An Extension of the Davis-Putnam Procedure and its Application to Preprocessing in SMT

Roberto Bruttomesso

Università della Svizzera Italiana, Lugano, Switzerland

Abstract. We present a decision procedure for SMT($\mathcal{LRA}$) that works by eliminating Boolean and rational variables. The algorithm we propose (DPFM) is based on a combination of the Davis-Putnam procedure and the Fourier-Motzkin elimination. We report on preliminary experiments where DPFM is not directly used to solve the formula (as its prohibitive complexity does not make it practical), but it is instead used in a controlled manner as a simplification and preprocessing device.

1 Introduction

Decision procedures for Satisfiability Modulo the Theory of Linear Rational Arithmetic (SMT($\mathcal{LRA}$)) have received a great attention in the area of formal verification in the last decade, in particular for encoding and solving verification conditions for software. Most state-of-the-art SMT-Solvers, such as Barcelogic [3], CVC3 [2], MathSAT [5], Yices [8], and Z3 [7], currently implement decision procedures for SMT($\mathcal{LRA}$).

In this paper we propose yet another procedure for SMT($\mathcal{LRA}$) that is based on a combination of the Davis-Putnam [6] (DP) and the Fourier-Motzkin [11] elimination procedures (DPFM). The main idea is to extend DP with a resolution rule, based on Fourier-Motzkin elimination, that natively operates on SMT($\mathcal{LRA}$) clauses. Despite the high complexity of the procedure, we show how DPFM can be used, in a controlled manner, to simplify the input formula or to statically derive and learn knowledge about it. Our preliminary experiments show that using DPFM as preprocessor may speed-up the overall solving effort.

The closest work to ours is the one proposed by Strichman in [18,19]: it is an eager [1] approach for SMT($\mathcal{LRA}$) that makes use of the Fourier-Motzkin elimination procedure. The approach can be summarized as follows. Each distinct $\mathcal{LRA}$ constraint c_i in the input problem φ is replaced with a fresh Boolean variable e_i to obtain the Boolean abstraction φ' of φ. Subsequently φ' is enriched with a sufficient number of implications of the form $(e_i \wedge e_j \rightarrow e_k)$ where e_k is a fresh Boolean variable associated to a new $\mathcal{LRA}$ constraint. e_k results from the application of a Fourier-Motzkin step on the $\mathcal{LRA}$ constraints associated to e_i and e_j. The final formula contains only propositional variables, and it can be solved with any SAT-Solver.

[1] See [1] for details about the eager and the lazy approach to SMT.

Our approach is also based on Fourier-Motzkin, which is however used to *eliminate* rational variables, instead of generating theory lemmata. In particular we use Fourier-Motzkin to derive a resolution rule for SMT($\mathcal{LRA}$) upon which we define our DPFM procedure. DPFM allows to choose the amount of variables to eliminate and which variables to eliminate. It is also possible to alternate Boolean and rational variable elimination.

The other work that inspired our approach is the SATElite algorithm [10], that is in turn an improved approach over the NiVER tool [20]. SATElite is a preprocessor for SAT formulæ based on the variable elimination rule of DP [6], which is applied under certain controlled conditions in order to produce a simplified version of the input formula at a little computational cost. In this work we follow the same philosophy of SATElite, in that we aim at simplifing the formula using DPFM instead of DP.

After the submission of this paper, we learned about another work [12] almost simultaneously submitted and published at CAV09[2]. The authors of [12] introduce a generalized version of DPLL, called GDPLL. GDPLL can be used, for instance, to explore rational values for variables in an SMT($\mathcal{LRA}$) formula in a similar way as DPLL explores values for Booelans. The so-called *shadow* rule that they introduce is extremely similar to our SMT($\mathcal{LRA}$) Resolution. An application of the shadow rule is, in fact, equivalent to many applications of our SMT($\mathcal{LRA}$) Resolution, after the trasformation in OCCF.

We also show that DPFM can be used as a device for performing static learning. Static learning [4,22] is the process of eagerly learning some lemmata before the actual search starts. We propose a static learning technique based on DPFM that, in contrast to [4], is capable of learning clauses that mix Boolean and theory atoms. Differently from [22] our method is controlled by two parameters, centrality and trade-off, that can be used to control the amount and the quality of information to be added. More importantly, the clauses learnt with DPFM are not consequences of the theory under consideration, but only of the structure of input formula. This is important in order to avoid learning combinations of theory atoms that are already prevented by the Boolean structure of the formula.

Finally we report on a set of preliminary experiments on the $\mathcal{DL}$ fragment of linear arithmetic that indicates that our preprocessing techniques effectively simplify the input formula.

2 Background

2.1 Preliminaries

In the following we use the letters $\{x, y\}$ to denote $\mathcal{LRA}$ variables, k to denote a rational constant, and $\{p, q\}$ to denote $\mathcal{LRA}$ polynomes. Also, we use $\{a\}$ to denote Boolean variables, and $\{C, D\}$ to denote (either Boolean or SMT($\mathcal{LRA}$)) clauses. We shall assume that clauses do not contain duplicate literals, or both the positive and the negative version of a literal.

[2] We thank an anonymous reviewer for pointing us to that work.

For the sake of simplicity we assume that the only predicate symbol used to express $\mathcal{LRA}$-atoms is $\leq$. Notice that this is not a restriction as $p < k$ can be rewritten into $p \leq k + \delta$, where $\delta > 0$ can be dealt with as an infinitesimal parameter, as described in [9]. Similarly, we will assume that $\mathcal{LRA}$-atoms always appear non negated in clauses, since $\neg(p \leq k)$ is equivalent to $k < p$, and therefore to $-p \leq -k + \delta$.

In the rest of the paper a *bound* is an expression of the form $x_i \leq p$ (upper bound), or $-x_i \leq p$ (lower bound), if x_i does not occur in p. If a variable x_i occurs (with a non-zero coefficient) in an $\mathcal{LRA}$-atom $p < k$, then we can always rewrite it into an upper or a lower bound with straightforward algebraic manipulations. We shall denote a set of upper and lower bounds for a variable x with B_x and B_{-x} respectively.

In the following we will recall two well-known techniques for variable elimination.

2.2 Resolution for Booleans and the Davis-Putnam Procedure

Boolean Resolution has been first introduced by Robinson [15,16]. It allows the derivation of an implied clause, called *resolvent*, from a pair of clauses that contain a complementary literal, called *pivot*.

Definition 1 (Boolean Resolution). *Let $C = C_1 \vee a \vee C_2$ and $D = D_1 \vee \neg a \vee D_2$ be two clauses. The* resolvent *$C \otimes_a D$ of C and D on the pivot variable a is $C_1 \vee C_2 \vee D_1 \vee D_2$.*

This notion can be also lifted to sets of clauses as follows.

Definition 2 (Boolean Resolution for two sets of clauses). *Let S_a, $S_{\neg a}$ two sets of clauses containg an occurrence of a or $\neg a$ respectively. Then*

$$S_a \otimes_a S_{\neg a} = \{C_1 \otimes_a C_2 \mid C_1 \in S_a, C_2 \in S_{\neg a}\}$$

An important property of Boolean Resolution is that it preserves equisatisfiability, i.e., the set of clauses $S_a \cup S_{\neg a}$ is equisatisfiable with $S_a \otimes_a S_{\neg a}$ defined above.

Boolean Resolution for sets of clauses is used in a well-known decision procedure for Boolean formulæ, described by M. Davis and H. Putnam in [6][3]. The procedure, named DP, takes as input a Boolean formula φ in Conjunctive Normal Form (CNF), and iteratively applies the following three rules.

Rule I-$\mathbb{B}$: Unit Propagation. If a variable a_i appears in a unary clause l, set a_i to v, v being $\top$ if $a_i = l$ or $\bot$ if $\neg a = l$. Then replace all positive occurences of a_i with v, and all occurrences of $\neg a_i$ with $\neg v$.

Rule II-$\mathbb{B}$: Affirmative-Negative Rule. If a variable a_i appears in φ only positively resp. negatively, then set a_i to $\top$ resp. $\bot$ and remove clauses containing a_i.

[3] Notice that DP was invented *before* the pubblication of Robinson's Resolution rule.

Rule III-$\mathbb{B}$: Variable Elimination. This rule is an application of the Boolean Resolution rule for sets of clauses.

Each rule aims at eliminating one Boolean variable from the set of clauses. Rules I and II are just optimizations, and they are not needed for the method to be complete. The algorithm terminates either when an empty clause is generated (unsatisfiable formula) or when every clause is removed (satisfiable formula).

2.3 $\mathcal{LRA}$ Resolution and the Fourier-Motzkin Elimination Method

As observed in [21], a similar notion of resolution can be defined for $\mathcal{LRA}$ constraints. A $\mathcal{LRA}$ constraint of the form $\sum_i k_i x_i \leq k$ plays a role similar to a Boolean clause, where addition "+" is the correpondent of disjunction "$\vee$" and negative k_i coefficients corresponds to negation "$\neg$". The obvious difference is that, in linear arithmetic, coefficients may assume values in $\mathbb{R}$. However if we focus on a particular variable x_i, it is always possible to rewrite a constraint in the form of a bound $\pm x_i \leq p$, where p is the polynome $\frac{1}{k_i}(k - \sum_{j \neq i} k_j x_j)$.

In order to stress the similarities with Boolean Resolution, we shall refer to the variable elimination process for $\mathcal{LRA}$ as to $\mathcal{LRA}$ Resolution.

Definition 3 ($\mathcal{LRA}$ Resolution). *Let $B_x = x \leq p$ an upper bound for x and $B_{-x} = -x \leq q$ a lower bound for x. Then $B_x \otimes_x B_{-x} = -q \leq p$.*

The resulting polynome can be then simplified into the canonical form $\sum_i k_i x_i \leq k$. As for Boolean Resolution, $\mathcal{LRA}$ Resolution can be lifted to sets of constraints.

Definition 4 ($\mathcal{LRA}$ Resolution for two sets of constraints). *Let S_x be a set of upper bounds for x and S_{-x} a set of lower bounds for x. Then*

$$S_x \otimes_x S_{-x} = \{B_1 \otimes_x B_2 \mid B_1 \in S_x, B_2 \in S_{-x}\}$$

To avoid unnecessary complication in the notation, we shall assume that whenever we generate a new $\mathcal{LRA}$ constraint it is automatically put in its canonical form. Similarly, we will not explicitly mention the rewriting process that is necessary to obtain bounds from $\mathcal{LRA}$ constraints. Rewriting $\mathcal{LRA}$ constraints is both conceptually and computationally easy.

$\mathcal{LRA}$ Resolution for two sets of constraints is the basic rule behind the well-knonw Fourier-Motzkin Elimination method (see [17] for a thorough description).

3 An Extension of the Davis-Putnam Procedure to SMT($\mathcal{LRA}$)

This section presents a method for eliminating an arbitrary variable, either Boolean and Real, from an SMT($\mathcal{LRA}$) formula. In particular, we will introduce a new form of resolution that can be seen as a combination of Boolean and $\mathcal{LRA}$ Resolutions discussed above. We shall call the new rule SMT($\mathcal{LRA}$) Resolution.

3.1 A preliminary preprocessing step

The decision procedure we are about to present is defined to work on a particular format for SMT($\mathcal{LRA}$) formulæ, that we call *One Constraint per Clause Form (OCCF)*. Formally, an SMT($\mathcal{LRA}$) formula φ is in OCCF if (i) φ is in CNF, and (ii) each clause in φ contains at most one $\mathcal{LRA}$-atom.

Any SMT($\mathcal{LRA}$) CNF formula φ can be easily converted into an equisatisfiable OCCF in linear time in the number of literals, using an additional set of fresh Boolean variables: the idea is to consider one clause at a time, and to split those clauses that contain more than one $\mathcal{LRA}$-atom into smaller ones. The correspondence with original clauses is guaranteed by introducing new Boolean variables acting as "links".

For instance consider a clause $C = (a_1 \vee x_1 + x_2 \leq 5 \vee x_3 - x_1 \leq 10 \vee a_2)$. In order to separate the two theory atoms $x_1 + x_2 \leq 5$ and $x_3 - x_1 \leq 10$, we introduce a fresh Boolean variable a_3, and split C into the clauses $C' = (a_1 \vee x_1 + x_2 \leq 5 \vee a_3)$ and $C'' = (\neg a_3 \vee x_3 - x_1 \leq 10 \vee a_2)$. C' and C'' are both in OCCF, and they are equisatisfiable with C. In general one needs $n + 1$ clauses to replace a clause containing n $\mathcal{LRA}$-atoms.

3.2 Resolution for SMT($\mathcal{LRA}$)

In the following we introduce a notion of resolution that operates on SMT($\mathcal{LRA}$) clauses in OCCF. The SMT($\mathcal{LRA}$) Resolution eliminates one rational variable x from two clauses in which x appears in an upper and in a lower bound respectively.

Definition 5 (SMT($\mathcal{LRA}$) Resolution). *Let $C = C_1 \vee (x \leq p) \vee C_2$ and $D = D_1 \vee (-x \leq q) \vee D_2$ be two clauses in OCCF. The* resolvent $C \otimes_x D$ *of C and D on the rational variable x is $C_1 \vee C_2 \vee (-q \leq p) \vee D_1 \vee D_2$.*

Example 1. Let $C = (a_1 \vee x_1 \leq 5 - x_2 \vee a_2)$ and $D = (a_3 \vee -x_1 \leq 10 - x_3)$. Then $C \otimes_{x_1} D = (a_1 \vee -(10 - x_3) \leq 5 - x_2 \vee \neg a_2 \vee a_3) = (a_1 \vee x_3 + x_2 \leq 15 \vee a_2)$.

We have the following lemma[3].

Lemma 1 (Soundness of SMT($\mathcal{LRA}$) Resolution). *Let $C = C_1 \vee (x \leq p) \vee C_2$, $D = D_1 \vee (-x \leq q) \vee D_2$ two clauses in OCCF. Then $C \wedge D$ is equisatisfiable with $C \otimes_x D$.*

As for the resolution methods recalled in § 2, we can extend SMT($\mathcal{LRA}$) Resolution to a set of OCCF clauses as follows.

Definition 6 (SMT($\mathcal{LRA}$) Resolution for two sets of clauses). *Let B_x and B_{-x} two sets of clauses in OCCF where x appears upper bounded and lower bounded respectively. We define $B_x \otimes_x B_{-x}$ as the set of clauses $\{C_1 \otimes_x C_2 \mid C_1 \in B_x, C_2 \in B_{-x}\}$.*

[3] The proof is given in the Appendix

SMT($\mathcal{LRA}$) Resolution for two sets of clauses preserves satisfiability[4].

Lemma 2 (Soundness of SMT($\mathcal{LRA}$) Resolution for sets of clauses). *Let B_x and B_{-x} two sets of clauses in OCCF form, x appears upper bounded and lower bounded respectively. Then $B_x \cup B_{-x}$ is equisatisfiable with $B_x \otimes_x B_{-x}$.*

Using the variable elimination for SMT($\mathcal{LRA}$) defined above it is possible to define a decision procedure for SMT($\mathcal{LRA}$) that works by eliminating variables, both Boolean and Real.

```
DPFM
Input: $\varphi$ in OCCF
Apply the following rules in a fair manner, until
every clause is removed or and empty clause is found:
    Rule I-$\mathbb{B}$: Unit Propagation for Booleans
    Rule II-$\mathbb{B}$: Affirmative-Negative Rule for Booleans
    Rule III-$\mathbb{B}$: Variable Elimination for Booleans[5]
    Rule I-$\mathbb{R}$: Unit Propagation for Reals
    Rule II-$\mathbb{R}$: Affirmative-Negative Rule for Reals
    Rule III-$\mathbb{R}$: Variable Elimination for Reals
```

[5] Taking care of transforming newly added clauses in OCCF.

The first three rules are inherited from DP, while the remaining three rules are described as follows.

Rule I-$\mathbb{R}$: Unit Propagation for Reals. If two bounds $x \leq p$ and $-x \leq -p$ are present in the formula as two unit clauses (i.e., $x = p$), then substitute x with p in all the remaining clauses.

Rule II-$\mathbb{R}$: Affirmative-Negative Rule. If a variable x appears only in lower or upper bounds than all contraints containing x can be replaced with $\top$, and therefore all the clauses containing such constraints can be removed. In fact suppose that in a formula φ x appears only in upper bounds. Let $P = \{x \leq p_1, \ldots, x \leq p_n\}$ be the set of the upper bounds, and let φ' the formula resulting from the removal of clauses containing x. If φ is satisfied by a model μ, than φ' can be satisfied by μ, as it contains less clauses. On the other hand, let μ' be a model for φ'; we can extend μ' by choosing a value for x that is lower than $\min(\mu'(p_1), \ldots, \mu'(p_n))$, which satisfies all the clauses in $\varphi \setminus \varphi'$.

Rule III-$\mathbb{R}$: Variable Elimination. This rule is an application of the SMT($\mathcal{LRA}$) Resolution for set of clauses on a chosen real variable x.

4 Practical Applications

The DPMF procedure presented in the previous section yields a worst-case computational complexity which cannot be inferior to the ones of DP and FM. In both situations the amount memory required to store new constraints is $O(m^{2^n})$, where m is the number of initial constraints and n is the number of variables. The high complexity makes the procedure as it is of impractical use for large values of m and n.

However, as we are going to show in the following, we can still benefit from a controlled execution of DPFM in order to simplify or generate new knowledge about the formula, in order to speed up the solving time for a state-of-the-art SMT-Solver. In particular we would like to follow the same philosophy underlying the SATElite approach [10], where Boolean resolution steps are applied only under certain controlled conditions.

In our approach in particular we focus on two *structural* aspects of the formula that concern theory variables: *centrality* and *trade-off*. With the term "centrality" for a variable x we indicate a measure of the degree of connection of x with other theory variables. Precisely the centrality for x is the cardinality of the set $\{y \mid y \text{ and } x \text{ appear both in some } \mathcal{LRA} \text{ constraint }\}$. The other parameter that we consider is a "trade-off" value between the elimination of a theory variable and the number of new clauses to be added. In other words the trade-off is the maximum price, in terms of new clauses, that the procedure is allowed to pay for eliminating one theory variable.

We implemented our DPFM procedure, parametrized with centrality and trade-off values, within OpenSMT [13]. After DPFM the SATElite algorithm is also run[5] in order to further simplify the formula and in particular to remove unnecessary auxiliary Boolean variables introduced during the OCCF transformation.

In the following we report on two preliminary experiments conducted on some SMT($\mathcal{DL}$) formulæ taken from the SMT-LIB [6]. We decided to restrict to SMT($\mathcal{DL}$) as they allow a simpler implementation of DPFM. $\mathcal{DL}$ constraints have the form $x - y \leq k$, and they are therefore easier to manipulate. It is sufficient to track positive and negative occurences of theory variables in constraints. Notice also that an application of SMT($\mathcal{LRA}$) resolution to a pair of $\mathcal{DL}$-atoms is always guaranteed to produce another $\mathcal{DL}$-atom. Since difference logic is a fragment of linear arithmetic, our results can be still applied. As future work we plan to extensively test our procedure on a wider set of benchmarks.

4.1 Formula Simplification

Empirically we observed that some SMT($\mathcal{DL}$) problems contain a relevant number of $\mathcal{DL}$-variables with very low centrality value. For instance this is the case

[5] Only on Boolean variables.
[6] Logs available at `www.inf.unisi.ch/postdoc/bruttomesso/SMT2009`.

for the SMT-LIB `qlock` benchmarks, which encode a queuing locking mechanism, where around 40% of $\mathcal{DL}$-variables have centrality 1 or 2. A selective application of a DPFM-based preprocessing on those variables always results in a simplified formula.

Table 1 shows a comparison of the structural properties of the `qlock` benchmarks, without (WO) and with (W) the application of our DPFM-based preprocessor, with centrality=2 and trade-off=128. In all benchmarks we observe a reduction of clauses, Boolean atoms, $\mathcal{DL}$-atoms, and $\mathcal{DL}$-variables, at the cost of few milliseconds in the preprocessing phase (preprocessing time includes parsing, DPFM, and SATELITE). At the same time, as shown in Table 2, the preprocessed formulæ are also simpler to solve.

	P.Time (s)		Clauses		BAtoms		TAtoms		TVars	
Bench	WO	W	WO	W	WO	W	WO	W	WO	W
Ind 37	1.08	6.57	41137	35299	6580	5371	6129	5285	829	185
Ind 38	1.16	6.62	42265	36244	6757	5515	6299	5423	851	188
Ind 39	1.19	7.02	43381	37150	6934	5659	6467	5562	873	189
Ind 40	1.17	7.05	44457	38114	7111	5803	6619	5702	895	203
Base 18	0.80	1.87	18630	16314	2946	2405	2867	2559	375	137
Base 19	0.82	2.31	19780	17269	3123	2548	3045	2702	397	150
Base 20	0.95	2.47	20914	18246	3300	2693	3215	2851	419	151
Base 21	0.94	2.54	22052	19193	3477	2836	3389	2995	441	155

Table 1. Structural properties for `qlock` benchmarks without (WO) and with (W) DPFM-based preprocessing with centrality=2 and trade-off=128. Columns show the comparison of preprocessing time (P.Time), number of clauses (Clauses), number of Boolean atoms (BAtoms), number of $\mathcal{DL}$-atoms (TAtoms), number of $\mathcal{DL}$-variables. In both cases SATELITE preprocessor is enabled.

Bench	Time WO (s)	Time W (s)	Bench	Time WO (s)	Time W (s)
Base 18	61.3	**59.0**	Ind 37	90.5	**18.0**
Base 19	146.1	**138.4**	Ind 38	105.7	**54.6**
Base 20	> 1800	**940.1**	Ind 39	64.4	**46.7**
Base 21	1367.9	**765.0**	Ind 40	98.3	**37.3**

Table 2. Total run-time for the most challenging `qlock` instances (for OPENSMT) with (W) and without (WO) preprocessing. For benchmarks Base 22 to 40 both versions timeout. Tests were executed on an Intel Xeon 3.0 GHz, with a timeout of 1800 s.

Centr.	Trade-Off	T. Vars Elim.	P.Time	Clauses	TAtoms	BAtoms	T.Time (s)
-	-	0	0.05	216	612	0	> 1800
12	64	0	0.05	216	612	0	> 1800
12	256	2	0.06	458	832	22	180.0
12	1024	4	0.04	1094	968	42	91.4
12	4096	6	0.09	3076	1032	60	67.2
12	16384	6	0.10	3076	1032	60	67.1
18	64	0	0.02	216	612	0	> 1800
18	256	4	0.02	714	1054	56	192.3
18	1024	8	0.07	2005	1566	109	105.6
18	4096	12	0.15	5702	2254	156	125.6
18	16384	12	0.16	5702	2254	156	125.9
24	64	0	0.02	216	612	0	> 1800
24	256	4	0.03	781	1108	66	193.2
24	1024	8	0.07	1978	1638	117	157.1
24	4096	11	0.19	5005	2198	153	89.4
24	16384	12	0.32	5519	2294	163	92.2

Table 3. Structural properties and total run-time for a job-shop benchmark with 12 jobs and 2 machines (precisely, the SMT-LIB benchmark `QF_IDL/job_shop/jobshop12-2-6-6-2-4-9.smt`) on an Intel Xeon 3.0 GHz. The first row corresponds to running OpenSMT without any DPFM-based preprocessing step (with SATElite enabled). The other rows show the behaviour of the preprocessor with different values for centrality and trade-off parameters.

4.2 Mixed Boolean-Theory Static Learning

By running a set of controlled experiments we observed that the application of our DPFM-based preprocessor to variables with centrality greater than 5, generally does not reduce the size of the formula. Still the newly added clauses may be learnt before starting the Boolean search.

Traditional static learning techniques [4,22] tend to generate clauses representing theory atoms incompatibilities, in order to prevent too many calls to the theory solver at runtime. One possible drawback of these approaches, is that one may add too many unnecessary clauses to the problem: the Boolean structure of the formula itself may already prevent some theory atoms combinations. With our procedure, instead, we statically learn clauses which mix Boolean atoms and theory constraints that are *implications* of the original formula and not only consequences of the specific theory under consideration.

In order to test the effectiveness of our mixed static learning with different configurations of centrality and trade-off, we ran some experiments on some `job_shop` instances of the SMT-LIB. In particular we report, in Table 3, a detailed analysis using multiple runs of an (unsatisfiable) job shop scheduling benchmark with 12 jobs and 2 machines, where we play with different values for centrality and trade-off (a similar trend can be also observed for other job shop instances).

Using static learning generally results into smaller run-times. The combinations for centrality and trade-off that result in the best performance are however different for different benchmarks: as future work we want to investigate good heuristics for choosing the values for the parameters, based on the formula's structure.

5 Conclusion and Future Work

We have presented a decision procedure for SMT($\mathcal{LRA}$) that is a combination of the Davis-Putnam and Fourier-Motzkin proceudes. The procedure uses the notion of SMT($\mathcal{LRA}$) resolution that allows the elimination of rational variables from an SMT($\mathcal{LRA}$) formula. It can be used as a preprocessor and simplification device.

As future work we plan to extend the current implementation of the preprocessor in order to be able to experiment with a broader set of benchmarks, including SMT($\mathcal{LIA}$) formulæ (using the integer version of Fourier-Motzkin described in [14]), and to study heuristics for an optimal control of our simplification parameters. Also we would like to experiment with different interleaving strategies with the SATElite preprocessor.

Acknowledgments

We thank Edgar Pek for the interesting discussions and his support on the $\mathcal{DL}$ solver.

References

1. C. Barrett, R. Sebastiani, S. Seshia, and C. Tinelli. *Handbook on Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, chapter Satisfiability Modulo Theories. IO Press, 2009.
2. C. Barrett and C. Tinelli. CVC3. In *CAV'07*, 2007.
3. M. Bofill, R. Nieuwenhuis, A. Oliveras, E. Rodríguez Carbonell, and A. Rubio. The Barcelogic SMT Solver. In A. Gupta and S. Malik, editors, *CAV'08*, volume 5123 of *Lecture Notes in Computer Science*, pages 294–298. Springer, 2008.
4. M. Bozzano, R. Bruttomesso, A. Cimatti, T. Junttila, P. van Rossum, S. Schulz, and R. Sebastiani. MathSAT: Tight Integration of SAT and Mathematical Decision Procedures. *JAR*, 35(1-3):265–293, 2005.
5. R. Bruttomesso, A. Cimatti, A. Franzén, A. Griggio, and R. Sebastiani. The MathSAT 4 SMT Solver. In *CAV*, pages 299–303, 2008.
6. Martin Davis and Hilary Putnam. A Computing Procedure for Quantification Theory. *J. ACM*, 7(3):201–215, 1960.
7. L. de Moura and N. Bjørner. Z3: An Efficient SMT Solver. In *TACAS'08*, pages 337–340, 2008.
8. B. Dutertre and L. de Moura. The Yices SMT Solver. Tool paper available at `http://yices.csl.sri.com/tool-paper.pdf`.

9. B. Dutertre and L. M. de Moura. A Fast Linear-Arithmetic Solver for DPLL(T). In *CAV'06*, pages 81–94, 2006.
10. N. Eén and A. Biere. Effective Preprocessing in SAT Through Variable and Clause Elimination. In *SAT*, pages 61–75, 2005.
11. J.B.J. Fourier. Solution d'une question particulire du calcul des angalits. *Oevres*, II:314–328, 1826.
12. K. L. McMillan, A. Kuehlmann, and M. Sagiv. Generalizing dpll to richer logics. In *CAV*, pages 462–476, 2009.
13. OpenSMT. `http://verify.inf.unisi.ch/opensmt`.
14. William Pugh. The Omega Test: a Fast and Practical Integer Programming Algorithm for Dependence Analysis. In *Supercomputing*, pages 4–13, 1991.
15. J. A. Robinson. A Machine-Oriented Logic Based on the Resolution Principle. *J. ACM*, 12(1):23–41, 1965.
16. J. A. Robinson. A Generalized Resolution Principle. *Machine Intelligence*, 3:77–93, 1968.
17. A. Schrijver. *Theory of linear and integer programming.* Paperback, 1998.
18. Ofer Strichman. On Solving Presburger and Linear Arithmetic with SAT. In *FMCAD*, pages 160–170, 2002.
19. Ofer Strichman. Deciding Disjunctive Linear Arithmetic with SAT. *CoRR*, 2004.
20. S. Subbarayan and D. K. Pradhan. NiVER: Non increasing variable elimination resolution for preprocessing SAT instances. In *SAT'04*, pages 276–291, 2004.
21. H. P. Williams. Logic applied to integer programming and integer programming applied to logic. *European Journal of Operational Research*, 81(3):605 – 616, 1995.
22. Y. Yu and S. Malik. Lemma Learning in SMT on Linear Constraints. In *SAT*, pages 142–155, 2006.

A Proofs

Lemma 1 (Soundness of SMT($\mathcal{LRA}$) Resolution).
Let $C = C_1 \vee (x \leq p) \vee C_2$, $D = D_1 \vee (-x \leq q) \vee D_2$ two clauses in OCCF. Then $C \wedge D$ is equisatisfiable with $C \otimes_x D$.

Proof. Suppose that $C \wedge D$ is satisfied by a model μ. If either $(x \leq p)$ or $(-x \leq q)$ are evaluated to $false$ by μ, then at least one of C_1, C_2, D_1, D_2 must evaluate to $true$, and therefore $C \otimes_x D$ is satisfied.

Suppose that $C \otimes_x D$ is satisfied by a model μ. If $(-q \leq p)$ evaluates to $true$, then we can choose a value for x such that $(\mu(x) \leq \mu(p))$ and $(-\mu(x) \leq \mu(q))$. If $(-q \leq p)$ evaluates to $false$, then at least one of C_1, C_2, D_1, D_2 must evaluate to $true$. If C_1 evaluates to $true$, than C is satisfied. In order to satisfy D we can choose a value for x such that $(-\mu(x) \leq \mu(q))$. □

Lemma 2 (Soundness of SMT($\mathcal{LRA}$) Resolution for sets of clauses). Let B_x and B_{-x} two sets of clauses in OCCF form, x appears upper bounded and lower bounded respectively. Then $B_x \cup B_{-x}$ is equisatisfiable with $B_x \otimes_x B_{-x}$.

Proof. Assume $B_x \cup B_{-x}$ and $B_x \otimes_x B_{-x}$ to be of the form:

$$\underbrace{\begin{array}{l} (x \le p_1) \vee C_1 \\ (x \le p_2) \vee C_2 \\ \dots \\ (x \le p_n) \vee C_n \\ \dots \\ (-x \le q_1) \vee D_1 \\ (-x \le q_2) \vee D_2 \\ \dots \\ (-x \le q_m) \vee D_m \end{array}}_{\substack{B_x \cup B_{-x} \\ (n+m)\text{ clauses}}} \qquad \underbrace{\begin{array}{l} (-q_1 \le p_1) \vee C_1 \vee D_1 \\ (-q_2 \le p_1) \vee C_1 \vee D_2 \\ \dots \\ (-q_m \le p_1) \vee C_1 \vee D_m \\ \dots \\ (-q_1 \le p_2) \vee C_2 \vee D_1 \\ (-q_2 \le p_2) \vee C_2 \vee D_2 \\ \dots \\ (-q_m \le p_n) \vee C_n \vee D_m \end{array}}_{\substack{B_x \otimes_x B_{-x} \\ (nm)\text{ clauses}}} \begin{array}{l} [B_{11}] \\ [B_{12}] \\ \dots \\ [B_{1m}] \\ \dots \\ [B_{21}] \\ [B_{22}] \\ \dots \\ [B_{nm}] \end{array}$$

In order to prove that $B_x \cup B_{-x}$ and $B_x \otimes_x B_{-x}$ are equisatisfiable we show that (i) if a model μ for $B_x \cup B_{-x}$ exists, then μ also satisfies $B_x \otimes_x B_{-x}$, and (ii) that any model μ for $B_x \otimes_x B_{-x}$ can be extended with a suitable value in $\mathbb{R}$ for x to satisfy $B_x \cup B_{-x}$.

(i) Suppose $B_x \cup B_{-x}$ is satisfied by a model μ. Let $C_\top = \{C_i \mid \mu \models C_i\}$, $C_\bot = \{C_i \mid \mu \not\models C_i\}$, and let $D_\top = \{D_i \mid \mu \models D_i\}$, $D_\bot = \{D_i \mid \mu \not\models D_i\}$. Then the set of clauses $B_\top = \{B_{ij} \mid C_i \in C_\top \text{ or } D_j \in D_\top\}$ is also satisfied by μ. Consider now the remaining B_{ij} clauses $((B_x \otimes_x B_{-x}) \setminus B_\top)$. These clauses contain constraints in $B_{rest} = \{-q_j \le p_i \mid C_i \in C_\bot, D_j \in D_\bot\}$. However since $B_x \cup B_{-x}$ is satisfied by μ, then all $\{x \le p_i, -x \le q_j \mid C_i \notin C_\top, D_j \notin D_\top\}$ are satisfied by μ, and therefore all their combinations, i.e. B_{rest}, are satisfied by μ, as a consequence of the Fourier-Motzkin theorem.

(ii) Suppose $B_x \otimes_x B_{-x}$ is satisfied by a model μ, and $C_\top$, $C_\bot$, $D_\top$, $D_\bot$, $B_\top$, B_{rest} are defined as in (i). Since $\mu \models B_{rest}$ for hypotesis, and since B_{rest} contains all the combinations of inequalities for $\{p_i \mid C_i \in C_\bot\}$ and $\{-q_j \mid D_j \in D_\bot\}$, then, as a consequence of the Fourier-Motzkin theorem it is possible to extend μ to a μ' with a value for x such that all $\{x \le p_i, -x \le q_j \mid C_i \notin C_\top, D_j \notin D_\top\}$, and therefore $B_x \cup B_{-x}$, are satisfied by μ'. □