Comparing Different Projection Operators in Cylindrical Algebraic Decomposition for SMT Solving

Tarik Viehmann, Gereon Kremer, and Erika Ábrahám

RWTH Aachen University, Germany

Abstract. Satisfiability-modulo-theories (SMT) solving is a technique to check the satisfiability of logical formulas. In the context of SMT solving for non-linear real arithmetic, the cylindrical algebraic decomposition (CAD) can be embedded as a theory solver to solve sets (conjunctions) of polynomial constraints. When developing such a CAD theory solver, a design choice is given by the selection of the projection operator used in the CAD method. In this paper we provide some experimental evaluations to analyse how the choice of the projection operator affects the computational efficiency of SMT solving.

1 Introduction

Satisfiability-modulo-theories (*SMT*) solving [BBH+09,KS08] aims at the fully automated check of the satisfiability of (usually quantifier-free) first-order logic formulas. Most SMT solvers implement algorithms which divide this task into two parts: a *combinatorial* part (responsible for the satisfaction of the Boolean structure of a formula) and a *theory-solving* part (to check the satisfiability of conjunctions of theory constraints).

When solving *quantifier-free non-linear real arithmetic* (*QF_NRA*) formulas, the theory-solving part requires the invocation of a decision procedure to check the satisfiability of conjunctions of polynomial equalities and inequalitites over the real domain. Besides some incomplete algorithms like interval constraint propagation [GGI+10,HR97] or the virtual substitution method [Wei97], the *cylindrical algebraic decomposition* (*CAD*) method [Col75] can be expoited for this purpose.

To check the satisfiability of a conjunction of polynomial constraints (which we call in the following the input formula), the CAD method works in two phases. In its first phase it uses a *projection operator* to generate sets of polynomials of decreasing dimensionality; the sets of real roots of these polynomials are the borders of a finite number of semi-algebraic sets (cells), such that in each cell either all points satisfy the input formula or none of them does so. In its second phase the CAD method identifies a sample point from each of the cells and checks whether the input formula is satisfied by one of those sample points.

In the first phase of the CAD method, different projection operators can be used. Some of them are incomplete, some are complete. In both cases, if one

of the sample points satisfies the input formula then a solution is found. When using a complete projection operator, if none of the sample points satisfies the input formula then the input formula is unsatisfiable. However, if none of the sample points satisfy the input formula, incomplete projection operators might lead to inconclusive results.

Besides completeness, different projection operators differ also in the required computation effort (both for computing the projection and generating the sample points). Naturally, one expects that a complete projection operator needs more computational effort than an incomplete one. The natural question of how large these differences are has been looked at, for example in [McC84], but a conclusive answer has not yet been found. Hence, it is not a priori known how large these differences are, and how different incomplete projection operators relate in this respect.

The aim of this work is to analyse these aspects via experimental evaluation in the SMT solving context, i.e., when the CAD method gets as input only conjunctions of polynomial constraints.

2 Preliminaries

2.1 SMT Solving

SAT-modulo-theories (*SMT*) solvers [BBH+09,KS08] aim at checking the satisfiability of (usually quantifier-free) first-order-logic formulas over an underlying theory (or combined theories [NO79]). Typically, lazy SMT solvers combine a *SAT solver* with one or more *theory solvers*. Thereby the SAT solver handles the input formula's logical structure and is responsible for finding solutions for the *Boolean skeleton* of the input formula, which is gained by substituting fresh Boolean propositions for the theory atoms. To be able to check the consistency of theory atoms, the SAT solver communicates with the theory solvers, which implement decision procedures for the underlying theory.

Figure 1 illustrates the lazy SMT solving framework. The SAT solver iteratively searches for a satisfying solutions for the Boolean skeleton. During its search, it consults the theory solver(s) to check whether the current Boolean assignment is consistent in the theory. To do so, it collects all theory constraints whose abstraction proposition is `true` and appears non-negated in the formula, and those whose abstraction proposition is `false` and appears negated in the formula. The resulting theory constraint set is sent to the theory solver(s), which checks whether it is consistent. In the *full*

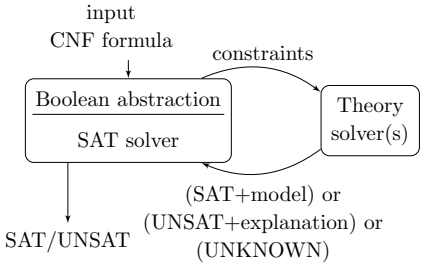


Fig. 1: The SMT solving framework

lazy approach, this communication takes place only for full Boolean solutions, whereas in the *less* lazy approach the theory checks are invoked more frequently also for partial solutions.

If the constraints are consistent in the theory and the SAT solver's assignment is already complete then a satisfying solution is found for the input formula. If the constraints are consistent but the Boolean assignment is not yet complete, the SAT solver continues its search. Otherwise, if the theory constraints are conflicting, the invoked theory solver returns an *explanation* for the conflict. The explanation is often an *infeasible subset* $\{c_1, \ldots, c_n\} \subseteq C$ of the theory solver's input constraints C, which leads to a tautology $(\neg c_1 \vee \ldots \neg c_n)$, whose abstraction can be added to the SAT solver's clause set to exclude this combination of (conflicting) theory constraints from the further search.

First SMT solvers addressed more light-weight theories like equality logic and uninterpreted functions. Aiming at program verification, theories for arrays, bit-vectors and floating-point arithmetic followed. Nowadays there are also highly tuned SMT solvers for linear arithmetic theories. Recent developments allow also to solve quantifier-free non-linear arithmetic (QF_NRA) problems with SMT solvers [JdM12,CKJ+15].

2.2 Non-linear Real Arithmetic

Given a finite set $V = \{x_1, \ldots, x_n\}$ of *variables* and a coefficient ring R, a *polynomial* $p \in R[x_1, \ldots, x_n]$ is an expression of the form $p = \sum_{i=1}^{d} a_i \Pi_{j=1}^{n} x_j^{e_{ij}}$ with $a_i \in R$ and $e_{i,j} \in \mathbb{N}_0$ for all $i = 1, \ldots, d$ and $j = 1, \ldots, n$. The expressions $a_i \Pi_{j=1}^{n} x_j^{e_{ij}}$ are called *terms*, in which $\Pi_{j=1}^{n} x_j^{e_{ij}}$ is a *monomial* and a_i its *coefficient*. If $n = 1$ then we call p *univariate*, otherwise *multivariate*.

Note that a multivariate polynomial $p \in R[x_1, \ldots, x_n]$ in variables $x_1, \ldots, x_n$ and coefficients from R can also be viewed as a univariate polynomial $p \in R[x_1, \ldots, x_{n-1}][x_n]$ in the *main variable* x_n having polynomial coefficients from the polynomial ring $R[x_1, \ldots, x_{n-1}]$.

We use the following properties of a polynomial p with the usual meaning: the *degree* $deg(p)$ and *total degree* $tdeg(p)$, as well as the *leading term* $ldt(p)$ and the *leading coefficient* $lcf(p)$. The *reductum* $red(p)$ of a non-zero polynomial p is $p - ldt(p)$; for $p = 0$ we define $red(0) = 0$. The kth reductum of p is defined recursively by $red^0(p) = p$, $red^j(p) = red(red^{j-1}(p))$.

QF_NRA formulas are Boolean combinations of *polynomial constraints* $p \sim 0$, where p is a polynomial and $\sim \in \{<, \leq, =, \neq, \geq, >\}$ is a comparison operator. The QF_NRA *satisfiability problem* poses the question whether we can assign a real value to each of the variables in a QF_NRA formula such that the formula evaluates to true (under the standard semantics of arithmetic operators). The satisfiability problem for QF_NRA is decidable [Tar48], but it has an exponential time-complexity even for conjunctions of polynomial (in)equalities (in contrast to its linear fragment, in which conjunctions of inequalities are solvable in polynomial time). In this work we will use the *cylindrical algebraic decomposition* (*CAD*) method [Col75] as a QF_NRA theory solver module in the SMT solving context.

2.3 Cylindrical Algebraic Decomposition

The *sign* of a polynomial $p \in \mathbb{Z}[x_1, \ldots, x_n]$ under some $r \in \mathbb{R}^n$ is either -1 if r evaluates p to a negative value, 1 if the evaluation results in a positive value, and 0 otherwise. We say that p is *sign-invariant* over a set $R \subseteq \mathbb{R}^n$ if the sign of p is the same under all points in R. A set $P_n \subseteq \mathbb{Z}[x_1, \ldots, x_n]$ of polynomials is said to be *sign-invariant* over $R \subseteq \mathbb{R}^n$ if each polynomial in P_n is sign-invarint over R; the set R is then called a P_n*-sign-invariant* set.

A partition $\mathcal{C} = \{C_1, \ldots, C_k\}$ of $\mathbb{R}^n$ (with $C_i \subseteq \mathbb{R}^n$, $C_i \cap C_j = \emptyset$ for all $i \neq j$, and $\cup_i C_i = \mathbb{R}^n$) is called a *cylindrical algebraic decomposition* (*CAD*) of $\mathbb{R}^n$ if each $C_i \in \mathcal{C}$ is a connected semi-algebraic[1] set, and for each $C_i, C_j \in \mathcal{C}$ and each $1 \leq k < n$ the projections of C_i and C_j to $\mathbb{R}^{n-k}$ (by removing the last k coordinates) are either disjoint or identical. We call the sets C_i the *cells* of the CAD. Intuitively, the last condition assures that the cells of the decomposition are cylindrically ordered, i.e., that the projections of the cells C_i to $\mathbb{R}^n$ define a CAD of $\mathbb{R}^{n-1}$.

The cylindrical algebraic decomposition method takes as input a set $P_n \subseteq \mathbb{Z}[x_1, \ldots, x_n]$ of polynomials and *sign conditions*[2] for them[3], and computes a CAD for $\mathbb{R}^n$ such that the CAD cells are all P_n-sign-invariant. That means, it suffices to take a representative sample point from each cell and test whether it satisfies the sign conditions for the input polynomials.

The CAD method works in two phases: the *projection phase* and the *construction phase*. In the projection phase, a *projection operator* is used to compute the polynomials that describe the boundaries of the CAD cells. Starting with the input polynomials $P_n \subseteq \mathbb{Z}[x_1, \ldots, x_n]$, the projection operator is applied to P_n to compute a set $P_{n-1} \subseteq \mathbb{Z}[x_1, \ldots, x_{n-1}]$ of polynomials, for which a CAD is computed recursively. The projection operator has the important property that each CAD $\mathcal{C}'$ for P_{n-1} can be extended to a CAD $\mathcal{C}$ for P_n by defining the cells of $\mathcal{C}$ to be the P_n-sign-invariant regions in the cylinders $C'_i \times \mathbb{R}$ for each cell $C'_i \in \mathcal{C}'$.

2.4 Projection operators

Assume two multivariate polynomials $p, q \in \mathbb{Z}[x_1, \ldots, x_n]$, which can also be seen as univariate polynomials $p = \sum_{i=0}^{k} a_i x_n^i$ and $q = \sum_{i=0}^{l} b_i x_n^i$ in x_n with polynomial coefficients $a_i, b_i \in \mathbb{Z}[x_1, \ldots, x_{n-1}]$. Assume furthermore that $k, l \geq 1$ with $a_k \neq 0$ and $b_l \neq 0$, i.e., $deg(p) = k \geq 1$ and $deg(q) = l \geq 1$ are the degrees of the polynomials in x_n. The *Sylvester matrix* of p and q (with respect to x_n)

[1] A set $R \subseteq \mathbb{R}^n$ is *semi-algebraic* if it can be described by a conjunction of polynomial constraints.

[2] A sign condition for a polynomial p is a polynomial constraint $p \sim 0$.

[3] In general it could be a real arithmetic formula, but in the SMT context we focus of polynomial constraint sets.

is the following $(k+l) \times (k+l)$-matrix.

$$Syl_{x_n}(p,q) := \begin{pmatrix} a_k & \cdots & a_0 & & \\ & a_k & \cdots & a_0 & \\ & & \ddots & & \ddots \\ & & & a_k & \cdots & a_0 \\ b_l & \cdots & b_0 & & \\ & b_l & \cdots & b_0 & \\ & & \ddots & & \ddots \\ & & & b_l & \cdots & b_0 \end{pmatrix} \begin{matrix} \left.\vphantom{\begin{matrix}a\\a\\a\\a\end{matrix}}\right\} \text{l} \\ \left.\vphantom{\begin{matrix}a\\a\\a\\a\end{matrix}}\right\} \text{k} \end{matrix}$$

For some $0 \leq i \leq j \leq k+l$ let $M_{i,j}$ be the the matrix obtained from $Syl_{x_n}(p,q)$ by deleting the last j rows of p coefficients, the last j rows of q coefficients and the last $2j+1$ columns except the column $m+n-i-j$.

- The *j-th subresultant* of p and q is defined as $S_j(p,q) = \sum_{i=0}^{j} det(M_{j,i}) x_n^i$.
- The *j-th principal subresultant coefficient* of p and q is $pcs_j(p,q) = det(M_{j,j})$ (i.e., the leading coefficient of $S_j(p,q)$).
- The *resultant* of p and q is defined as $res(p,q) = det(Syl_x(p,q)) = psc_0(p,q)$.
- The *pcs set* of p and q is defined as

$$PSC(p,q) = \{pcs_j(p,q) \mid 0 \leq j \leq min(k,l),\ pcs_j(p,q) \neq 0\}.$$

- The *reducta set* of p is defined as

$$RED(p) = \{red^i(p) \mid 0 \leq i \leq k,\ red^i(p) \neq 0\}.$$

Assume in the following a set $P_n = \{p_1, \ldots, p_m\} \subseteq \mathbb{Z}[x_1, \ldots, x_{n-1}][x_n]$ of polynomials, which we view as polynomials in x_n with polynomial coefficients from $\mathbb{Z}[x_1, \ldots, x_{n-1}]$. Let furthermore d be an upper bound on the degrees of the polynomials p_i in the main variable x_n. A projection operator takes P_n as input and generates a set $P_{n-1} \subseteq \mathbb{Z}[x_1, \ldots, x_{n-1}]$ of polynomials as output.

Next we describe the projection operators whose efficiency we will compare in Section 3. Due to space limitations, we refer for more detailed explanations to the references given below.

Collins' projection operator [Col75]

$$\begin{aligned} projC(P_n) &= projC_1(P_n) \cup projC_2(P_n) \\ projC_1(P_n) &= \bigcup_{p_i \in P_n} \bigcup_{r \in RED(p_i)} (\{ldcf(r)\} \cup PSC(r, r')) \\ projC_2(P_n) &= \bigcup_{1 \leq i < j \leq m} \bigcup_{\substack{r_1 \in RED(p_i) \\ r_2 \in RED(p_j)}} PSC(r_1, r_2) \end{aligned}$$

Using Collins' operator in the CAD method, a P_n-sign-invariant CAD of $\mathbb{R}^n$ can be established. The number of polynomials in $projC(P_n)$ is dominated by m^2d^3. Note that Collins' operator has been replaced by Hong's operator below without any drawback. As the first projection operator, we include it nevertheless in this comparison to give an impression on the progress made.

Hong's projection operator [Hon90] Hong showed that within $ProjC_2$, the added *psc* sets have redundancies and proposed the following operator:

$$projH(P_n) = projC_1(P_n) \cup projH_2(P_n)$$
$$projH_2(P_n) = \bigcup_{1 \le i < j \le m} \bigcup_{r \in RED(p_i)} PSC(r, p_j)$$

The number of polynomials included in $projH(P_n)$ is dominated by m^2d^2, which implies a possible significant impact on the overall performance compared to Collins' operator, since the degree bound impacts the size in quadratic instead of cubic order.

McCallum's projection operator [McC85] Let $P'_n = \{p'_1, \ldots, p'_m\}$ be the finest square-free basis of P_n.

$$projM(P_n) = projM_1(P'_n) \cup projM_2(P'_n) \cup \bigcup_{p_i \in P_n} \{cont_{x_\tau}(p_i)\}$$
$$projM_1(P'_n) = \bigcup_{p_i \in P'_n} \left(\{discr(p_i)\} \cup \bigcup_{r \in RED(p_i)} \{ldcf(r)\} \right)$$
$$projM_2(P'_n) = \bigcup_{1 \le i < j \le m} \{res(p_i, p_j)\}$$

Although it seems substantially different to the previous operators, McCallum showed that one could also redefine Collins' operator to operate with subdiscriminants rather than subresultants such that $projM \subseteq projC$ (see Chapter 3.1 in [McC85]). McCallum's operator not only requires the use of finest square-free basis in each step, but it can only guarantee the correctness of a CAD procedure with his operator when the input polynomials are well-oriented. In addition to this restriction, the finite amount of real roots of the well oriented polynomials in the set $projM(A)$ requires a slight modification of the lifting phase to preserve the order-invariance in every step. McCallum also provided incomplete methods to check whether a polynomial has finitely many zeros or not. It is also possible to detect if the input polynomials were not well-oriented during lifting phase, if one keeps track while creating sample points whether they present a space of positive dimension or not. If a sample point $p \in \mathbb{R}^i$ that represents a cell of positive dimension – thus was chosen to represent a sector in some lower dimensional lifting step – causes a $(i+1)$-variate projection polynomial to vanish, then this polynomial has an infinite number of zeros: all sample points that could

have been chosen instead of the one that lead to p in this lower dimensional lifting step. McCallum further argued that the vast majority of polynomials are well-oriented including all polynomials with at most three variables.

Brown's projection operator While McCallum's operator produces order-invariant CADs which might be needed in some scenarios, it is not required to solve QF_NRA formulas, because only the sign-invariance is mandatory there. Brown showed that it is sufficient for the $projM_1$ operator to only contain the leading coefficient of each polynomial instead of all coefficients. This however needs to come along with further modifications in the lifting stage.

Let again $P'_n = \{p'_1, \dots, p'_m\}$ be the finest square-free basis of P_n.

$$projB(P_n) = projB_1(P'_n) \cup projM_2(P'_n) \cup \bigcup_{p_i \in P_n} \{cont(p_i)\}$$

$$projB_1(P'_n) = \bigcup_{p_i \in P_n} \{discr(p_i)\} \cup \{ldcf(p_i)\}$$

Brown also made another modification to the lifting stage, that allows the CAD to stay correct in some cases, where the polynomials are not well-oriented. Note that essentially $projB \subseteq projM \subseteq projH \subseteq projC$.

Further projection operators There exist a number of further projection operators that we do not consider in this paper. Lazard [Laz94] improved on McCallum's operator, however the proof contained an error that was only recently fixed by [MH16]. Seidl and Sturm [SS03] proposed an specialized version of Hong's operator for what they call a partial CAD targeting inputs with many parameters. Motivated by applications in SMT solving there are also results on how to compute projections only locally for specific cells, for example by Strzeboński [Str16] or Brown and Košta [BK15].

3 Experimental results

We implemented the four projection operators presented in the previous section in the SMT solver `SMT-RAT` [CKJ+15]. The test setting was based on `SMT-RAT`'s CAD theory module, which uses some optimizations, including the following. Constants get removed instantly and are not projected down. The same holds for positive definite and negative definite functions, as they are always sign- and order-invariant. It does however not provide the calculation of a square-free basis yet – and thus this restriction of McCallum's and Brown's operator is ignored – and also does not attempt to fix the projection if a polynomial vanishes during lifting. Both are under active development and some preliminary results are presented at the end.

While `SMT-RAT` can be run in an incremental fashion and thus can deal with backtracking scenarios by offering ways of adding and removing polynomials on each projection level, this feature was not utilized here as the study of interest

Operator	Collins	Hong	McCallum	Brown
Completed projections	5698	6052	6828	6828

Table 1: Number of completed projections

	average	median	75% qt	max
Number of polynomials	≈ 6,37	6	8	34
Max degree	≈ 5,23	3	6	44
Max total degree	≈ 6,06	4	7	44

Table 2: Benchmark structure (qt=quartile), 5698 instances

is the efficiency of the operators by the theoretical improvements, that were made on the field of CAD. See section 3.2 for more details on how incremental projection works in `SMT-RAT`. Note that we do not look at the question of variable ordering here. It is well established that choosing a proper variable ordering can have a huge impact on the projection. A comparison of projection operators using a variable ordering tuned to the specific projection operator would be very interesting, but we have not done this yet. As for this paper, we use a static variable ordering that is essentially the order in which the variables are declared in the input formula.

We used non-linear benchmarks from the SMT competition 2014 and instantiated `SMT-RAT` in four variants, using the four different projection operators in its CAD theory solver.

3.1 CAD projection only

As a first experiment, the CAD modules were programmed to project all polynomials of an SMT problem instance down to the univariate level and then stop. The tests were run on an AMD Opteron 6172 with a time limit of 60 seconds. Table 1 shows for how many benchmarks the projections could be completed within the time limit, when using the different projection operators.

In a next step we identified those benchmarks for which the projection could be completed within the time limit using all four operators. These benchmarks are exactly those which Collins' operator was able to handle. The performance of the four participants on these benchmarks was observed more closely. As factors having a major implact on performance, the number of different polynomials, their maximum degree (in the respective main variable) and their maximum total degree over all polynomials were identified. Note that other factors like the number of terms or the coefficient size were not considered here. This information is shown in Table 2, based on the average and the median, as well as the 75% quartile, being the largest data point that is smaller than 25% of the data and the maximum.

From the theoretical analysis, we expect to have a clear order on the projection operators as we established $projB \subseteq projM \subseteq projH \subseteq projC$. We

Number of polynomials	average	median	75% qt	max
Collins	≈ 10,89	6	16	99
Hong	≈ 8,62	5	12	57
McCallum	≈ 6,06	4	8	31
Brown	≈ 5,29	4	7	25
Max degree	average	median	75% qt	max
Collins	≈ 7,06	1	10	182
Hong	≈ 7,05	3	10	182
McCallum	≈ 6,05	2	8	182
Brown	≈ 6,04	2	8	182
Max total degree	average	median	75% qt	max
Collins	≈ 7,76	4	10	182
Hong	≈ 7,75	4	10	182
McCallum	≈ 6,68	3	8	182
Brown	≈ 6,68	3	8	182

Table 3: Results on projection level 1 (5693 instances)

also know that using McCallum's (and Brown's) operator have the theoretical drawback of incompleteness. Therefore we try to measure how significant the individual gaps are and whether the incompleteness actually occurs in practice. Unfortunately, only a small number of our benchmarks contain more than three variables and only these could reveal this incompleteness.

These questions cover the performance of the different projection operators in a single CAD computation, but it is not at all clear whether these results immediately transfer to the SMT setting. Here, multiple incremental calls are performed and – if the problem turns out to be satisfiable – the computation can be aborted if a single satisfying sample is found. How the different operators perform within a full SMT solver is thus analyzed separately.

We tracked the number of polynomials, the maximum degree and the maximum total degree for the first 4 projection levels and got the results depicted in the Tables 3, 4, 5 and 6. Only few solved instances caused projections with more then 5 variables, which is why this restriction is made. Note that the numbers for the maximum degree and the maximum total degree are essentially the same for all projection levels, only on the first level a slight difference exists. This shows that most benchmarks have a rather simple structure as most polynomials are univariate and the multivariate polynomials never have the maximum degree.

We also noticed that with Brown's operator, at times a whole projection level could be skipped. This can happen if a variable is technically present, but does not contribute to any polynomial that actually needs to be considered for projection. This was the case for 11 instances on projection level 1 and for 137 instances on projection level 2. The improvements of Brown's operator compared to McCallum's are rather small and they can handle the exact same benchmarks. This however is no big surprise, as the real strength of Brown's operator is in

Number of polynomials	average	median	75% qt	max
Collins	≈ 783,14	11	116	24489
Hong	≈ 158,77	6	43	6056
McCallum	≈ 16,74	5	15	227
Brown	≈ 11,62	4	12	168
Max degree	average	median	75% qt	max
Collins	≈ 26,32	6	26	419
Hong	≈ 26,16	6	26	419
McCallum	≈ 13,29	3	16	264
Brown	≈ 13,48	3	16	264
Max total degree	average	median	75% qt	max
Collins	≈ 26,39	6	26	419
Hong	≈ 26,23	6	26	419
McCallum	≈ 13,34	3	16	264
Brown	≈ 13,53	3	16	264

Table 4: Results on projection level 2 (5583 instances)

the lifting phase, where the amount of lifting points are kept smaller due to the manual adding of points, only when it is necessary.

Looking at the statistics that regard polynomial degrees, Collins' and Hong's operators scored similar results. The same can be seen, when comparing McCallum's operator to the one of Brown. However, there is a major gap between these two groups (Collins' approach vs McCallum's design). This indicates the huge step, that McCallum's operator was able to make in the context of CAD.

Though a somewhat poor performance of Collins's operator was to be expected, it performs much worse than we expected. The increase in the number of polynomials from level 1 to level 2 was about 72 while Hong's operator kept it at about 18 and both McCallum's and Brown's operator stayed below 3. Also the maximum degree is much higher for Collins's operator compared to McCallum's while there is no significant difference between Collins's and Hong's operator or McCallum's and Brown's operator. Altogether, Hong's operator seems to be a viable possibility for cases where McCallum's operator is not applicable as it at least reduces the number of polynomials significantly.

3.2 Full SMT solver

In a second test run, the previously selected 5698 samples were given to the full SMT solver. The results are shown in Table 7 with a timeout of again 60 seconds. Note that in this scenario, most benchmarks would not cause a single call to the CAD module but multiple theory calls due to the Boolean structure of the problems. We can see that only a relatively small amount of benchmarks is above the timeout, indicating that `SMT-RAT` does a reasonably good job in exploiting the incremental nature of these theory calls. The previous scenario

Number of polynomials	average	median	75% qt	max
Collins	≈ 117,00	3	10	22806
Hong	≈ 20,20	3	9	2009
McCallum	≈ 5,06	2	7	80
Brown	≈ 4,65	2	7	73
Max degree	average	median	75% qt	max
Collins	≈ 11,79	1	4	286
Hong	≈ 11,66	1	4	286
McCallum	≈ 5,25	1	4	64
Brown	≈ 5,03	1	4	64
Max total degree	average	median	75% qt	max
Collins	≈ 11,82	1	4	286
Hong	≈ 11,68	1	4	286
McCallum	≈ 5,28	1	4	64
Brown	≈ 5,06	1	4	64

Table 5: Results on projection level 3 (789 instances)

also considered all polynomials present in the problem while here, the theory solver will oftentimes never see the full set of constraints as a subset of these is enough to satisfy the formula or to establish unsatisfiability.

Furthermore, `SMT-RAT` interleaves the projection and lifting phases so that a satisfying assingment can be found before the projection is finished. To do that, the projection operator is deconstructed into what we call *projection candidates*, for example the resultant of two polynomials. Whenever a projection candidate is calculated, new polynomials may appear one level below and thereby spawn new projection candidates on this level. All these projection candidates can be processed individually, thus the projection is decomposed into many individual tasks. After every projection candidate, lifting is continued on the incomplete projection hoping that a satisfying assignment is found.

This can speed up the overall solving time significantly, but also has some other effects that impede gaining any insight about the projection operators in this scenario. Note that the projection operators not only produce different polynomials, but may also generate the same polynomials in a different order. Every newly produced polynomial induces new sample points which may be satisfying and thus may immediately terminate the solving process. Hence, the purely heuristic decisions of projection order – the order in which projection candidates are processed – and lifting order – the order in which sample points are lifted – can easily supersede any impact of the projection operator. Thus it is even plausible that generating more polynomials could have a beneficial effect, at least on satisfiable problems.

Close analysis within the SAT solver enables `SMT-RAT` to filter out constraints based on the Boolean structure in many cases. In fact, some of the benchmarks, that caused the greatest amount of polynomials in the first test run, turned out to

Number of polynomials	average	median	75% qt	max
Collins	≈ 15,61	4	11	302
Hong	≈ 10,27	4	8	106
McCallum	≈ 7,91	3	6	29
Brown	≈ 5,46	3	5	26
Max degree	average median	75% qt	max	
Collins	≈ 4,02	2	5	32
Hong	≈ 3,84	2	5	24
McCallum	≈ 3,43	1	4	16
Brown	≈ 2,74	1	4	16
Max total degree	average	median	75% qt	max
Collins	≈ 5,25	2	6	32
Hong	≈ 5,07	2	6	24
McCallum	≈ 3,84	2	4	16
Brown	≈ 3,48	2	4	16

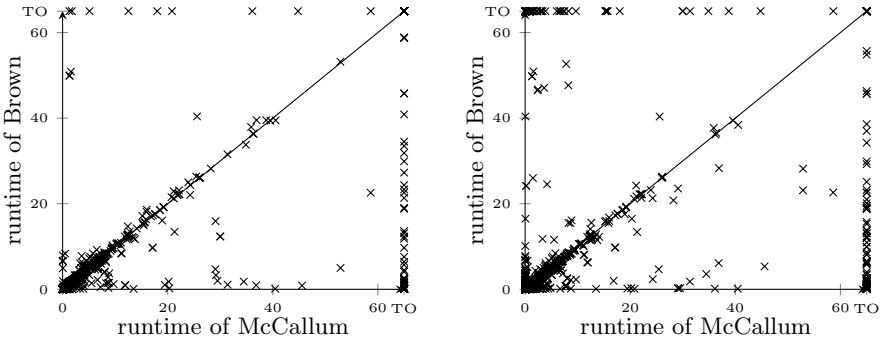
Table 6: Results on projection level 4 (147 instances)

Operator	Solved	Timeout	average	median	75% qt	max
Collins	5041	657	≈ 452,80	32	47	56538
Hong	5125	573	≈ 233,30	36	55	48504
McCallum	5284	414	≈ 216,38	45	72	38727
Brown	5299	399	≈ 220,11	35	52	49497

Table 7: Statistics of full SMT-procedure, including runtimes in ms for solved instances

be solvable in a very short amount of time. One example is the benchmark "meti-tarski/polypaver-bench-sqrt-3d-chunk-0479.smt2". Generating a total amount of 24575 polynomials in a full projection, it could be solved in 33 milliseconds due to some contradicting bounds.

The running times on the benchmarks that could be finished are given in the last four columns of Table 7. This data shows that the above effects dominate the whole solving process. More than 75% of the solvable examples are solved in less than 75 milliseconds with any projection operator, Collins's even being the fastest for the 75% quartile. Note that the four solvers solve different benchmarks, for example the solver using McCallum's operator solves instances that Brown's operators cannot cope with and vice-versa. Hence these numbers are a bit skewed. Though it may be surprising that Brown's operator only has a small lead over McCallum's, this can be explained by the structure of the inputs. Brown's operator eliminates coefficients from the projection, but as we have seen in the comparison between the maximum degree and the maximum total degree, the coefficients will usually have a small degree. Furthermore, removing these coefficients of small degree may actually hinder the solver. It may be sufficient to



(a) Default ordering (b) Resultants deferred in Brown

Fig. 2: Running times in seconds of Brown vs. McCallum

consider these coefficients to generate a satisfying solution so that we can spare the effort of computing resultants and discriminants.

3.3 McCallum vs. Brown

We now want to compare McCallum's and Brown's operator in more detail. First we compared the overall performance of the solvers with McCallum's and Brown's operator in Figure 2a where every cross represents a single SMT benchmark. As expected the solvers behave rather similarly and Brown's operator tends to solve more instances, but still a notable number of outliers exist for both directions. It seems that in a full SMT solver, being lucky in the heuristics can be more important than the choice of the projection operator. This suggests that running multiple different projections in parallel could be beneficial. Note that if the projection order of McCallum's operator is forced to mirror the one of Brown's operator – that is the coefficients are projected last – both behave almost identically.

In order to make use of a portfolio approach, several strategies that behave as differently as possible are desirable. We tried several modifications to the projection order, including deferring the projection of resultants as they tend to lead to the greatest increase of polynomial degrees. Figure 2b shows the two solver configurations that diverged most: McCallum with our regular projection order that is mostly based on the polynomial degree, and Brown's operator deferring resultant computations.

As for the question whether the incompleteness of McCallum's operator is a problem in practice, we performed a test on 5889 benchmarks from the SMT competition, being all those that were not trivially solved before even reaching the CAD module. In 510 cases some polynomial vanished and 353 from those were found to be satisfiable. Note that a vanishing polynomial does not imply that the projection is incomplete: if the polynomial vanished over a zero-

dimensional cell, this could be repaired by adding delineating polynomials. The remaining 157 were found to be unsatisfiable, and all of them are in fact unsatisfiable. Hence we can conclude that McCallum's operator indeed produces incomplete projections for actual benchmarks, but they did not cause a single error on our benchmark set.

We are working on using CoCoALib [AB10] to compute a square-free basis in the projection and present some the preliminary results here. The solver becomes slower by about 10% on average due to the additional effort spent in computing the square-free basis and converting the polynomials. Nevertheless, it manages to solve more problems than without these computations. Overvall, this seems to indicate that for many benchmarks this computations is superfluous, but it has a great impact for some other benchmarks.

Altogether, we can recognize the theoretically expected order of the different projection operators with respect to the number of solved instances, though it may not be as significant as expected and is not directly reflected in the observed solving times. While Hong's operator is a significant improvement over Collins's and McCallum's again improves on Hong's operator, the difference between McCallum's and Brown's operator is small, though existent.

4 Conclusion

In this paper we provided some experimental results to compare the efficiency of different projection operators in the cylindrical algebraic decomposition method, both in a isolated theory solver and within a full SMT solver. In the isolated setting, the practical analysis showed that the theoretical results hold on practical examples. In a full SMT solver however, other effects tend to dominate for individual instances, but the overall trend is still evident.

Future work might include a closer look at the side cases, when using Brown's projection on examples, where points need to be added to the projection. Also the runtimes on the tests, where Brown's operator competed versus McCallum's operator, suggest a dual approach employing multiple projection operators in parallel.

Another quite promising approach that is not yet considered is to use equational constraints to simplify the CAD procedure as described in [BDE+16].

References

[AB10] John Abbott and Anna M. Bigatti, *CoCoALib: A C++ Library for Computations in Commutative Algebra... and Beyond*, pp. 73–76, Springer, 2010.

[BBH+09] A. Biere, A. Biere, M. Heule, H. van Maaren, and T. Walsh, *Handbook of satisfiability*, Frontiers in Artificial Intelligence and Applications, vol. 185, IOS Press, 2009.

[BDE+16] Russell Bradford, James H Davenport, Matthew England, Scott McCallum, and David Wilson, *Truth table invariant cylindrical algebraic decomposition*, Journal of Symbolic Computation **76** (2016), 1–35.

[BK15] Christopher W. Brown and Marek Košta, *Constructing a single cell in cylindrical algebraic decomposition*, Journal of Symbolic Computation **70** (2015), 14 – 48.

[CKJ+15] Florian Corzilius, Gereon Kremer, Sebastian Junges, Stefan Schupp, and Erika Ábrahám, *SMT-RAT: An open source C++ toolbox for strategic and parallel SMT solving*, Proc. of SAT'15, LNCS, vol. 9340, Springer, 2015, pp. 360–368.

[Col75] G. E. Collins, *Quantifier elimination for real closed fields by cylindrical algebraic decomposition*, Automata Theory and Formal Languages, LNCS, vol. 33, Springer, 1975, pp. 134–183.

[GGI+10] Sicun Gao, Malay Ganai, Franjo Ivančić, Aarti Gupta, Sriram Sankaranarayanan, and Edmund M. Clarke, *Integrating ICP and LRA solvers for deciding nonlinear real arithmetic problems*, Proc. of FMCAD'10, IEEE, 2010, pp. 81–90.

[Hon90] Hoon Hong, *An improvement of the projection operator in cylindrical algebraic decomposition*, ISSAC '90 Proceedings of the international symposium on Symbolic and algebraic computation (1990), 261–264.

[HR97] S. Herbort and D. Ratz, *Improving the efficiency of a nonlinear-system-solver using a componentwise Newton method*, Tech. Report 2/1997, Inst. für Angewandte Mathematik, University of Karlsruhe, 1997.

[JdM12] D. Jovanović and L. de Moura, *Solving non-linear arithmetic*, Proc. of IJCAR'12, LNAI, vol. 7364, Springer, 2012, pp. 339–354.

[KS08] Daniel Kroening and Ofer Strichman, *Decision procedures: An algorithmic point of view*, Springer, 2008.

[Laz94] D Lazard, *An improved projection for cylindrical algebraic decomposition*, Algebraic geometry and its applications: collections of papers from Shreeram S. Abhyankar's 60th birthday conference, Springer Verlag, 1994, p. 467.

[McC84] Scott McCallum, *An improved projection operation for cylindrical algebraic decomposition (computer algebra, geometry, algorithms)*, Ph.D. thesis, The University of Wisconsin - Madison, 1984, AAI8500835.

[McC85] ———, *An improved projection operation for cylindrical algebraic decomposition*, Tech. report, University of Wisconsin Madison, 1985.

[MH16] Scott McCallum and Hoon Hong, *On using lazard's projection in cad construction*, Journal of Symbolic Computation **72** (2016), 65 – 81.

[NO79] Greg Nelson and Derek C Oppen, *Simplification by cooperating decision procedures*, ACM Transactions on Programming Languages and Systems **1** (1979), no. 2, 245–257.

[SS03] Andreas Seidl and Thomas Sturm, *A generic projection operator for partial cylindrical algebraic decomposition*, Proceedings of the 2003 International Symposium on Symbolic and Algebraic Computation (New York, NY, USA), ISSAC '03, ACM, 2003, pp. 240–247.

[Str16] Adam Strzeboński, *Cylindrical algebraic decomposition using local projections*, Journal of Symbolic Computation **76** (2016), 36 – 64.

[Tar48] Alfred Tarski, *A decision method for elementary algebra and geometry*, Tech. report, University of California Press, 1948.

[Wei97] Volker Weispfenning, *Quantifier elimination for real algebra - the quadratic case and beyond*, Appl. Algebra Eng. Commun. Comput. **8** (1997), no. 2, 85–101.