## References


1. Miné, A., Breck, J., Reps, T.: An algorithm inspired by constraint solvers to infer inductive invariants in numeric programs. In: Proceedings of ESOP. (2016) 560–588
2. Goubault, E., Putot, S.: A zonotopic framework for functional abstractions. Formal Methods in System Design **47**(3) (2015) 302–360
3. Colón, M.A., Sankaranarayanan, S., Sipma, H.B.: Linear invariant generation using non-linear constraint solving. In: Proceedings of CAV, Springer (2003) 420–432
4. Garg, P., Löding, C., Madhusudan, P., Neider, D.: Ice: A robust framework for learning invariants. In: Proceedings of CAV, Springer (2014) 69–87
5. Thakur, A., Lal, A., Lim, J., Reps, T.: Posthat and all that: Automating abstract interpretation. Electronic Notes in Theoretical Computer Science **311** (2015) 15–32
6. de Oliveira, S., Bensalem, S., Prevosto, V.: Synthesizing invariants by solving solvable loops. In: Proceedings of ATVA, Springer (2017) 327–343
7. Ghorbal, K., Goubault, E., Putot, S.: The zonotope abstract domain taylor1+. In: International Conference on Computer Aided Verification, Springer (2009) 627–633
8. Jeannet, B., Miné, A.: Apron: A library of numerical abstract domains for static analysis. In: Proceedings of CAV, Springer (2009) 661–667
9. Cousot, P., Cousot, R.: Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In: Proceedings of POPL, ACM (1977) 238–252
10. Stolfi, J., De Figueiredo, L.H.: Self-validated numerical methods and applications. In: Monograph for 21st Brazilian Mathematics Colloquium, IMPA. (1997)
11. Le, V.T.H., Stoica, C., Alamo, T., Camacho, E.F., Dumur, D.: Zonotope-based set-membership estimation for multi-output uncertain systems. In: 2013 IEEE International Symposium on Intelligent Control (ISIC), IEEE (2013) 212–217
12. Tabatabaeipour, S.M., Stoustrup, J.: Set-membership state estimation for discrete time piecewise affine systems using zonotopes. In: 2013 European Control Conference (ECC), IEEE (2013) 3143–3148
13. Girard, A., Le Guernic, C.: Zonotope/hyperplane intersection for hybrid systems reachability analysis. In: International Workshop on Hybrid Systems: Computation and Control, Springer (2008) 215–228
14. Combastel, C., Zhang, Q., Lalami, A.: Fault diagnosis based on the enclosure of parameters estimated with an adaptive observer. IFAC Proceedings Volumes **41**(2) (2008) 7314–7319
15. Ghorbal, K., Goubault, E., Putot, S.: A logical product approach to zonotope intersection. In: Proceedings of CAV. (2010) 212–226
16. Althoff, M., Krogh, B.H.: Zonotope bundles for the efficient computation of reachable sets. In: 2011 50th IEEE conference on decision and control and European control conference, IEEE (2011) 6814–6821
17. Dreossi, T., Dang, T., Piazza, C.: Parallelotope bundles for polynomial reachability. In: Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control, ACM (2016) 297–306
18. Guibas, L.J., Nguyen, A., Zhang, L.: Zonotopes as bounding volumes. In: Proceedings of the ACM-SIAM symposium on Discrete algorithms. (2003) 803–812
19. Bailey, G.D.: Tilings of Zonotopes: Discriminantal Arrangements, Oriented Matroids and Enumeration. University of Minnesota (1997)
20. Richter-Gebert, J., Ziegler, G.M.: Zonotopal tilings and the bohne-dress theorem. Contemporary Mathematics **178** (1994) 211–211

21. Ferrez, J.A., Fukuda, K., Liebling, T.M.: Solving the fixed rank convex quadratic maximization in binary variables by a parallel zonotope construction algorithm. European Journal of Operational Research **166**(1) (2005) 35–50
22. Adjé, A., Gaubert, S., Goubault, E.: Coupling policy iteration with semi-definite relaxation to compute accurate numerical invariants in static analysis. In: European Symposium on Programming, Springer (2010) 23–42
23. Roux, P., Garoche, P.L.: Practical policy iterations. Formal Methods in System Design **46**(2) (2015) 163–196
24. DSilva, V., Haller, L., Kroening, D., Tautschnig, M.: Numeric bounds analysis with conflict-driven learning. In: International Conference on Tools and Algorithms for the Construction and Analysis of Systems, Springer (2012) 48–63

## 6 Proof of Lemma 1

*Proof.* Let us prove first that inequalities (5) are sufficient conditions for inclusion of $X$ into $Y$.

Based on Lemma 1 of [2], let us define $\gamma_{lin}(Y_+)$ as

$$\gamma_{lin}(Y_+) = \bigcap_{1\leq i\leq k} \left\{ x \in \mathbb{R}^p | \left| {u_i}^{\mathrm{T}} x \right| \leq ||Y_+ u_i||_1 \right\}$$

where each $u_i$ is normal to the faces of $\gamma(Y)$ (or equivalently of $\gamma_{lin}(Y_+)$). Let $x$ be any point such that $x \in \gamma(X)$. Let $x = x' + c_x$ such that $x' \in \gamma_{lin}(X_+)$ and $x''$ be any point such that $x'' = x' + (c_x - c_y)$. Let us assume that

$$\left| \langle u_i, c_x - c_y \rangle \right| \leq ||Y_+ u_i||_1 - ||X_+ u_i||_1, \forall i = 1, \cdots, k$$

Under this assumption and also by Lemma 2 of [2] i.e., $\sup\limits_{x' \in \gamma_{lin}(X_+)} \langle u, x' \rangle = ||X_+ u||_1$, we can say that

$$\left| \langle u_i, c_x - c_y \rangle \right| + \langle u_i, x' \rangle \leq ||Y_+ u_i||_1$$

This implies $\langle u_i, x'' \rangle \leq ||Y_+ u_i||_1$ which means $x'' \in \gamma_{lin}(Y_+)$. Thus, $x \in \gamma(Y)$ where the difference between $\gamma_{lin}(Y_+)$ and $\gamma(Y)$ is the translation $c_y$.

Let us prove now that inequalities (5) are necessary conditions for inclusion of $X$ into $Y$.

By Lemma 4 of [2], we know that if $\gamma(X) \subseteq \gamma(Y)$ then $\forall u$, $\left| \langle u_i, c_x - c_y \rangle \right| \leq ||Y_+ u||_1 - ||X_+ u||_1$. Thus, we can say that if $\gamma(X) \subseteq \gamma(Y)$ then
$\forall i = 1, \cdots, k \left| \langle u_i, c_x - c_y \rangle \right| \leq ||Y_+ u_i||_1 - ||X_+ u_i||_1$

□

## 7 Example for computing $\boldsymbol{u}$

*Example 2.* Consider a zonotope defined by $n$=4 generators in 3-dimension ($p$=3): $((2, -4, 2), (-1, 2, -4), (0, 0, 1), (0, 1, 0))$. We want to compute the normals $u_i$ such that each face (dimension p-1) of the zonotope has a vector in $u$ that is normal to it (Lemma 1). Consider one of the faces characterized by the set $((2, -4, 2), (-1, 2, -4))$. Now, we compute the singular value decomposition of the matrix formed from these vectors and we obtain $U\Sigma V^{\mathrm{T}} =$

$$\begin{pmatrix} -0.3374 & 0.2935 & -0.8944 \\ 0.6748 & -0.5871 & -0.4472 \\ -0.6564 & -0.7545 & 0.0000 \end{pmatrix} \begin{pmatrix} 6.3689 & 0 \\ 0 & 2.1066 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} -0.7359 & 0.6771 \\ 0.6771 & 0.7359 \end{pmatrix} \tag{12}$$

The vector normal to the face in question is $(-0.8944, -0.4472, 0)$. In this manner, we compute the remaining normal vectors.

## 8 Example with the proposed meet operation

*Example 3.* We set $\mathfrak{Z}_1$ to be the initial box $(x, y) \in S_0 = [-2, 2]^2$. It can be abstracted using zonotopes as $\mathfrak{Z}_1 = \begin{pmatrix} 2\varepsilon_1 & 2\varepsilon_2 \end{pmatrix}^{\mathrm{T}}$. Consider now $\mathfrak{Z}_2$ to be the effect of the body of a loop on $\mathfrak{Z}_1$:

$$F^\sharp(S_0) = \begin{pmatrix} 1.4\varepsilon_1 & 1.4\varepsilon_2 \\ 1.4\varepsilon_1 & -1.4\varepsilon_2 \end{pmatrix}.$$

The intersection $S_0 \cap F^\sharp(S_0)$ is over-approximated by

$$\alpha \begin{pmatrix} 2\varepsilon_1 \\ 2\varepsilon_2 \end{pmatrix} + (1 - \alpha) \begin{pmatrix} 1.4 \times (\varepsilon_3 - \varepsilon_4) \\ 1.4 \times (\varepsilon_3 + \varepsilon_4) \end{pmatrix}.$$

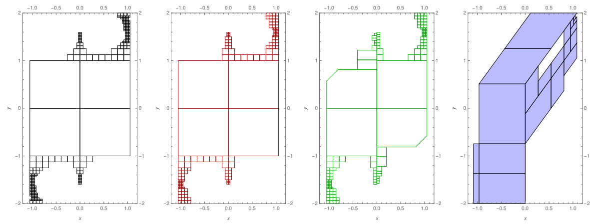
We then choose $\alpha$ so as to minimize the distance $||A_+ u_i||_1$ (this can be solved by linear programming), for $u_i$ the normals to the faces of zonotopes $\mathfrak{Z}_1$ and $\mathfrak{Z}_2$. Here, we obtain $\alpha$ equal to 0.5.

## 9 Inductive invariants found on Sine programs

Fig. 7(a), 7(b), 7(c) and 7(d) compare the result of the algorithm on the *Sine* program using intervals, octagons, polyhedras and zonotopes.

## 10 Programs

The programs are illustrated as shown in the Fig. 8, a structure compliant with respect to the analyzer. In Figure 8, init stands for the intial state of the program, which the final inductive invariant must contain. If any abstract element does not intersect with the init, then it is not a *necessary* abstract element. Body denotes the loop of the program and the goal is the candidate invariant.



(a) (b) (c) (d)

**Fig. 7.** Inductive invariant for *Sine* example (a) 240 boxes 1448 iterations, 0.4395 s; (b) 154 octagons, 348 iterations, 0.1102 s; (c) 136 polyhedras, 286 iterations, 1.1145 s; (d) 21 zonotopes, 33 iterations, 0.0547 s.

```
init
  {

  }
body
  {

  }
goal
  {

  }
```

**Fig. 8.** Structure of a program for the analyzer

Octagon

```
init {
  x=[-1,1];
  y=[-1,1];
}

body {
  t=0.7*(x+y);
  y=0.7*(x-y);
  x=t;
}

goal {
  x=[-2,2];
  y=[-2,2];
}
```

Filter

```
init
  {
  x=[-0.1,0.1];
  y=[-0.1,0.1];
  }
body
  {
  r = 1.5*x - 0.7*y + [-0.1,0.1];
  y = x;
  x = r;
  }
goal
  {
  x=[-4,4];
  y=[-4,4];
  }
```

Filter2

```
init {
  x=[0,1];
  y=[0,1];
}

body {
  x = (3.0/4.0) * x - (1.0/8.0)* y;
  y = x;
}

goal {
  // This goal was not taken from the paper itself,
  // but was chosen as a reasonable property to prove.
  x=[-0.2,1];
  y=[-0.2,1];
}
/*
Example file: from LMCS '12: ACCURATE NUMERICAL INVARIANTS
              by Adje, Gaubert, and Goubault
              Figure 11: The program Filter
*/

/*
  // Code from the paper:
  x = [0,1];
  y = [0,1];
  while ( true ) {
    x = (3/4)* x-(1/8)*y;
    y = x;
  }
*/
```

Arrow-Hurwicz

```
/*  Example file: from ESOP'10: Coupling Policy Iteration with
                 Semi-definite Relaxation to Compute
                 Accurate Numerical Invariants in Static
                 Analysis by Adje, Gaubert, and Goubault
****************************************************************
NOTE: This is a *modified* version of the Arrow-Hurwicz example
      because it does not represent u and v as separate
      variables, and it has no loop condition.
****************************************************************
*/
/*
  // Original code from
  // http://www.lix.polytechnique.fr/~adje/uploads/Codes.pdf
  a = 1;b = -1;c = -1;r = 1/2;
  u = [0, 1];v = [0, 1];x = [0, 3/2];y = [3/8, 11/8];
  while (max(|x-u|, |y-v|) > 1e-9) {
    u = x;
    v = y;
    x = u - r * (a * u + b * v);
    y = v + (r / 2) * (b * u - c);
    if (y <= 0) {
      y = 0;
    }
  }
*/
init {
  x = [0,1.5];
  y = [0.375,1.375];
}
body {
  // Note: We have removed the loop condition,
  // so u and v are temporary variables,
  //   not carried from one iteration to the next.
  //   We have also substituted in constants a,b,c,r
  u = x;
  v = y;
  x = u - 0.5 * (1 * u + (-1) * v);
  y = v + (0.5 / 2) * ((-1) * u - (-1));
  if (y <= 0) {
    y = 0;
  }
}
goal {
// Note: these bounds are not from the paper,
// they are simply a reasonable guess.
  x = [-1.73,1.73];
  y = [-1.73,1.73];
}
```

Harm

```
init {
  x0=[0,1];
  x1=[0,1];
}

body {
  x0p = x0; x1p = x1;
  x0 =  0.95*x0p + 0.09975*x1p;
  x1 = -0.1 *x0p + 0.95 *x1p;
}

goal {
  // Bounds given in the Roux paper (Table 3)
  x0=[-1.27,1.27];
  x1=[-1.27,1.27];
  // Goal from ESOP'10 paper (policy iteration goal)
  //x=[-1.42,1.42];
  //v=[-1.42,1.42];
}

/*
Example file:
   Practical policy iterations
   Roux and Garoche
   Form Methods Syst Des 2015
   DOI 10.1007/s10703-015-0230-7
   Also known as Ex.8 (Harmonic oscillator)
   ultimately from LMCS '12: ACCURATE NUMERICAL INVARIANTS
   by Adje, Gaubert, and Goubault
   Figure 12: An implementation of the Symplectic method
*/
/*
  // Code from benchmarks tarball
  node top(ix0, ix1 : real) returns (x0, x1 : real);
  let
    assert(ix0 > 0. and ix0 < 1.);
    assert(ix1 > 0. and ix1 < 1.);
    x0 = ix0 -> 0.95 * pre x0 + 0.09975 * pre x1;
    x1 = ix1 -> -0.1 * pre x0 + 0.95 * pre x1;
  tel
*/
/*
  // Original code from the Goubault paper
  tau = 0.1;
  x = [0,1];
  v = [0,1];
  while (true) {
    x = (1-( tau / 2 ) ) * x + (tau -(( tau^3 ) / 4 ) ) * v ;
    v = -tau *x+(1-( tau / 2 ) )* v ;
  }
*/
```

Harm-reset

```
init {
  x0=[0,1];
  x1=[0,1];
}
body {
  x0p = x0; x1p = x1;
  x0 =  0.95*x0p + 0.09975*x1p;
  x1 = -0.1 *x0p + 0.95 *x1p;
  if ([0,1] > 0.5) {
    x0 = 1;
    x1 = 1;
  }
}
goal {
  // Bounds given in the Roux paper (Table 3) (for reset Ex8)
  //x0=[-1.00,1.00];
  //x1=[-1.01,1.01];
  // Bounds given in the Roux paper (Table 3) (for original Ex8)
  x0=[-1.27,1.27];
  x1=[-1.27,1.27];
  // Goal from ESOP'10 paper (policy iteration goal)
  //x=[-1.42,1.42];
  //v=[-1.42,1.42];
}
/*  Example file:
     Practical policy iterations
     Roux and Garoche
     Form Methods Syst Des 2015
     DOI 10.1007/s10703-015-0230-7
     Also known as Ex.8 (Harmonic oscillator reset)
     ultimately from LMCS '12: ACCURATE NUMERICAL INVARIANTS
     by Adje, Gaubert, and Goubault
     Figure 12: An implementation of the Symplectic method */
/*
  // Code from benchmarks tarball
  node top(r : bool; ix0, ix1 : real) returns (x0, x1 : real);
  let
    assert(ix0 > 0. and ix0 < 1.);
    assert(ix1 > 0. and ix1 < 1.);
    x0 = ix0 -> if r then 1. else 0.95*pre x0 + 0.09975*pre x1;
    x1 = ix1 -> if r then 1. else -0.1*pre x0 + 0.95*pre x1;
  tel
*/
/* Original code from the Goubault paper
  tau = 0.1;
  x = [0,1];
  v = [0,1];
  while (true) {
    x = (1-( tau / 2 ) ) * x + (tau -(( tau^ 3 ) / 4 ) ) * v ;
    v = -tau *x+(1-( tau / 2 ) )* v ;
  }
*/
```

Harm-saturated

```
init {
  x0=[0,1];
  x1=[0,1];
}
body {
  x0p = x0; x1p = x1;
  x0 =  0.95*x0p + 0.09975*x1p;
  x1 = -0.1 *x0p + 0.95*x1p;

  if (x0 > 0.5) { x0 = 0.5; }
  if (x0 < -0.5) { x0 = -0.5; }
}
goal {
  // Bounds given in the Roux paper (Table 3)
  x0=[-1.27,1.27];
  x1=[-1.27,1.27];
  // Goal from ESOP'10 paper (policy iteration goal)
  //x=[-1.42,1.42];
  //v=[-1.42,1.42];
}
/*  Example file:
     Practical policy iterations
     Roux and Garoche
     Form Methods Syst Des 2015
     DOI 10.1007/s10703-015-0230-7
     Also known as Ex.8 (Harmonic oscillator saturate)
     ultimately from LMCS '12: ACCURATE NUMERICAL INVARIANTS
     by Adje, Gaubert, and Goubault
     Figure 12: An implementation of the Symplectic method
*/
/* Code from benchmarks tarball
  node top(ix0, ix1 : real) returns (sx0, x0, x1 : real);
  let
    assert(ix0 > 0. and ix0 < 1.);
    assert(ix1 > 0. and ix1 < 1.);
    x0=ix0 -> 0.95 * pre sx0 + 0.09975 * pre x1;
    x1=ix1 -> -0.1 * pre sx0 + 0.95 * pre x1;
    sx0=if x0 > 0.5 then 0.5 else if x0<-0.5 then -0.5 else x0;
  tel
*/
/*
  // Original code from the Goubault paper
  tau = 0.1;
  x = [0,1];
  v = [0,1];
  while (true) {
    x = (1-( tau / 2 ) ) * x + (tau -(( tau^3 ) / 4 ) ) * v ;
    v = -tau *x+(1-( tau / 2 ) )* v ;
  }
*/
```

Sine

```
init
  {
  x = [-1.57079632679,1.57079632679];
  y=[0,0];
  }
body
  {
  y=x - x^3/6 + x^5/120 - x^7/5040;
  }
goal
  {
  x=[-2,2];
  y=[-1.05,1.05];
  }
/*
  Example file from Leopold Haller's benchmark
  Original code from
  http://www.cprover.org/cdfpl/
  Simple Taylor expansion of sine

  =>
  can prove a bound of 1.05 for the output
  We wanted to illustrate that it works for tighter bounds

  the vertical bar for y=0 is expected
  (the inductive invariant must
  include both the init state,
  where y=0, and the end state, where
  y is the approximate sine)
*/
```

Newton

```
init {
  x=[-1,1];
  out=[0,0];
}

body {
  y = x - x*x*x/6 + x*x*x*x*x/120 + x*x*x*x*x*x*x/5040;
  z = 1 - x*x/2 + x*x*x*x/24 + x*x*x*x*x*x/720;
  x = x - y / z;
  out = x;
}

goal {
  x=[-1,1];
  out=[-0.56,0.56];
}
/*
  Example file from Leopold Haller's benchmark
  Original code from
  http://www.cprover.org/cdfpl/
  Newton iterations: one step
  =>
  can prove an output bound of 0.56
*/
```

Newton2

```
init {
  x=[-1,1];
  out=[0,0];
}

body {
  y = x - x*x*x/6 + x*x*x*x*x/120 + x*x*x*x*x*x*x/5040;
  z = 1 - x*x/2 + x*x*x*x/24 + x*x*x*x*x*x/720;
  x = x - y / z;

  y = x - x*x*x/6 + x*x*x*x*x/120 + x*x*x*x*x*x*x/5040;
  z = 1 - x*x/2 + x*x*x*x/24 + x*x*x*x*x*x/720;
  x = x - y / z;

  out = x;
}

goal {
  x=[-1,1];
  out=[-0.1,0.1];
}
/*
Example file from Leopold Haller's benchmark
  Original code from
  http://www.cprover.org/cdfpl/
  Newton iterations: two steps
  =>
  works for bound of 0.1
*/
```

Square root

```
init {
  x=[0,1];
  out=[0,0];
}

body {
out = 1 + 0.5*x - 0.125*x*x + 0.0625*x*x*x - 0.0390625*x*x*x*x;
}

goal {
  x=[0,1];
  out=[0,1.5];
}
/*
  Example file from Leopold Haller's benchmark
  Original code from
  http://www.cprover.org/cdfpl/
  Simple polynomial interpolation function for square root.
  =>
  we can prove that the output is <= 1.5
  we can prove tighter bound (such as 1.4 or 1.39844)
  using zonotopes but not using boxes and octagons
*/
```

Corner

```
init {
  x=[0.9,1.1];
  y=[0.9,1.1];
}

body {
  d = (0.2 + x*x + y*y + 1.53*x*x*y*y)/2;
  x = x / d;
  y = y / d;
}

goal {
  x=[-2.1,2.1];
  y=[-2.1,2.1];
}
```