$ **then return unknown**
5: $//status(lp) == optimal$
6: $r \leftarrow Solution(lp)$
7: $r \leftarrow \begin{cases} r_i & \text{if } l_i \leq r_i \leq u_i \\ l_i & \text{if } l_i \neq -\infty \\ u_i & \text{otherwise} \end{cases}$
8: $M'_{i,j} \leftarrow \begin{cases} M_{i,j} & \text{if } r_j \neq 0 \\ 0 & \text{otherwise} \end{cases}$
9: $U \leftarrow Echelon(M')$
10: $w \leftarrow Eval(U, r)$
11: **if** $l \leq w \leq u$ **then return unsat** // the witness is w
12: **return unknown** // probably unsat

Figure 1. Witness reconstruction algorithm

The witness linear program lp is constructed from the primal problem (line 1). We write M for the constraint matrix of lp and l and u for the bound conditions. The linear program lp is then solved using an inexact floating-point LP solver (line 2). If the status of the LP solver is *not_feasible* we conclude (without formal guarantee) that the primal problem is probably satisfiable (line 3). Though it never happens in our experiments, in general, the LP solver might also crash or report an error. In such cases, we would return that the status of the problem is unknown (line 4). The interesting case is when the LP solver succeeds and finds an optimum solution vector r (line 6). In line 7 we make sure that the vector r respects the bound conditions of lp. In line 8, we make the assumption that a witness can be reconstructed by keeping the zero entries of the proposed witness r. We simplify the constraint matrix M accordingly and nullify all the

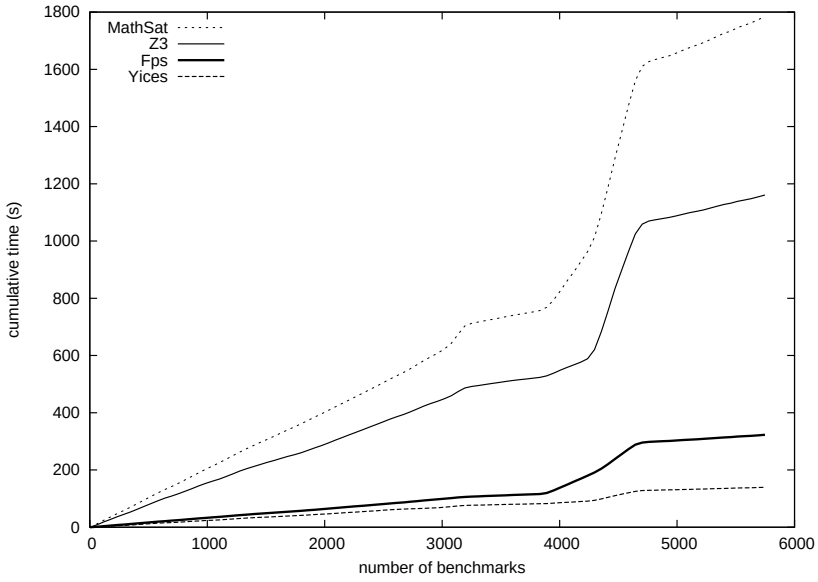

Figure 2. Unsatisfiable conjunctions extracted from SMT problems

columns of M corresponding to a zero entry of r (line 8). In practise the obtained matrix M' is very sparse (compared to M). From now on, the computations are done using exact rational arithmetic on matrix M'. In line 9, we apply Gaussian elimination to matrix M' to obtain a triangular matrix (also called echelon matrix) U. To get solutions, we fix the unknowns using the values of r and iteratively solve the remaining equations of U to get a vector w. By construction, the vector w verifies the constraint matrix M' and therefore M. If the bound conditions still hold (conditions that are checked for line 11) we return that the primal problem is unsatisfiable and that vector w is a proof witness. If the bound conditions are violated, we conclude (without formal guarantee) that the primal problem is probably unsatisfiable.

6 Experiments

We have an Ocaml prototype Fps[1], based on the Simplex solver of Glpk 4.40 [10], implementing the witness reconstruction approach described above. The experiments have been carried out on a 3.40 Ghz Intel Xeon with 4 GB of RAM running a 64bit Linux kernel. We have benchmarked our approach against Z3 2.5, MathSAT 4.3 and Yices 2.0 using the benchmark suite of already used in [11]. These are unsatisfiable conjunctions extracted from SMT verification problems provided by Leonardo de Moura (Figure 2) and random dense benchmarks (Fig-

[1] Source code available at `http://www.irisa.fr/celtique/fbesson/bbfps.tgz`

ure 3). For all the unsatisfiable benchmarks the reconstruction succeeds at generating a witness from the result of the floating-point Simplex solver.

For the first benchmark suite our prototype is slower than Yices (but less than 3 times slower) but faster than Z3 (about 3 times faster) and faster than MathSAT 4.3 (about 5 times faster). As always, benchmarks are to be taken with a pinch of salt. Here, Yices is here about 12 times faster than MathSat whereas Yices was only about 1.6 times faster during the SMT COMP 2009. Hence, it is hard to draw strong conclusions about the relative efficiency. In an attempt to understand whether there is a chance for our approach to be more competitive, we have profiled the different phases : parsing, simplex and witness reconstruction. It appears that the Simplex phase takes, on the average, less than 0.015 seconds per benchmark *i.e,* a cumulative time of about 85 seconds. This is encouraging as this is a phase that cannot be optimised without optimising Glpk – the very thing we want to avoid. Moreover, witness reconstruction is not the bottleneck (cumulative time of 22 seconds). Interestingly, Yices outputs a status (here unsat) before the end of our parsing phase. There is therefore hope that a fine-tuned version of our algorithm could be more competitive.

As already observed by Monniaux [11], SMT provers are not efficient for solving dense random benchmarks. Figure 3 shows the (sorted) cumulative time

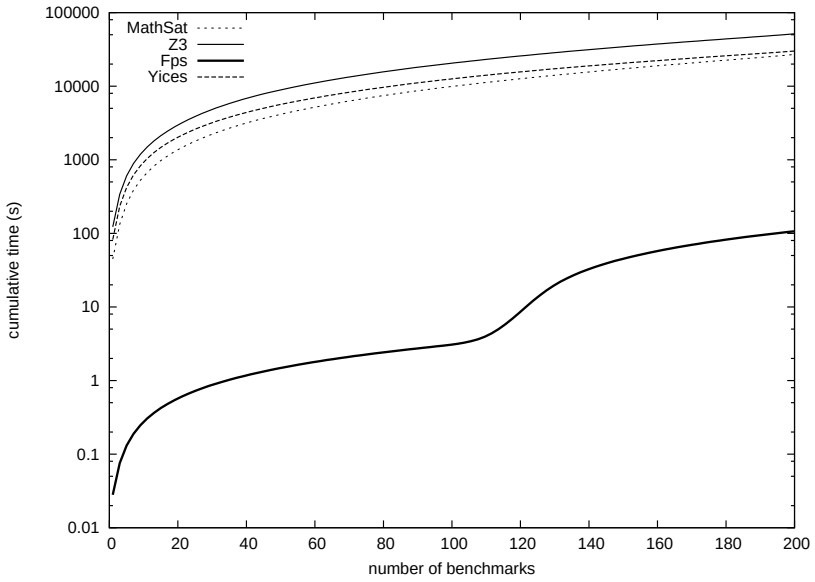

Figure 3. Random dense benchmarks

spent to solve the benchmarks. Time is plotted using a logarithmic scale and Fps outperforms Yices, Z3 and MathSat. For Fps, there is a clear inflexion

point around 100 benchmarks. This delimits the satisfiable benchmarks from the unsatisfiable ones and corresponds to the overhead of the witness reconstruction.

7 Related work

Dutertre and de Moura [6] have detailed how to design an efficient Simplex algorithm compatible with the requirements of SMT solvers. We have shown how to relax certain of those requirements by exploiting linear duality and encoding the *witness linear program*. One limitation is that a model is not generated when the primal problem is satisfiable – we only have the knowledge that the dual is unsatisfiable. This can be mitigated by solving the primal problem. Notice that this penalty is only paid once and only for satisfiable problems which are on the average faster to solve.

Faure *et al.,* couple exact and inexact LP solvers in the SMT solver Barcelogic [7]. The coupling requires the inexact LP solver to generate conflict clauses. Our approach is less demanding on the solver as the conflict clause is a by-product of turning the primal problem into a dual witness search problem. As inexact LP solvers do not handle strict inequalities, Faure *et al.,* relax them into non-strict inequalities by subtracting a small ϵ. Another advantage of our witness linear program is that it has no strict inequalities whatever the form of the primal problem. Like in this paper, when the inexact solver reports an unsatisfiable problem, the alleged conflict clause is post-processed in order to get a provably correct answer. Their post-processing which amounts to running a LP solver in exact rational arithmetic is complete – if the inexact LP solver return a genuine conflict – but much more costly than our optimistic witness reconstruction algorithm.

Dhiflaoui *et al.,* [5] exploit the basis computed by an inexact floating-point Simplex solver to compute the exact results of linear programs. The basis is *verified* using exact methods. If it is not optimal, it is *repaired* using a Simplex solver running in exact rational arithmetic. Repairing the basis using exact arithmetic can be costly and is not practical for certain LP. When the basis is wrong a more scalable approach consists in rerunning the Simplex solver with extended (floating-point) precision [9,1]. Using this approach, Koch [9] obtains the exact results of the NETLIB-LP benchmarks. Monniaux [11] adapts the approach of [5] for deciding LRA. The basis reconstruction has a complexity cubic in the size of the initial problem. In our case, because we make optimistic assumption that the conflict clause is correct our post-processing is cubic in the size of the conflict clause. For SMT problems, the conflict clauses are usually small and the gain is therefore substantial. Monniaux reports than the basis is sound in 77% of the benchmarks. When it is not, additional pivots are needed; this can be very costly. For the same benchmark suite, our witness reconstruction algorithm always succeeds. Investigation is still needed to fully understand why this is the case. What is for sure is that solving the *witness linear program* has the theoretical advantage of dealing with strict inequalities without loss of precision. More speculatively, it might possess better numeric properties.

8 Conclusion

Exploiting duality results of linear programming, we have proposed a notion of *witness linear program* which has the advantage of simplifying the interface between an SMT-solver and an off-the-shelf (exact or inexact) LP solver which can lack support for strict inequalities, conflict clauses and backtracking. For inexact LP solver, we have also proposed a witness reconstruction algorithm which despite being theoretically incomplete is very fast and effective in practise. More work is needed to fully assess the potential of using an inexact LP solver for deciding LRA. As future work, we aim at implementing the algorithm described here in an SMT solver. This would enable more thorough comparisons with state-of-the-art SMT solvers. We anticipate that our optimistic witness reconstruction will sometimes fail – hopefully on very rare occasions. To get a decision procedure, a simple method consists in coupling our witness reconstruction with Monniaux's algorithm [11]. We also intend to investigate how to exploit further properties of linear programming in order to better resist numeric inaccuracies: the existence of integer witnesses and the fact that interior points *e.g.,* the Chebyshev centre should be more immune to floating-point approximations.

References

1. D. Applegate, W. Cook, S. Dash, and D. Espinoza. Exact solutions to linear programming problems. *Oper. Res. Lett.*, 35(6):693–699, 2007.
2. S. Cotton and O. Maler. Fast and Flexible Difference Constraint Propagation for DPLL(t). In *SAT*, volume 4121 of *LNCS*, pages 170–183. Springer, 2006.
3. L. M. de Moura and N. Bjørner. Z3: An Efficient SMT Solver. In *TACAS*, volume 4963 of *LNCS*, pages 337–340. Springer, 2008.
4. L. M. de Moura, B. Dutertre, and N. Shankar. A Tutorial on Satisfiability Modulo Theories. In *CAV*, volume 4590 of *LNCS*, pages 20–36. Springer, 2007.
5. M. Dhiflaoui, S. Funke, C. Kwappik, K. Mehlhorn, M. Seel, E. Schömer, R. Schulte, and D. Weber. Certifying and repairing solutions to large LPs how good are LP-solvers? In *SODA*, pages 255–256, 2003.
6. B. Dutertre and L. M. de Moura. A Fast Linear-Arithmetic Solver for DPLL(T). In *CAV*, volume 4144 of *LNCS*, pages 81–94. Springer, 2006.
7. G. Faure, R. Nieuwenhuis, A. Oliveras, and E. Rodríguez-Carbonell. SAT Modulo the Theory of Linear Arithmetic: Exact, Inexact and Commercial Solvers. In *SAT*, volume 4996 of *LNCS*, pages 77–90. Springer, 2008.
8. ILOG. *ILOG CPLEX v.12.0*, 2009.
9. T. Koch. The final NETLIB-LP results. *Oper. Res. Lett.*, 32(2):138–142, 2004.
10. A. Makhorin. *GNU Linear Programming Kit, Version 4.40*. GNU Software Foundation, 2009.
11. D. Monniaux. On Using Floating-Point Computations to Help an Exact Linear Arithmetic Decision Procedure. In *CAV*, volume 5643 of *LNCS*, pages 570–583. Springer, 2009.
12. V. Pratt. Two easy theories whose combination is hard. Technical report, Massachusetts Institute of Technology, 1977.
13. A. Schrijver. *Theory of Linear and Integer Programming.* Wiley, 1998.

A Notations

$\mathbb{Q}^{m,n}$ is the set of rational-valued matrices with m rows and n columns. Given $A \in \mathbb{Q}^{m,n}$, A_i is the ith row of A and $A_{i,j}$ is the element at row i and column j. $A^t \in \mathbb{Q}^{m,n}$ is the matrix transpose of A. Given $A \in \mathbb{Q}^{n,k}$ and $B \in \mathbb{Q}^{k,m}$, $A \cdot B \in \mathbb{Q}^{n,m}$ is the matrix product of A by B. Given $A \in \mathbb{Q}^{m,n}$ and $B \in \mathbb{Q}^{m,n}$, $A + B$ is the component-wise addition of matrices. A vector is a matrix with 1 column and we write $\mathbb{Q}^m$ for $\mathbb{Q}^{m,1}$ the set of vectors with m elements. We write $\mathbf{0}^n$ the vector filled with zeros. When it is clear from the context, we drop the index and simply write $\mathbf{0}$. Likewise $\mathbf{1}$ is a vector filled with ones (and should not be confused with the identity matrix I). To streamline notations, we write A/B for the block matrix $\begin{pmatrix} A \\ B \end{pmatrix}$.

B Proofs

Corollary 1 (Soundness of *wit*). *Let A and B be matrices and a and b be vectors.*

$$\forall (y, z), wit(A, a, B, b, \mathbf{0}, \mathbf{0}, (y, z, \mathbf{0})) \Rightarrow \forall x, \neg(A \cdot x < a \wedge B \cdot x \leq b)$$

Proof. Ad Absurdum suppose that we have vectors y and z such that

$$wit(A, a, B, b, \mathbf{0}, \mathbf{0}, (y, z, \mathbf{0}))$$

and a vector x such that $A \cdot x < a$ and $B \cdot x \leq b$. By Lemma 1 and Definition 1, we deduce that:

(a) $a^t \cdot y + b^t \cdot z \geq 0$
(b) $\neg(y = \mathbf{0}) \Rightarrow a^t \cdot y + b^t \cdot z > 0$

To conclude the proof, we do a case analysis over item (iii) of Definition 1. If $a^t \cdot y + b^t \cdot z + c^t \cdot t < 0$, this contradicts (a). Otherwise, we have $\neg(y = \mathbf{0}) \wedge a^t \cdot y + b^t \cdot z + c^t \cdot t \leq 0$ and this contradicts (b). □

Corollary 2 (Completeness of *wit*). *Let A and B be matrices and a and b be vectors.*

$$\forall x, \neg(A \cdot x < a \wedge B \cdot x \leq b) \Rightarrow \exists (y, z), wit(A, a, B, b, \mathbf{0}, \mathbf{0}, (y, z, \mathbf{0}))$$

Proof. Suppose that $A \cdot x < a \wedge B \cdot x \leq b$ is unsatisfiable. Formally, $\neg(\exists x, A \cdot x < a \wedge B \cdot x \leq b$. Given P defined as

$$P(y, z) = y \geq \mathbf{0} \wedge z \geq \mathbf{0} \Rightarrow \begin{cases} A^t \cdot y + B^t \cdot z = \mathbf{0} & \Rightarrow a^t \cdot y + b^t \cdot z \geq 0 \\ A^t \cdot y + B^t \cdot z = \mathbf{0} \wedge \neg(y = \mathbf{0}) & \Rightarrow a^t \cdot y + b^t \cdot z > 0 \end{cases}$$

by Lemma 1, we have $\neg(\forall (y, z), P(y, z))$ *i.e.,* $\exists (y, z), \neg P(y, z)$. By propositional reasoning, $\neg(P(y, z))$ is equivalent to

$$y \geq \mathbf{0} \wedge z \geq \mathbf{0} \wedge A^t \cdot y + B^t \cdot z = \mathbf{0} \wedge ((\neg a^t \cdot y + b^t \cdot z \geq 0) \vee (\neg(y = \mathbf{0}) \wedge \neg(a^t \cdot y + b^t \cdot z > 0)))$$

which (by arithmetic reasoning) is equivalent to Definition 1. As a result, a pair (y, z) such that $\neg P(y, z)$ is a witness and verifies $wit(A, a, B, b, \mathbf{0}, \mathbf{0}, (x, y, \mathbf{0}))$. □

Remark 1. Let C be a matrix and c be a vector, the following holds:

$$C \cdot x = c \text{ iff } (C/-C) \cdot x \leq (c/-c)$$

Theorem 1. *Let A, B and C be matrices and a, b and c be vectors.*

$$\exists (y, z, t), wit(A, a, B, b, C, c, (y, z, t)) \text{ iff } \forall x, \neg(A \cdot x < a \wedge B \cdot x \leq b \wedge C \cdot x = c)$$

Proof. We first prove that for all vectors y and z the following holds:

$$\begin{gathered} \exists t, wit(A, a, B, b, C, c, (y, z, t)) \\ \text{if and only if} \\ \exists (u, v), wit(A, a, (B/C/-C), (b/c/-c), \mathbf{0}, \mathbf{0}, (y, (z/u/v), \mathbf{0})) \end{gathered}$$

The proof relies on the following distributivity laws

$$X \cdot (w - w') = X \cdot w + (-X) \cdot w' \tag{1}$$
$$(X/Y)^t \cdot (w/w') = X^t \cdot w + Y^t \cdot w' \tag{2}$$

⇒ Suppose that (y, z, t) is a witness for some t. The vector t can be uniquely decomposed into u and v such that $u \geq \mathbf{0}$, $v \geq \mathbf{0}$ and $t = u - v$. Equation (1) and equation (2) allow to prove

$$wit(A, a, (B/C/-C), (b/c/-c), \mathbf{0}, \mathbf{0}, (y, (z/u/v), \mathbf{0}))$$

This concludes the ⇒ part.

⇐ Suppose that $(y, (z/u/v), \mathbf{0})$ is a witness for some u and v. Again, the two previous equations allow to conclude that $wit(A, a, B, b, C, c, (y, z, u - v))$.

Theorem 1 follows from Corollary 1, Corollary 2 and Remark 1. □

Lemma 1 (Soundness of *wlp*). *If $wlp(A, a, B, b, C, c)$ has optimum $y/z/t/u/v$ then $wit(A, a, B, b, C, c, (y, z, t))$ holds.*

Proof. Because this is an optimal, the vector $y/z/t/u/v$ is a feasible solution of the constraint matrix and the bound conditions of *wlp*. As a result, the conditions (i) and (ii) of *wit* are trivially fulfilled. From equation ⓑ we have

$$u = -(a^t \cdot y + b^t \cdot z + c^t \cdot t) \tag{3}$$

Consider the exclusive cases $y = \mathbf{0}$ and $\neg y = \mathbf{0}$.

$y = \mathbf{0}$ From equations ⓒ and (3) and bound condition ⓗ we obtain

$$a^t \cdot y + b^t \cdot z + c^t \cdot t - \mathbf{1}^t \cdot y \leq -1$$

However, $\mathbf{1}^t \cdot y = 0$ and therefore $a^t \cdot y + b^t \cdot z + c^t \cdot t \leq -1 < 0$. This fulfills condition (iii) of *wit* and conclude the proof.

$\neg y = \mathbf{0}$ From equation (3) and bound condition ⓖ we obtain

$$a^t \cdot y + b^t \cdot z + c^t \cdot t \leq 0$$

As we have $\neg y = \mathbf{0}$ this fulfills condition (iii) of *wit* and conclude the proof. □

Lemma 2 (Completeness of *wlp*). *If* $wit(A, a, B, b, C, c, (y, z, t))$ *then there exists* $\alpha > 0$*,* u *and* v *such that* $wlp(A, a, B, b, C, c)$ *has optimum* $\alpha \cdot (y/z/t/u/v)$*.*

Proof. Suppose that $wit(A, a, B, b, C, c, (y, z, t))$ and define τ, u and v by:

$$\tau = a^t \cdot y + b^t \cdot z + c^t \cdot t \tag{4}$$
$$\alpha = -(\tau - \mathbf{1}^t \cdot y)^{-1} \tag{5}$$
$$u = -\tau \tag{6}$$
$$v = -(\tau - \mathbf{1}^t \cdot y) \tag{7}$$

We first prove that α is strictly positive *i.e.,* $\tau - \mathbf{1}^t \cdot y < 0$. From condition (iii) we obtain that $a^t \cdot y + b^t \cdot z + c^t \cdot t \leq 0$. Moreover, because $y \geq \mathbf{0}$ we have $1^t \cdot y \geq 0$. As a result, we have $\tau - \mathbf{1}^t \cdot y \leq 0$. *Ad Aburdum* suppose $\tau - \mathbf{1}^t \cdot y$ is null. Consider the exclusive cases $y = \mathbf{0}$ and $\neg y = \mathbf{0}$.

$y = \mathbf{0}$ We have $1^t \cdot y = 0$. The proof is by case analysis over condition (iii) of *wit*. Either $a^t \cdot y + b^t \cdot z + c^t \cdot t < 0$ and this is contradictory or $\neg y = 0$ and this is also contradictory.

$\neg y = \mathbf{0}$ In this case, $\mathbf{1}^t \cdot y$ is strictly positive and this is contradictory.

As the objective function of *wlp* is the constant null function, it remains to prove that the solution $\alpha \cdot (y/z/t/u/v)$ is feasible. First remark that scaling a witness by a strictly positive constant preserves the witness property. As a result, we have $wit(A, a, B, b, C, c, (\alpha \cdot y, \alpha \cdot z, \alpha \cdot t))$. This establishes the equation ⓐ and bound conditions ⓓ , ⓔ , ⓕ . Equations ⓑ and ⓒ are direct consequences of equations (6) and (7). The bound condition ⓖ $(0 \leq \alpha \cdot u \leq +\boldsymbol{\infty})$ holds because u and α are strictly positive. It remains to prove bound condition ⓗ . We have

$$v = -(\tau - \mathbf{1}^t \cdot y) \tag{8}$$
$$\alpha \cdot v = -(\tau - \mathbf{1}^t \cdot y)^{-1} \times -(\tau - \mathbf{1}^t \cdot y) \tag{9}$$
$$\alpha \cdot v = 1 \tag{10}$$

As a result the bound condition ⓗ is satisfied. This concludes the proof. □