Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 334 (2018) 3–16

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

Experimental Evaluation of Numerical Domains for Inferring Ranges

Gianluca Amato Marco Rubino

Università di Chieti-Pescara, Pescara, Italy

Abstract

Among the numerical abstract domains for detecting linear relationships between program variables, the polyhedra domain is, from a purely theoretical point of view, the most precise one. Other domains, such as intervals, octagons and parallelotopes, are less expressive but generally more efficient. We focus our attention on interval constraints and, using a suite of benchmarks, we experimentally show that, in practice, polyhedra may often compute results less precise than the other domains, due to the use of the widening operator.

Keywords: Static analysis, abstract interpretation, numerical domains, polyhedra, widening.

1 Introduction

Many numerical abstract domains have been defined in the literature with the aim of discovering relations among numerical variables in imperative programs. These abstract domains differ on the shape and number of constraints on program variables which may be represented. The most common numerical domains are the interval [14], octagon [20] and polyhedra [15] abstract domains.

The domain Int of intervals encodes a finite set of constraints of the form $x \leq u$ or $l \leq x$, where x is a variable of the program and l, u are numbers representing lower and upper bounds respectively. This is a classical example of *non-relational domain*, since it is not able to explicitly represent relationships between two different program variables.

On the contrary, the domain Poly of polyhedra is able to represent any finite set of linear constraints on the program variables. Each linear constraint has the form $\boldsymbol{a} \cdot \boldsymbol{x} \leq u$, while a finite number of them may be represented as $A\boldsymbol{x} \leq \boldsymbol{u}$, where A is the matrix of coefficients, $\boldsymbol{x}$ is the vector of program variables and $\boldsymbol{u}$ the vector of

[1] Email: {gianluca.amato@unich.it, marco.rubino@unich.it}

https://doi.org/10.1016/j.entcs.2018.03.002

upper bounds. Poly is a *relational domain*, since each constraint may involve many different variables.

Finally, the domain Oct of octagons lies in-between the other two: it may represent constraints with up to two program variables, but only of the form $ax + by \leq u$ with $a, b \in \{-1, 0, 1\}$. These are called *octagonal constraints*. Oct is a typical example of the so-called *weakly relational domains*. This class of domains allows to represent constraints involving different variables, but the form of constraints is severely limited.

The interval and octagon abstract domains are also examples of *template abstract domains* [22]. This is the family of all the domains which may represent any linear constraints $\boldsymbol{a} \cdot \boldsymbol{x} \leq u$, but the set of available coefficient vectors $\boldsymbol{a}$ is chosen *a priori*, and cannot change during the analysis. Each abstract object may be represented as $A\boldsymbol{x} \leq \boldsymbol{u}$ where A is the *template* matrix and $\boldsymbol{u}$ the vector of upper bounds. It may seem the same as polyhedra, but the fundamental difference is that in the polyhedra abstract domain the coefficient matrix A may vary freely during the analysis, while in a template abstract domain the template matrix A is fixed. For example, for the interval domain, A is the matrix $(I \mid -I)^T$.

The widespread use of these domains is due, at least in part, to the fact that they are implemented in two of the most famous libraries for numerical abstract domains, namely APRON [18] and PPL [12].

There are many other template abstract domains in the literature, and there are also examples of domains that do not fall in this category, although they are less precise than polyhedra. For example, TVPI [23] (two variables per linear inequality) may represent a finite set of constraints of the form $ax + by \leq u$ without any limitation on a and b. This is a weakly relational abstract domain, since it may encode relationships between two variables only, but not a template domain. Another example is the domain Par of parallelotopes [8], implemented in the Jandom static analyzer [3]. Abstract objects in this domain can be represented as $A\boldsymbol{x} \leq \boldsymbol{u}$ like for the polyhedra domain, but it is required that the coefficient matrix A is invertible. Therefore, parallelotopes definitely have a limited expressive power if compared to polyhedra, although they do not fall in the class of template or weakly relational domains. There exists a template variant of the domain of parallelotopes [4,5], but it is not be used in this paper.

Precision of abstract domains

The aim of this paper is to experimentally compare a selection of abstract domains, including intervals, octagon, polyhedra and parallelotopes, from the point of view of the attainable precision of the analysis. Comparing the precision of abstract domains at the theoretical level is difficult, because a greater expressive power of a domain does not always produce a more precise overall analysis.

A numerical abstract domain is formalized as a set A of abstract properties preordered by $\leq_A$ and endowed with a monotone (w.r.t. the subset ordering in $\wp(\mathbb{R}^n)$) *concretization map* $\gamma : A \to \wp(\mathbb{R}^n)$, where n is the number of program variables. We say that domain A is more expressive than B, and we write $A > B$, if the image of

γ_A is a strict superset of the image of γ_B.

According to the expressive power, we have Poly > Oct > Int and Poly > Par > Int, while octagons and parallelotopes are incomparable. Obviously, the more expressive a domain is, the more accurately it may track, at least in principle, the values of program variables during the program execution. If we compare two analyses performed using the domains A and B with $A > B$, we would expect the analysis using A to find more constraints or more precise bounds than the analysis using B. However, expressiveness of a domain does not tell the whole story. At least other two factors may influence the result of the analysis: abstract operators and widening.

If abstract operators are not the best correct approximations of the concrete ones, the result of the analysis may not be as precise as it could. While abstract operators for intervals, polyhedra and octagons are generally implemented as the best correct abstraction of the concrete operators, this does not happen for parallelotopes. The reason is that for many operators on parallelotopes there is no best correct abstraction. This is because, given a subsets of $\mathbb{R}^n$, in general there is no least parallelotope which approximates it, but there are many minimal competing ones, and heuristic considerations are used to choose among different minimal possibilities. Note that the same also happens for polyhedra (for example, there is no best polyhedral approximation for a sphere), but this is not a big problem for static analysis since most of the operations used in this context transform polyhedra into polyhedra (therefore, abstract operators in Poly are mostly γ-complete [7]).

However, a greater impact on the precision of an analysis is arguably given by widening. For template domains, the implementation of widening is straightforward: since the number and form of constraints is fixed and finite, we just need to enlarge bounds to infinity to force termination of the analysis. However, for non-template domains, constraints may freely change at each iteration. Therefore, widening operators have more freedom and may try different solutions to determine a stable set of constraints. For the polyhedra domain, the most common widenings in use are the so-called *standard widening* described in [15] and later refined in [17], and the widening described in [11]. In the following, they will be called the H79 and BHRZ03 widening, adopting the names used in the PPL.

The widening H79 maintains all the constraints of the polyhedra in the previous iteration, under the condition that the constraint is satisfied by all the points in the polyhedra of the current iteration. The widening BHRZ03 improves on the standard widening by combining four different heuristic techniques, derived from upper bound operators. Both widenings present cases where they lose precision in such a way that the resulting analysis is less precise than what may be attained even with the much simpler interval domain. A detailed example appears in [21].

The widening on parallelotopes differs from the ones on polyhedra, since it dynamically chooses the whole coefficient matrix to be used for the result, either the one of the preceding iteration or the one of the new iteration, according to an heuristic which evaluates the *distance* between the parallelotopes in two successive iterations.

Performance of abstract domains

From the theoretical point of view, it is easy to study the computational complexity, in space and time, of the abstract operators. Most operations on intervals are linear in the number of variables. For octagons and parallelotopes operations are at most cubic on the number of variables, while polyhedra have a worst-case exponential complexity on the number of variables.

However, just knowing how each abstract operator behaves does not give a complete account of the performance of a domain in a real analysis. In particular, predicting the behavior of polyhedra is difficult because the cost of operations heavily depends on the complexity of the polyhedra found during the analysis. Therefore, there are cases when polyhedra are faster than octagons, and cases in which they are much slower.

Another factor which may influence performance is the convergence speed of the analysis. From this point of view, any attempt to improve precision by reducing the effect of widening and narrowing (for example, delayed widening or widening with threshold [13]) generally increases the time required for the analysis.

Relative precision

In this paper, we will focus on comparing the precision of analyses run with the same algorithm but different domains. However, we do not compare directly the results as returned by the analysis, for two reasons. First of all, we would get many cases of incomparable results. Second, domains such as parallelotopes and polyhedra find many complex constraints involving a lot of variables which, although may be useful to track the execution of the program, are not particularly useful in the final result.

Generally, simpler constraints are more easily applicable. For example, interval constraints may be used to prove that some run-time errors, such as division by zero or out-of-bound access to array, do not occur in practice. In case it is needed, simple program transformations may replace complex expressions with new synthetic variables, so that every interesting constraint in the original program becomes an interval constraint in the transformed one. Moreover, interval constrains are the largest set of constraints which can be explicitly represented in all the domains. For these reasons, we think that evaluating the precision of the analysis only on the interval constraints is a valuable approach.

We also include for completeness a different comparison on octagonal constraints. These constraints are useful, for example, to check for out-of-bound array accesses when arrays are created with a dimension known at run-time only. However, since new synthetic variables may be created to transform all problems to interval checking problems, we think this comparison is not as relevant as the one on interval constraints.

2 Experimental Evaluation

We have carried out an experimental evaluation to compare the relative precision of different numerical abstract domains w.r.t. the interval constraints. We have considered the following domains: intervals, octagons, polyhedra, parallelotopes and a reduced product of parallelotopes and intervals. For parallelotopes, several variants are available which differ in the heuristics used for the abstract operations which do not have a best correct abstraction. In particular, we have used the variant $\mathsf{Par}_{+\mathsf{axes}}$ [6], where abstract operations prefer to generate parallelotopes whose constraint matrix contains interval constraints. On the contrary, in the reduced product of parallelotopes and intervals we have used the variant $\mathsf{Par}_{-\mathsf{axes}}$, where abstract operations generate parallelotopes which hardly use interval constraints. For polyhedra, since we are interested in the impact of widening in the actual precision, we consider two variants, one with the H79 widening and another one with BHRZ03.

Benchmarks were performed using the Jandom static analyzer [3] on the ALICe benchmarks [19]. Jandom is an analyzer for simple imperative programs, linear transitions systems and Java bytecode. Intervals, parallelotopes and their product are natively implemented in Jandom. For octagons and polyhedra we use the implementation in the PPL.

The test-suite comprises a total of 108 models (linear transition systems) with a total of 326 locations, 161 of which are loop heads. Each model has at most 11 different locations, 4 loop heads and 10 variables. Most of the models (102 out of 108) are part of the ALICe benchmarks, the remaining 6 are taken from our previous work.

For each model a classical two-phase analysis is performed, consisting of an ascending chain with widening and a descending chain with narrowing. Widening and narrowing are applied on all loop heads. For polyhedra, the trivial narrowing which always returns the previous value of the descending chain is used. A delay is applied for both widening and narrowing. We have experimented with different values of the delay: for widening, we have used values between 0 and 6, while for narrowing values between 0 and 3. Further experiments carried out with bigger narrowing delays are not shown here, since there are practically no improvements.

All results are reproducible by running the `NSAD17Comparison` program in the `nsad17` branch of Jandom, which is available on GitHub [2].

2.1 Effect of delayed widening and narrowing on interval constraints

Table 1 shows, for each domain and for each setting of delays, the number of *non-trivial interval constraints* found by the analysis. Given an abstract object S and a program variable x, we determine the maximum value of the linear form x in S. If it is a real number or $-\infty$ (which is possible when S is empty), we have found a non-trivial interval constraint. The same is repeated with the linear form $-x$. The value which appears in the table is obtained by summing the number of non-trivial

[2] `https://github.com/jandom-devel/Jandom`

Narrowing delay \ Widening delay	Domains	0	1	2	3	4	5	6
0	Intervals	889	890	907	919	919	920	923
	Octagons	878	878	880	899	922	920	920
	Parallelotopes	847	848	876	883	884	884	884
	$\mathsf{Par}_{-\mathsf{axes}} \sqcap \mathsf{Int}$	905	921	935	946	962	961	980
	Polyhedra H79	783	771	729	752	778	794	800
	Polyhedra BHRZ03	779	785	791	807	826	838	846
1	Intervals	889	890	907	919	919	920	923
	Octagons	878	878	880	899	922	920	920
	Parallelotopes	850	851	881	886	886	887	887
	$\mathsf{Par}_{-\mathsf{axes}} \sqcap \mathsf{Int}$	912	926	940	953	963	966	983
	Polyhedra H79	921	909	863	879	885	889	893
	Polyhedra BHRZ03	901	907	909	912	921	923	927
2	Intervals	889	890	907	919	919	920	923
	Octagons	878	878	880	899	922	920	920
	Parallelotopes	850	851	881	886	886	887	887
	$\mathsf{Par}_{-\mathsf{axes}} \sqcap \mathsf{Int}$	914	928	939	955	965	968	985
	Polyhedra H79	930	918	870	886	888	892	896
	Polyhedra BHRZ03	909	912	914	920	925	927	931
3	Intervals	889	890	907	919	919	920	923
	Octagons	878	878	880	899	922	920	920
	Parallelotopes	850	851	881	889	889	890	890
	$\mathsf{Par}_{-\mathsf{axes}} \sqcap \mathsf{Int}$	910	925	942	952	962	965	982
	Polyhedra H79	930	918	870	886	888	892	896
	Polyhedra BHRZ03	909	912	914	920	925	927	931

Table 1
Number of non-trivial interval constraints found by the analysis

interval constraints found for each location of each model.

The case with narrowing delay 0 is very unfavorable for polyhedra, since it means that no descending chain is performed at all. It is shown only for completeness, but it is not particularly interesting in practice. Actually, if we exclude the polyhedra domain, delayed narrowing seems to have a very marginal benefit. Results for intervals and octagons, in particular, do not show any improvements with delayed narrowing. The fact that descending chains are generally quite short was already observed in [1,2].

The situation is very different for delayed widening. In this case all of the domains, with the exception of polyhedra with H79 widening, gradually improve precision when delay increases. There are some minor exceptions to this rule, like octagons which lose precision when delay increases from 4 to 5, and $\mathsf{Par}_{-\mathsf{axes}} \sqcap \mathsf{Int}$ which also loses precision when narrowing delay is 0 and widening delay increases from 4 to 5. Finally, polyhedra with H79 widening hardly combines with delayed widening: precision is lost moving from delay 0 to delay 2. From delay 3 onward the analysis recovers some lost precision, but it never comes back to the precision it had with delay 0.

2.2 Comparing domains w.r.t. interval constraints

By comparing the result of the different domains, we see that, starting from widening delay 2, the reduced product of parallelotopes and intervals is the domain able to find the greatest number of non-trivial interval constraints. Polyhedra with standard widening is the best when no delayed widening is in use, while it is the worst with a high delay. Its results are in any case smaller than the best results with $\mathsf{Par}_{-\mathsf{axes}} \sqcap \mathsf{Int}$

Obviously, for a given variable, location and model, two domains may find non-trivial interval constraints with different bounds. In Table 2 we have shown some results which try to take into account this fact. In this table, for each widening and narrowing delay and for each domain, we show the number of non-trivial interval constraints found by the domain whose bounds are no worse than the bounds inferred by the other domains. We only show results for delayed narrowing 1 and 2 which are the most significant cases. Although numbers are slightly different, they are in line with the results shown in Table 1.

Narrowing delay \ Widening delay	Domains	0	1	2	3	4	5	6
1	Intervals	856	857	873	877	877	878	881
	Octagons	854	854	856	871	893	891	890
	Parallelotopes	825	826	856	855	855	856	856
	$\mathsf{Par}_{-\mathsf{axes}} \sqcap \mathsf{Int}$	893	908	922	930	940	942	955
	Polyhedra H79	917	905	858	874	877	881	885
	Polyhedra BHRZ03	901	907	909	912	921	922	925
2	Intervals	856	857	873	877	877	878	881
	Octagons	853	853	855	870	892	890	890
	Parallelotopes	825	826	856	854	854	855	855
	$\mathsf{Par}_{-\mathsf{axes}} \sqcap \mathsf{Int}$	891	904	917	930	940	942	953
	Polyhedra H79	924	909	863	879	878	882	886
	Polyhedra BHRZ03	909	912	914	920	925	927	931

Table 2
Number of non-trivial interval constraints found by the analysis which have the best bounds w.r.t. other domains

The results for narrowing delay 1 or 2 are very similar. Figure 1 graphically shows the values contained in the first row only of Table 2, where narrowing delay is set to 1. It is immediate to see that the number of better interval constraints bounds found by Polyhedra BHRZ03 strictly rises when increasing the widening delay. The same does not happen for the Polyhedra H79 which outperforms the other domains only for small widening delays.

In Table 4 we show the same data from a different perspective. For each selection of delay for widening and narrowing, and for each domain, the table contains a pair $+m/-n$. Here, $+m$ ($-n$) means that the given domain has found, for m (n) interval constraints, a bound which is strictly better (worse) than the one found by Polyhedra BHRZ03.

The experiment shows that all the domains (with the only exception of Polyhedra H79 with widening delay at least 2) are able to find at least one interval constraint

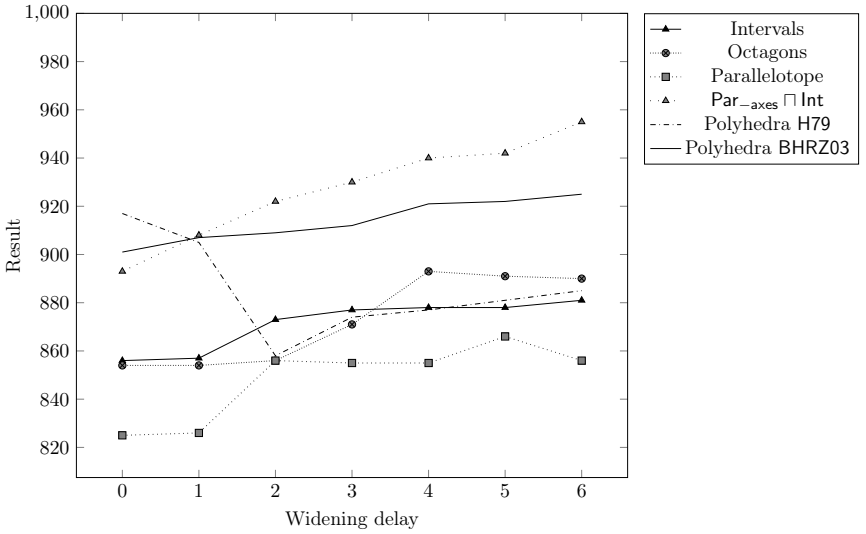

Fig. 1. Graphical representation for narrowing delay 1 in Table 2

with better bounds than Polyhedra BHRZ03, in other words we have that $m > 0$ almost in all cases. On the other side, comparing the positive and negative numbers we see that, in most case, n is almost the double of m. The only exception is the domain $\mathsf{Par}_{-\mathsf{axes}} \sqcap \mathsf{Int}$ where we have that $m > n$ when the widening delay is not 0. In particular, increasing the widening delay, the number of interval constraints with better bounds increases, till the point that m is almost the double of n.

This comparison suggests that parallelotopes, while it might not be a good domain when used alone, is able to improve the precision of other abstract domains when used in a reduced product.

2.3 Octagonal constraints

Table 3 is the analogous of Table 2 for octagonal constraints. We have omitted the analogous of Table 1 since we do not think it is relevant: all the domains find many non-trivial sub-optimal octagonal constraints, just as combination of two interval constraints.

We see that polyhedra give more precise bounds than Oct itself (with the exception of narrowing delay 0 which, as stated before, is not fair for polyhedra). Intervals and parallelotopes give the worst results, while octagons and $\mathsf{Par}_{-\mathsf{axes}} \sqcap \mathsf{Int}$ are comparable. However, we think that an hypothetical reduced product of parallelotopes and octagons would take the top spot.

Narrowing delay \ Widening delay	Domains	0	1	2	3	4	5	6
0	Intervals	1897	1900	1944	1971	1960	1963	1981
	Octagons	2533	2513	2522	2602	2678	2676	2670
	Parallelotopes	1988	1991	2077	2089	2081	2081	2080
	$\mathsf{Par}_{-\mathsf{axes}} \sqcap \mathsf{Int}$	2448	2494	2499	2500	2553	2523	2556
	Polyhedra H79	2507	2471	2303	2420	2476	2530	2598
	Polyhedra BHRZ03	2453	2482	2505	2581	2648	2714	2781
1	Intervals	1888	1892	1936	1954	1952	1955	1973
	Octagons	2520	2502	2509	2584	2665	2661	2658
	Parallelotopes	1992	1996	2090	2091	2089	2092	2091
	$\mathsf{Par}_{-\mathsf{axes}} \sqcap \mathsf{Int}$	2455	2497	2546	2555	2621	2595	2622
	Polyhedra H79	2948	2905	2708	2778	2738	2755	2821
	Polyhedra BHRZ03	2861	2880	2888	2915	2919	2957	3008
2	Intervals	1888	1892	1936	1953	1952	1955	1973
	Octagons	2516	2498	2505	2580	2661	2657	2656
	Parallelotopes	1988	1992	2090	2087	2086	2089	2088
	$\mathsf{Par}_{-\mathsf{axes}} \sqcap \mathsf{Int}$	2445	2484	2527	2548	2622	2591	2600
	Polyhedra H79	2969	2923	2720	2788	2741	2758	2824
	Polyhedra BHRZ03	2882	2896	2904	2934	2927	2968	3021
3	Intervals	1888	1892	1936	1953	1952	1955	1973
	Octagons	2516	2498	2505	2577	2658	2654	2653
	Parallelotopes	1992	1996	2090	2093	2092	2095	2094
	$\mathsf{Par}_{-\mathsf{axes}} \sqcap \mathsf{Int}$	2431	2477	2533	2515	2610	2583	2614
	Polyhedra H79	2969	2923	2720	2788	2741	2758	2824
	Polyhedra BHRZ03	2882	2896	2904	2934	2927	2968	3021

Table 3
Number of non-trivial octagonal constraints found by the analysis which have the best bounds w.r.t. other domains.

Narrowing delay \ Widening delay	Domains	0	1	2	3	4	5	6
1	Intervals	+56/-101	+54/-104	+65/-101	+64/-99	+55/-99	+55/-99	+54/-99
	Octagons	+56/-103	+53/-106	+53/-106	+59/-100	+65/-93	+62/-93	+58/-93
	Parallelotopes	+53/-129	+51/-132	+62/-115	+63/-120	+56/-122	+56/-122	+52/-122
	$\mathsf{Par}_{-\mathsf{axes}} \sqcap \mathsf{Int}$	+58/-56	+58/-47	+65/-43	+72/-44	+74/-45	+69/-39	+72/-32
	Polyhedra **H79**	+33/-17	+19/-21	+0/-51	+0/-38	+0/-44	+0/-42	+0/-42
	Polyhedra **BHRZ03**	+0/-0	+0/-0	+0/-0	+0/-0	+0/-0	+0/-0	+0/-0
2	Intervals	+51/-104	+49/-104	+60/-101	+59/-102	+54/-102	+53/-102	+52/-102
	Octagons	+51/-107	+48/-107	+48/-107	+51/-101	+61/-94	+57/-94	+53/-94
	Parallelotopes	+49/-133	+47/-133	+58/-116	+59/-125	+55/-126	+54/-126	+50/-126
	$\mathsf{Par}_{-\mathsf{axes}} \sqcap \mathsf{Int}$	+53/-61	+54/-52	+60/-48	+66/-46	+72/-47	+66/-41	+68/-34
	Polyhedra **H79**	+32/-17	+18/-21	+0/-51	+0/-41	+0/-47	+0/-45	+0/-45
	Polyhedra **BHRZ03**	+0/-0	+0/-0	+0/-0	+0/-0	+0/-0	+0/-0	+0/-0

Table 4
Number of interval constraints for which a domain improves/degrades bounds w.r.t. polyhedra with **BHRZ03** widening.

Narrowing delay \ Widening delay	Domains	0	1	2	3	4	5	6
0	Intervals	37 ± 8	33 ± 3	39 ± 7	42 ± 4	51 ± 5	53 ± 5	57 ± 4
	Octagons	624 ± 269	477 ± 57	741 ± 340	551 ± 44	810 ± 439	657 ± 55	698 ± 71
	Parallelotopes	1426 ± 27	1594 ± 26	1685 ± 17	1966 ± 16	2228 ± 95	2420 ± 29	2781 ± 22
	$\mathsf{Par}_{-\mathsf{axes}} \sqcap \mathsf{Int}$	3915 ± 27	4251 ± 52	4776 ± 38	5194 ± 54	5077 ± 24	6133 ± 27	7035 ± 52
	Polyhedra **H79**	591 ± 106	669 ± 150	768 ± 136	753 ± 37	907 ± 98	974 ± 99	1214 ± 42
	Polyhedra **BHRZ03**	803 ± 161	1004 ± 605	737 ± 88	1011 ± 327	1097 ± 82	1356 ± 131	2357 ± 291
1	Intervals	33 ± 5	34 ± 3	40 ± 5	48 ± 9	49 ± 3	48 ± 3	63 ± 9
	Octagons	528 ± 104	553 ± 34	539 ± 70	738 ± 283	619 ± 23	730 ± 264	777 ± 95
	Parallelotopes	1435 ± 30	1605 ± 24	1716 ± 20	1982 ± 10	2268 ± 22	2592 ± 144	2884 ± 19
	$\mathsf{Par}_{-\mathsf{axes}} \sqcap \mathsf{Int}$	4133 ± 30	4468 ± 63	5029 ± 10	5407 ± 43	5488 ± 43	6522 ± 48	7311 ± 55
	Polyhedra **H79**	719 ± 85	642 ± 25	812 ± 95	880 ± 136	947 ± 106	1155 ± 190	1258 ± 65
	Polyhedra **BHRZ03**	804 ± 113	867 ± 124	978 ± 138	1007 ± 91	1115 ± 136	1384 ± 108	2362 ± 97
2	Intervals	31 ± 4	38 ± 5	77 ± 88	44 ± 6	51 ± 9	109 ± 119	55 ± 5
	Octagons	484 ± 74	537 ± 41	633 ± 233	582 ± 46	609 ± 34	786 ± 285	825 ± 289
	Parallelotopes	1428 ± 14	1604 ± 29	1717 ± 23	2028 ± 95	2325 ± 125	2550 ± 48	2889 ± 40
	$\mathsf{Par}_{-\mathsf{axes}} \sqcap \mathsf{Int}$	4400 ± 7	4741 ± 30	5300 ± 13	5719 ± 137	5777 ± 35	6662 ± 26	7613 ± 18
	Polyhedra **H79**	676 ± 49	719 ± 55	753 ± 61	889 ± 125	982 ± 94	1025 ± 109	1338 ± 152
	Polyhedra **BHRZ03**	821 ± 88	785 ± 42	854 ± 139	948 ± 101	1078 ± 88	1762 ± 1128	3008 ± 1399
3	Intervals	31 ± 7	33 ± 2	37 ± 3	42 ± 4	49 ± 6	53 ± 6	59 ± 8
	Octagons	626 ± 275	546 ± 80	503 ± 78	560 ± 67	706 ± 135	794 ± 302	720 ± 12
	Parallelotopes	1429 ± 5	1625 ± 21	1709 ± 9	1996 ± 30	2294 ± 67	2552 ± 49	2942 ± 70
	$\mathsf{Par}_{-\mathsf{axes}} \sqcap \mathsf{Int}$	4509 ± 29	4866 ± 10	5455 ± 29	5925 ± 16	6102 ± 68	7030 ± 178	7971 ± 99
	Polyhedra **H79**	771 ± 126	758 ± 90	851 ± 145	906 ± 77	1049 ± 157	1123 ± 54	1248 ± 33
	Polyhedra **BHRZ03**	886 ± 102	896 ± 126	1267 ± 908	1011 ± 156	1129 ± 132	1443 ± 176	3017 ± 1323

Table 5
Mean execution time and standard deviation of the analyses in milliseconds.

2.4 Performance

In Table 5 we show the execution time of the analyses in milliseconds. Each analysis has been performed 5 times on a Intel Core i5-2400K with 8 Gb of RAM. The mean execution times and standard deviations are reported. The reported values comprise both the time required to convert the model into an equation system and the time required to solve the resulting equation system. It does not include neither loading of models from disk, nor parsing.

For intervals and octagons, widening delays do not have a big impact on performance. The opposite is true for the two variants of polyhedra. This is probably due to the fact that replacing widening with polyhedral hull gives origin to more complex polyhedra which severely harm performance of subsequent abstract operators. Narrowing delay has no big impact on execution time for any domain.

From the point of view of performance, the execution time of intervals, octagons and polyhedra are as expected. Intervals are much faster than anything else. For low values of widening delays, speed of octagons and polyhedra is comparable, but for high value of delays, octagons are faster. Parallelotopes and their reduced product with intervals are the slowest domains. Although this contrasts with the theoretical results, actually it is due to the fact that while octagons and polyhedra are part of the PPL, which is written in C++ and highly optimized, parallelotopes are written in Scala with a functional style which is not particularly well suited for this kind of application.

Actually, also intervals are written natively in Scala, but since the algorithms in this case are very simple, this is probably an advantage, since using the implementation in the PPL would incur in the overhead of calling native code from the Java Virtual Machine.

Some results have a very high standard deviation. This is probably due to some artifact of the Java Virtual Machine, such as garbage collection.

3 Conclusion

We have compared the relative precision of polyhedra, intervals, octagons, parallelotopes and a reduced product of parallelotopes and intervals w.r.t. the interval constraints on the ALICe benchmarks using the Jandom static analyzer. We have shown that, although the polyhedra domain is theoretically the most precise for inferring linear relationships, in practice the less expressive domains can find more precise results, in particular the reduced product of parallelotopes and intervals. We have also shown that delayed widening generally improves precision of the results up to a certain value (around 3, 4) with the exception of the polyhedra domain with standard widening, where it has a detrimental effect. Finally, we have shown that delayed narrowing has no significant effect on the precision of the analysis, with the exception of polyhedra domain which, lacking a narrowing operator, needs at least a delay of one during the descending phase to take a step.

As a future work, we plan to perform more experiments using techniques to improve the precision of the analysis, such as localized widening and narrowing [9],

widening with threshold [13], lookahead widening [16] and warrowing [10].

References

[1] Amato, G., S. Di Nardo Di Maio, M. C. Meo and F. Scozzari, *Narrowing operators on template abstract domains*, in: N. Bjøner and F. de Boer, editors, *FM 2015: Formal Methods, 20th International Symposium, Oslo, Norway, June 24-26, 2015, Proceedings*, Lecture Notes in Computer Science **9109**, Springer, Berlin Heidelberg, 2015 pp. 57–72.

[2] Amato, G., S. Di Nardo Di Maio, M. C. Meo and F. Scozzari, *Descending chains and narrowing on template abstract domains*, Acta Informatica **online** (2017), to be published in printed edition. DOI: 10.1007/s00236-016-0291-0.

[3] Amato, G., S. Di Nardo Di Maio and F. Scozzari, *Numerical static analysis with Soot*, in: *Proceedings of the ACM SIGPLAN International Workshop on State of the Art in Java Program analysis*, SOAP '13 (2013).

[4] Amato, G., M. Parton and F. Scozzari, *A tool which mines partial execution traces to improve static analysis*, in: H. Barringer and *et al.*, editors, *First International Conference, RV 2010, St. Julians, Malta, November 1-4, 2010. Proceedings*, Lecture Notes in Computer Science **6418**, Springer, Berlin Heidelberg, 2010 pp. 475–479.

[5] Amato, G., M. Parton and F. Scozzari, *Discovering invariants via simple component analysis*, Journal of Symbolic Computation **47** (2012), pp. 1533–1560.

[6] Amato, G., M. Rubino and F. Scozzari, *Inferring linear invariants with parallelotopes*, Science of Computer Programming **online** (2017), to be published in printed edition. DOI: 10.1016/j.scico.2017.05.011.

[7] Amato, G. and F. Scozzari, *Optimality in goal-dependent analysis of sharing*, Theory and Practice of Logic Programming **9** (2009), pp. 617–689.

[8] Amato, G. and F. Scozzari, *The abstract domain of parallelotopes*, in: J. Midtgaard and M. Might, editors, *Proceedings of the Fourth International Workshop on Numerical and Symbolic Abstract Domains, NSAD 2012*, Electronic Notes in Theoretical Computer Science **287**, Elsevier, 2012 pp. 17–28.

[9] Amato, G. and F. Scozzari, *Localizing widening and narrowing*, in: F. Logozzo and M. Fähndrich, editors, *Static Analysis. 20th International Symposium, SAS 2013, Seattle, WA, USA, June 20-22, 2013, Proceedings*, Lecture Notes in Computer Science **7935** (2013), pp. 25–42.

[10] Amato, G., F. Scozzari, H. Seidl, K. Apinis and V. Vojdani, *Efficiently intertwining widening and narrowing*, Science of Computer Programming **120** (2016), pp. 1–24.

[11] Bagnara, R., P. M. Hill, E. Ricci and E. Zaffanella, *Precise widening operators for convex polyhedra*, Science of Computer Programming **58** (2005), pp. 28–56.

[12] Bagnara, R., P. M. Hill and E. Zaffanella, *The Parma Polyhedra Library: Toward a complete set of numerical abstractions for the analysis and verification of hardware and software systems*, Science of Computer Programming **72** (2008), pp. 3–21.

[13] Blanchet, B., P. Cousot, R. Cousot, J. Feret, L. Mauborgne, A. Miné, D. Monniaux and X. Rival, *A static analyzer for large safety-critical software*, in: *Proceedings of the ACM SIGPLAN 2003 Conference on Programming Language Design and Implementation (PLDI'03)* (2003), pp. 196–207.

[14] Cousot, P. and R. Cousot, *Static determination of dynamic properties of programs*, in: *Proceedings of the Second International Symposium on Programming* (1976), pp. 106–130.

[15] Cousot, P. and N. Halbwachs, *Automatic discovery of linear restraints among variables of a program*, in: *POPL '78: Proceedings of the 5th ACM SIGACT-SIGPLAN symposium on Principles of programming languages* (1978), pp. 84–97.

[16] Gopan, D. and T. Reps, *Lookahead widening*, in: T. Ball and R. B. Jones, editors, *Computer Aided Verification, 18th International Conference, CAV 2006, Seattle, WA, USA, August 17-20, 2006. Proceedings*, Lecture Notes in Computer Science **4144** (2006), pp. 452–466.

[17] Halbwachs, N., "Détermination Automatique de Relations Linéaires Vérifoées par les Variables d'un Programme," Ph.D. thesis, Université scientifique et médicale de Grenoble, Grenoble, France (1993).

[18] Jeannet, B. and A. Miné, *APRON: A library of numerical abstract domains for static analysis*, in: A. Bouajjani and O. Maler, editors, *Computer Aided Verification, 21st International Conference, CAV 2009, Grenoble, France, June 26 – July 2, 2009. Proceedings*, Lecture Notes in Computer Science **5643**, Springer, Berlin Heidelberg, 2009 pp. 661–667.

[19] Maisonneuve, V., O. Hermant and F. Irigoin, *Alice: A framework to improve affine loop invariant computation*, in: *5th Workshop on INvariant Generation*, 2014.

[20] Miné, A., *The octagon abstract domain*, Higher-Order and Symbolic Computation **19** (2006), pp. 31–100.

[21] Monniaux, D. and J. Le Guen, *Stratified static analysis based on variable dependencies*, in: D. Massé and L. Mauborgne, editors, *Proceedings of the Third International Workshop on Numerical and Symbolic Abstract Domains, NSAD 2011*, Electronic Notes in Theoretical Computer Science **288** (2012), pp. 61–74.

[22] Sankaranarayanan, S., H. B. Sipma and Z. Manna, *Scalable analysis of linear systems using mathematical programming*, in: R. Cousot, editor, *Verification, Model Checking, and Abstract Interpretation, 6th International Conference, VMCAI 2005, Paris, France, January 17-19, 2005. Proceedings*, Lecture Notes in Computer Science **3385**, Springer, Berlin Heidelberg, 2005 pp. 25–41.

[23] Simon, A., A. King and J. M. Howe, *Two variables per linear inequality as an abstract domain*, in: M. Leuschel, editor, *Logic Based Program Synthesis and Transformation 12th International Workshop, LOPSTR 2002, Madrid, Spain, September 17–20, 2002. Revised Selected Papers*, Lecture Notes in Computer Science **2664**, Springer, Berlin Heidelberg, 2003 pp. 71–89.