Cylindrical Algebraic Decomposition in the RegularChains Library

Changbo Chen[1] and Marc Moreno Maza[2]

[1] Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences
[2] ORCCA, University of Western Ontario

August 9, 2014
ICMS 2014, Seoul, Korea

Outline

1. Introduction
2. Functionality
3. Underlying theory and technical contribution
4. Experimentation
5. Application
6. Conclusion

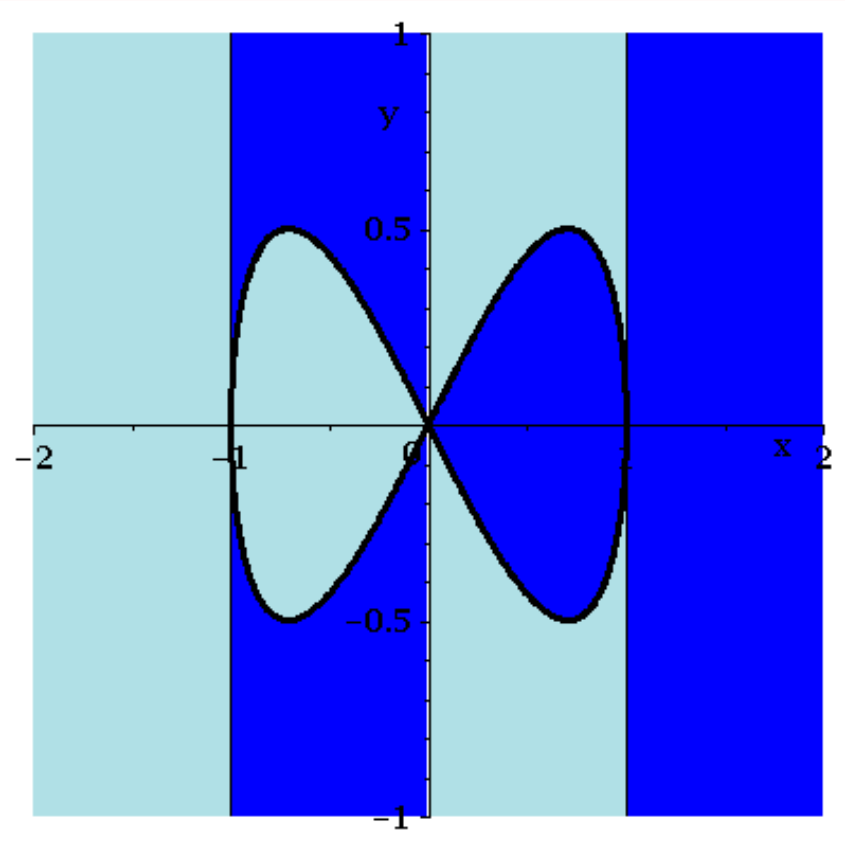
Outline

1. Introduction
2. Functionality
3. Underlying theory and technical contribution
4. Experimentation
5. Application
6. Conclusion

Cylindrical Algebraic Decomposition (CAD) of $\mathbb{R}^n$

A CAD of $\mathbb{R}^n$ is a partition of $\mathbb{R}^n$ such that each cell in the partition is a connected semi-algebraic subset of $\mathbb{R}^n$ and all the cells are cylindrically arranged.

Two subsets A and B of $\mathbb{R}^n$ are called cylindrically arranged if for any $1 \leq k < n$, the projections of A and B on $\mathbb{R}^k$ are either equal or disjoint.



Cylindrical algebraic decomposition based on projection and lifting

- Invented by G.E. Collins in 1973 for solving real Quantifier Elimination (QE) problems.
- The method is based on a projection-lifting scheme (PL-CAD).

Projection: Repeatedly apply a projection operator $Proj$:

$$F_n(x_1, \ldots, x_n) \xrightarrow{Proj} F_{n-1}(x_1, \ldots, x_{n-1}) \xrightarrow{Proj} \cdots \xrightarrow{Proj} F_1(x_1).$$

Lifting:

- The real roots of the polynomials in F_1 plus the open intervals between them form an F_1-invariant CAD of $\mathbb{R}^1$.
- For each cell C of the F_{k-1} invariant CAD of $\mathbb{R}^{k-1}$, isolating the real roots of the polynomials of F_k at a sample point of C, produces all the cells of the F_k-invariant CAD of $\mathbb{R}^k$ above C.

Best known software based on PL-CAD: QEPCAD (H. Hong, C. Brown), Mathematica (A. Strzeboński), Redlog (A. Dolzmann, T. Sturm), SynRAC (H. Iwane, H. Yanami, H. Anai, and K. Yokoyama).

Cylindrical algebraic decomposition based on regular chains (RC-CAD)

Motivation: potential drawback of Collins' projection-lifting scheme

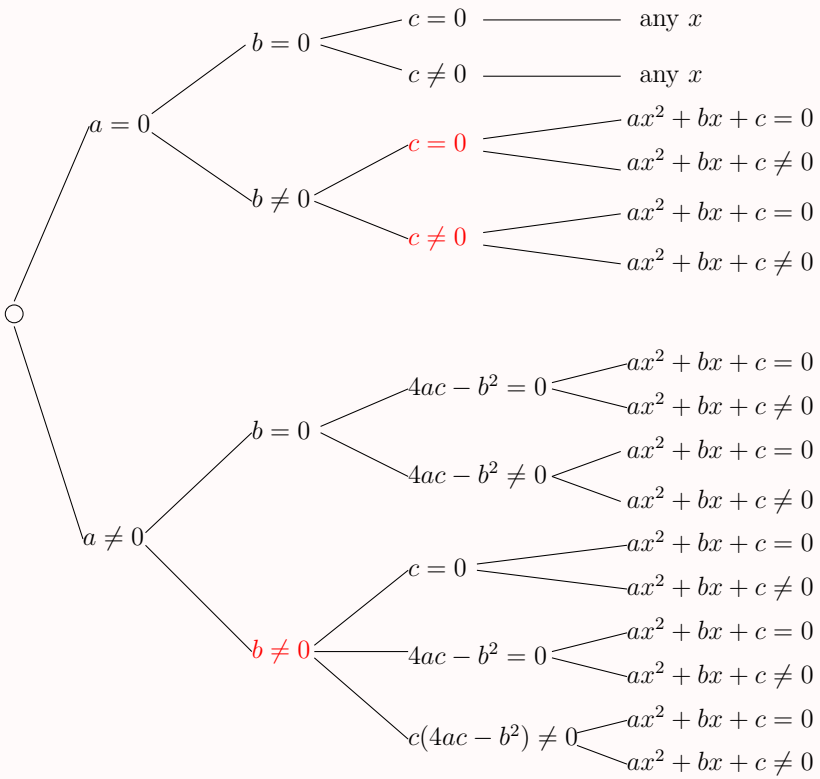
- The projection operator is a function defined independently of the input system.
- As a result, a strong projection operator (Collins-Hong operator) usually produces much more polynomials than needed.
- A weak projection operator (McCallum-Brown operator) may fail for non-generic cases.

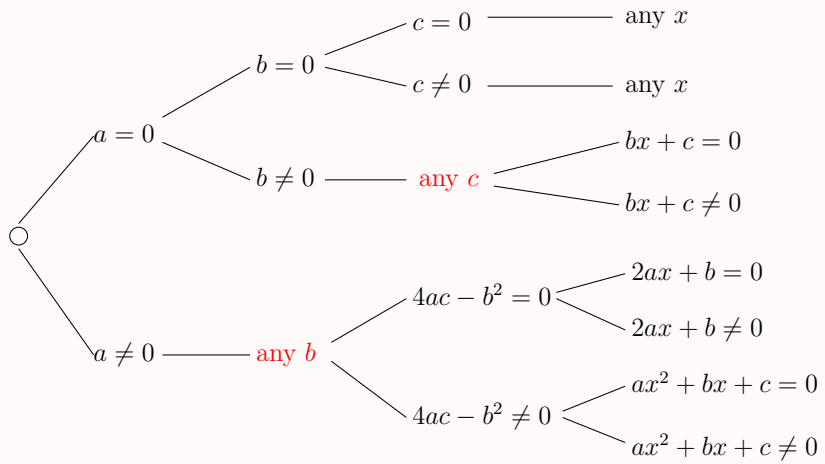
Solution: make case discussion during projection

- At ISSAC'09, we (with B. Xia and L. Yang) introduced case discussion into CAD computation based on the theory of regular chains and triangular decompositions.
- The new method consists of two phases. The first phase computes a complex cylindrical tree (CCT). The second phase decomposes each cell of CCT into its real connected components.

Illustrate PL-CAD and RC-CAD by parametric parabola example

Let $f := ax^2 + bx + c$. Suppose we like to compute a f-sign invariant CAD. The projection factors are $a, b, c, 4ac - b^2, ax^2 + bx + c$. Rethinking PL-CAD in terms of a *complex cylindrical tree*, we get the left tree.





Clearly, RC-CAD (see right tree) computes a smaller tree by *avoiding useless case distinction.*

Outline

1. Introduction
2. Functionality
3. Underlying theory and technical contribution
4. Experimentation
5. Application
6. Conclusion

Computer complex cylindrical decomposition by CylindricalDecompose

```
> R := PolynomialRing([y, x]):
  F := [y^2-x]:
  CylindricalDecompose(F, R, output=piecewise);
```

$$\begin{cases} \begin{cases} 1 & y = 0 \\ 1 & otherwise \end{cases} & x = 0 \\ \begin{cases} 1 & -y^2 + x = 0 \\ 1 & otherwise \end{cases} & otherwise \end{cases}$$

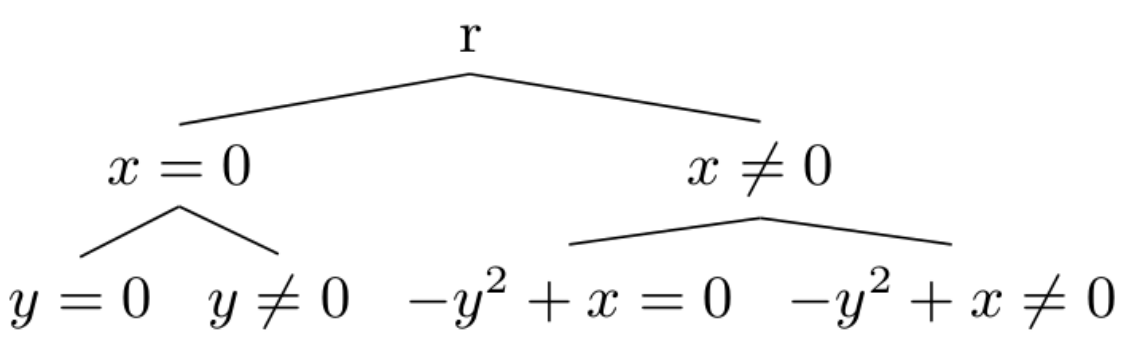


Figure: A complex cylindrical tree.

Compute CAD by CylindricalAlgebraicDecompose.

```
> R := PolynomialRing([y, x]):
  F := [y^2-x]:
  CylindricalAlgebraicDecompose(F,R,output=piecewise);
```

$$
\begin{cases}
1 & x < 0 \\
\begin{cases} 1 & y < 0 \\ 1 & y = 0 \\ 1 & 0 < y \end{cases} & x = 0 \\
\begin{cases} 1 & y < -\sqrt{x} \\ 1 & y = -\sqrt{x} \\ 1 & \mathrm{And}\left(-\sqrt{x} < y, y < \sqrt{x}\right) \\ 1 & y = \sqrt{x} \\ 1 & \sqrt{x} < y \end{cases} & 0 < x
\end{cases}
$$

Different input and output formats of CylindricalAlgebaicDecompose

```
> R := PolynomialRing([y, x]):
  cad := CylindricalAlgebraicDecompose([x^2+y^2-1=0, x*y-1/2=0],R,output=cadcell);
  Display(cad, R);
  sp := map(SamplePoints, cad, R);
  Display(sp, R);
  s := SignAtBox(x^2+y^2-2, sp[1], R);
```

$$cad := [cad\_cell,\ cad\_cell]$$

$$\left[\left\{\begin{array}{l} y = \frac{1}{2\,x} \\ x = -\frac{1}{2}\sqrt{2} \end{array}\right\},\ \left\{\begin{array}{l} y = \frac{1}{2\,x} \\ x = \frac{1}{2}\sqrt{2} \end{array}\right\}\right]$$

$$sp := [box,\ box]$$

$$\left[\left\{\begin{array}{l} y = \left[-\frac{46341}{65536}, -\frac{1482907}{2097152}\right] \\ x = \left[-\frac{46341}{65536}, -\frac{741455}{1048576}\right] \end{array}\right\},\ \left\{\begin{array}{l} y = \left[\frac{185363}{262144}, \frac{741457}{1048576}\right] \\ x = \left[\frac{741455}{1048576}, \frac{46341}{65536}\right] \end{array}\right\}\right]$$

$$s := -1$$

```
> R := PolynomialRing([y, x]):
  CylindricalAlgebraicDecompose([x^2+y^2-1<=0],R,output=
  rootof);
```

$$\left[\left[\mathrm{And}(-1 \le x,\ x \le 1),\ \mathrm{And}\left(-\sqrt{-x^2+1} \le y,\ y \le \sqrt{-x^2+1}\right)\right]\right]$$

Group CAD cells together

Different optimizations supporting CylindricalAlgebaicDecompose

- 'optimization=EC' : utilize equational constraints in single conjunction
- 'optimization=TTICAD', utilize equational constraints in multiple conjunction

```
> p1 := randpoly([x, y, z], terms=4, degree=2, coeffs=rand(-2..2));
  p2 := randpoly([x, y, z], terms=4, degree=2, coeffs=rand(-2..2));
  g1 := randpoly([x, y, z], terms=4, degree=2, coeffs=rand(-2..2));
  g2 := randpoly([x, y, z], terms=4, degree=2, coeffs=rand(-2..2));
```

$$p1 := 2x^2 - 2y^2 - 2z^2 - x$$

$$p2 := x^2 + 2y^2 + z^2 + x$$

$$g1 := z^2 + z$$

$$g2 := 2xy + xz - 2yz$$

```
> R := PolynomialRing([z, y, x]):
> t0 := time():CylindricalAlgebraicDecompose([p1=0, p2=0, g1>0, g2>0], R,
  optimization=false):t1 := time() - t0;
```

$$t1 := 24.421$$

```
> t0 := time():CylindricalAlgebraicDecompose([p1=0, p2=0, g1>0, g2>0], R,
  optimization=EC):t2 := time() - t0;
```

$$t2 := 0.154$$

```
> t0 := time():CylindricalAlgebraicDecompose([[p1=0, g1>0], [p2=0, g2>0]], R,
  optimization=false):t3 := time() - t0;
```

$$t3 := 27.353$$

```
> t0 := time():CylindricalAlgebraicDecompose([[p1=0, g1>0], [p2=0, g2>0]], R,
  optimization=TTICAD):t3 := time() - t0;
```

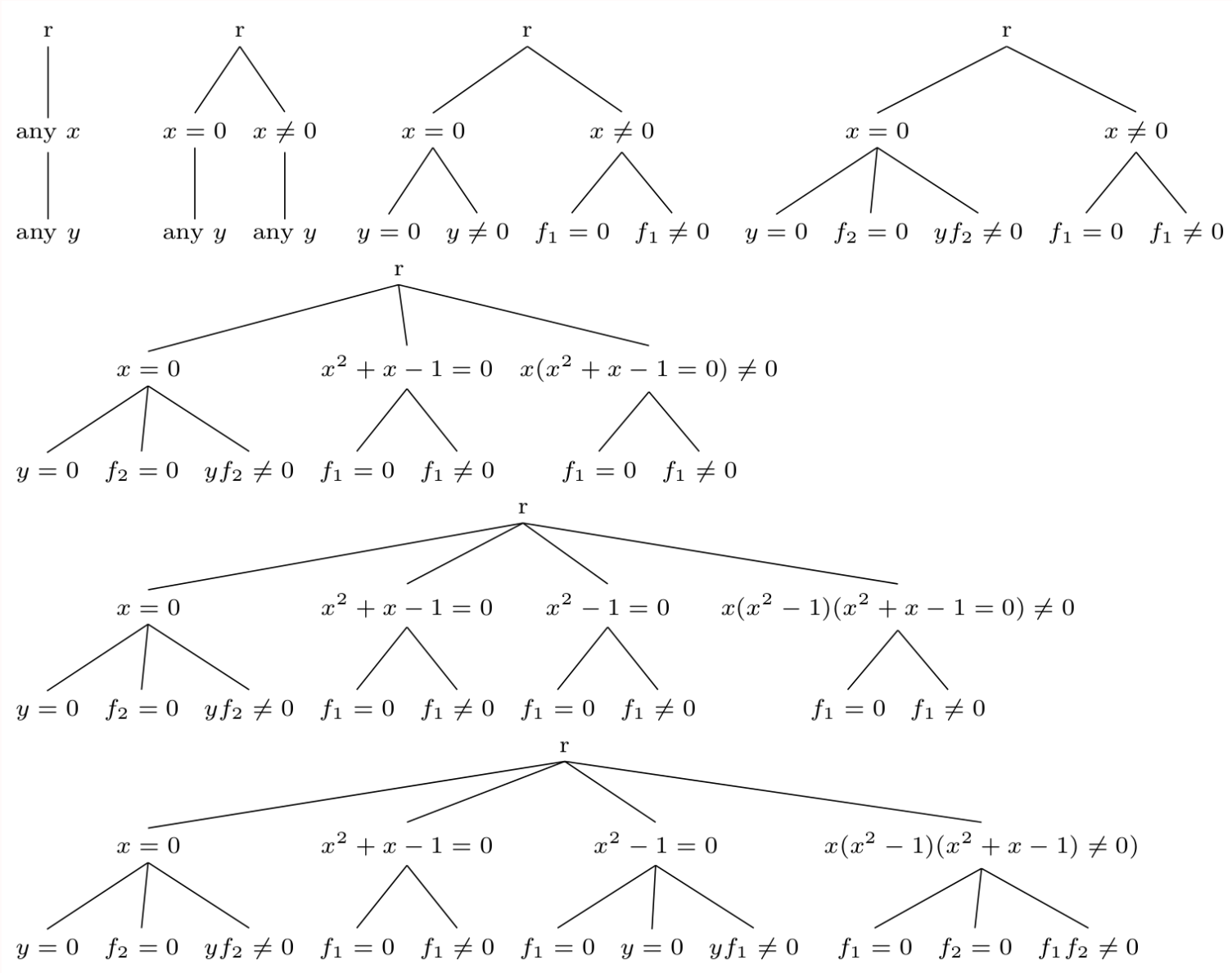
$$t3 := 4.941$$

Outline

1. Introduction
2. Functionality
3. Underlying theory and technical contribution
4. Experimentation
5. Application
6. Conclusion

The process of computing a CCT (C. Chen, M. Moreno Maza, ASCM 2012)

Let $F := \{y^2 - x, x^2 + y^2 - 1\}$, where $f_1 = y^2 - x$, $f_2 = x^2 + y^2 - 1$.



The underlying data structure for computing CCT and CAD

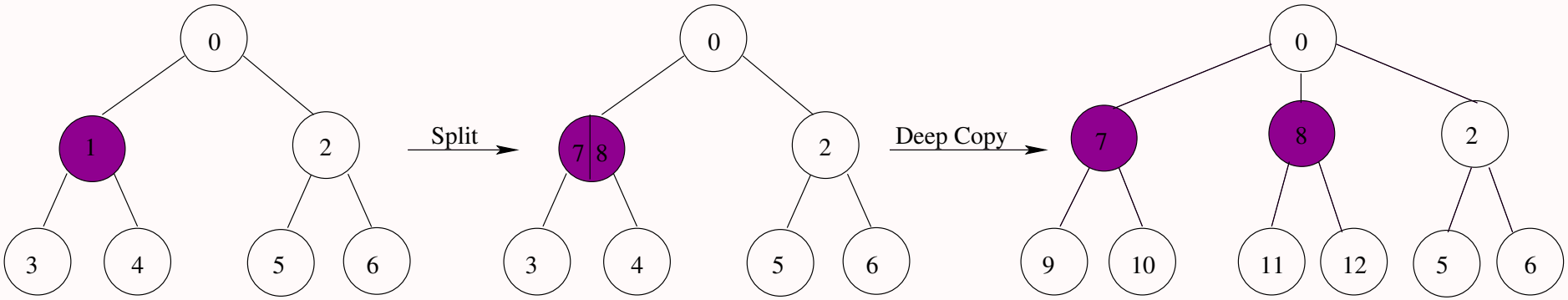


Figure: The universe tree and the Split operation.

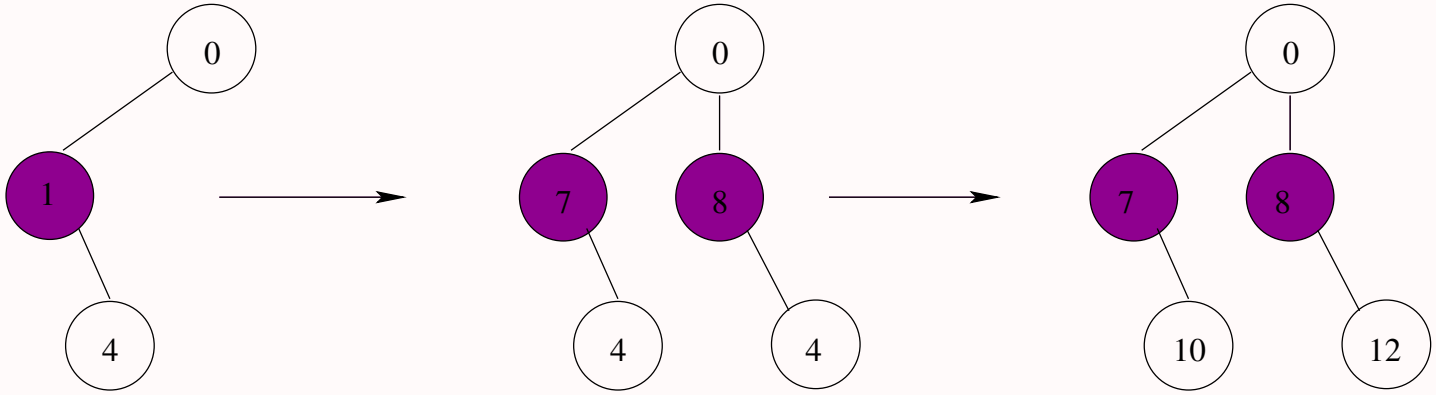


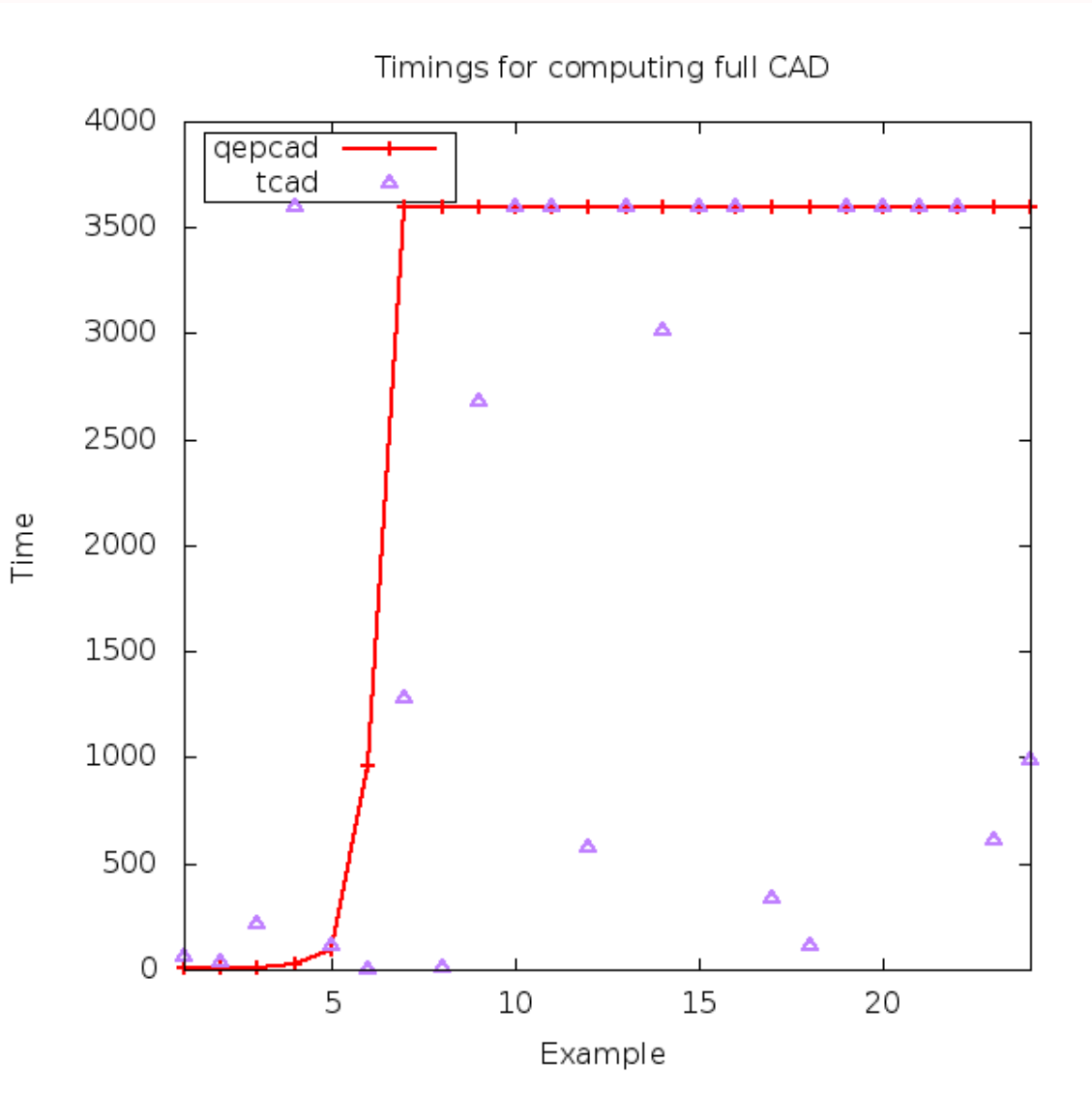
Figure: A sub-tree evolves with the universe tree.

Outline

1. Introduction
2. Functionality
3. Underlying theory and technical contribution
4. Experimentation
5. Application
6. Conclusion

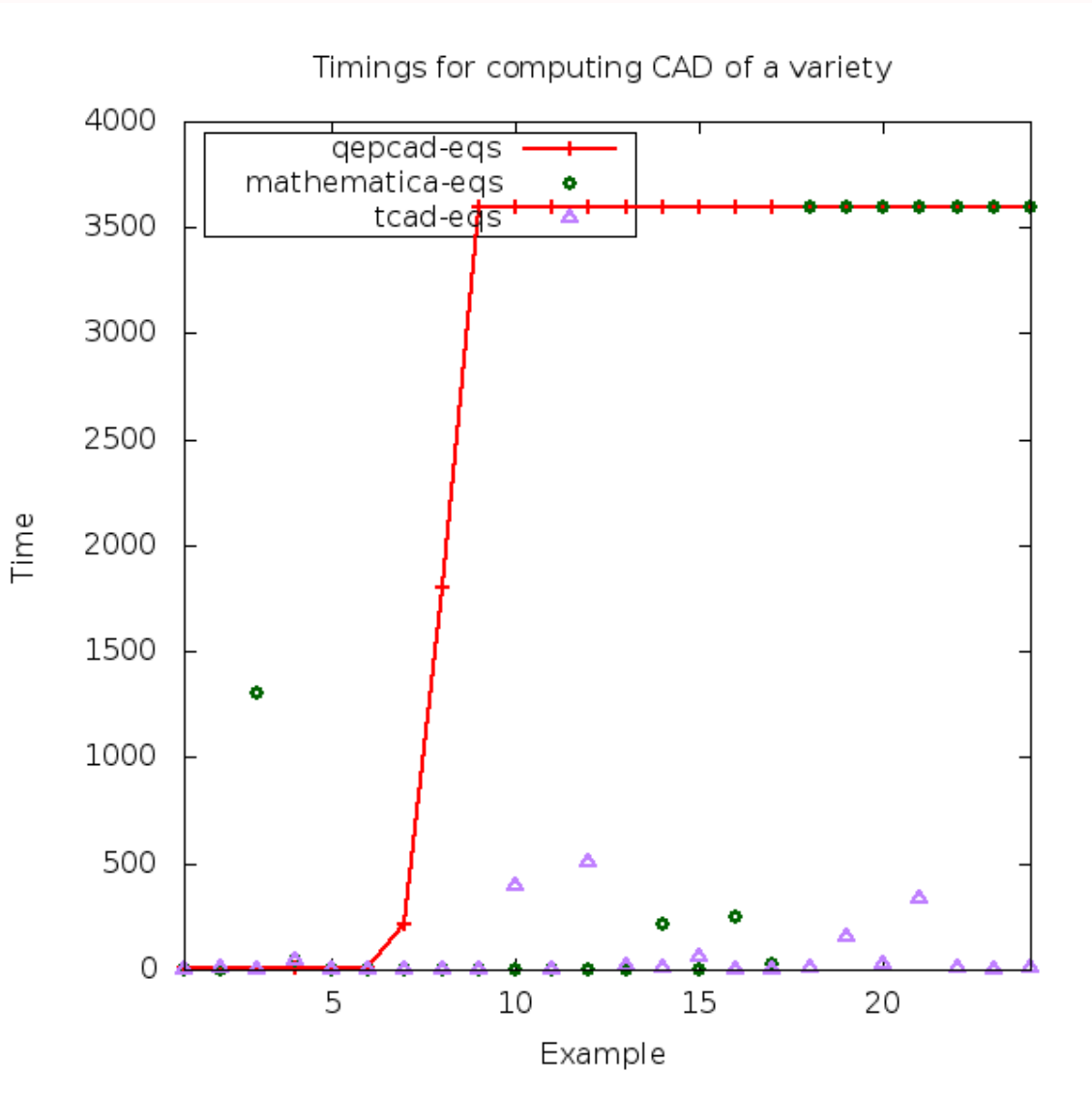
Benchmark between QEPCAD and RC-CAD (TCAD) for computing full CADs

This is taken from ASCM 12 paper by C. Chen and M. Moreno Maza.



Benchmark between QEPCAD,Mathematica and RC-CAD (TCAD) for computing CADs of a variety

This is taken from ASCM 12 paper by C. Chen and M. Moreno Maza.



Benchmark between RC-TTICAD and other CAD solvers for computing CADs having equations (1/2)

This is taken from CASC 14 paper by R. Bradford, C. Chen, J. H. Davenport, M. England, M. Moreno Maza and D. J. Wilson.

		RC-TTICAD		Mathematica	Qepcad		SyNRAC		Redlog	
Problem	n	Cells	Time	Time	Cells	Time	Cells	Time	Cells	Time
Intersection†	3	541	1.0	0.1	3723	4.9	3723	12.8	Err	—
Ellipse†	5	71231	317.1	11.2	500609	275.3	Err	—	Err	—
Solotareff†	4	2849	8.8	0.1	16603	5.2	Err	—	3353	8.6
Solotareff††	4	8329	21.4	0.1	16603	5.3	Err	—	8367	13.6
2D Ex†	2	97	0.2	0.0	249	4.8	317	1.1	305	0.9
2D Ex††	2	183	0.4	0.0	317	4.6	317	1.2	293	0.9
3D Ex†	3	109	3.5	0.1	739	5.4	—	T/O	Err	—
MontesS10	7	3643	19.1	T/O	—	T/O	—	T/O	Err	—
Wang 93	5	507	44.4	—	897.1	FAIL	—	Err	—	
Rose†	3	3069	200.9	T/O	FAIL	—	—	T/O	Err	—
genLinSyst-3-2†	11	222821	3087.5	T/O	FAIL	—	Err	—	Err	—
BC-Kahan	2	55	0.2	0.1	261	4.8	409	1.5	Err	—
BC-Arcsin	2	57	0.1	0.0	225	4.8	225	0.7	161	2.4
BC-Sqrt	4	97	0.2	0.0	105	4.7	105	0.4	73	0.0
BC-Arctan	4	211	3.5	T/O	—	T/O	Err	—	—	T/O
BC-Arctanh	4	211	3.5	T/O	—	T/O	Err	—	—	T/O
BC-Phisanbut-1	4	325	0.8	0.1	377	4.8	389	2.0	217	0.2
BC-Phisanbut-4	4	543	1.6	11.9	51763	8.6	Err	—	Err	—

Benchmark between RC-TTICAD and other CAD solvers for computing CADs having equations (2/2)

- Each problem had a declared variable ordering.
- RC-TTICAD never gives higher cell counts than any of our previous work (PL-TTICAD, PL-CAD CAD with McCallum projection, RC-Inc-CAD (ASCM 2012) RC-Rec-CAD (ISSAC 2019)).
- RC-TTICAD is usually the quickest in some cases offering vast speed-ups. Moreover, there are many examples where PL-TTICAD has a theoretical failure but for which RC-TTICAD will complete.
- TTICAD theory allows for lower cell counts than QEPCAD even when manually declaring an EC.
- We found that both SYNRAC and REDLOG failed for many examples, but we did not have access to the latest SYNRAC.
- MATHEMATICA is the quickest in general, but the output is a formula with a cylindrical structure, instead of a CAD; this cannot cover all applications like algebraic simplification by branch cut decomposition.
- Further, there are examples for which RC-TTICAD completes but MATHEMATICA times out.

Outline

1. Introduction
2. Functionality
3. Underlying theory and technical contribution
4. Experimentation
5. Application
6. Conclusion

The following two challenges were posted in "Program verification in the presence of complex numbers, functions with branch cuts etc." by J. H. Davenport et al, 2012.

Challenge

Demonstrate automatically the truth of Formula 1 over reals.

$$\forall x_1 \forall x_2 \forall y_1 \forall y_2 \quad (x_1^2 + y_1^2 > 1 \wedge x_2^2 + y_2^2 > 1 \wedge x_1 + \frac{x_1}{x_1^2+y_1^2} = x_2 + \frac{x_2}{x_2^2+y_2^2} \wedge y_1 - \frac{y_1}{x_1^2+y_1^2} = y_2 - \frac{y_2}{x_2^2+y_2^2} \implies (x_1 = x_2 \wedge y_1 = y_2)) \tag{1}$$

Challenge

Demonstrate automatically the truth of Formula 2 over reals.

$$\forall x_1 \forall x_2 \forall y_1 \forall y_2 \quad (y_1 > 0 \wedge y_2 > 0 \wedge x_1 + \frac{x_1}{x_1^2+y_1^2} = x_2 + \frac{x_2}{x_2^2+y_2^2} \wedge y_1 - \frac{y_1}{x_1^2+y_1^2} = y_2 - \frac{y_2}{x_2^2+y_2^2} \implies (x_1 = x_2 \wedge y_1 = y_2)) \tag{2}$$

- Convert the universal QE problem to two existential QE problems

$$\forall \mathbf{x}(A \implies (B \wedge C)) \text{ iff } \neg\exists \mathbf{x} \neg(A \implies (B \wedge C)). \text{ iff } \neg\exists \mathbf{x}\,((A \wedge \neg B) \vee (A \wedge \neg C))$$

- Use the option 'precondition'='TD' to precondition the input system by means of triangular decompositions.
- Use the option 'partial'='true' to enable partial lifting.

```
> t0 := time():
  f_1 := x_1+x_1/(x_1^2+y_1^2)-(x_2+x_2/(x_2^2+y_2^2)):
  f_2 := y_1-y_1/(x_1^2+y_1^2)-(y_2-y_2/(x_2^2+y_2^2)):
  g_1 := numer(normal(f_1)): g_2 := numer(normal(f_2)):

  sys := &not( (x_1^2+y_1^2-1>0) &and (x_2^2+y_2^2-1>0) &and (g_1=0) &and (g_2=0)
  &implies  (  (x_1=x_2) &and (y_1=y_2)   )  ):
  lsas := RegularChains:-TRDcadLogicFormulaToLsas(sys):
  nops(lsas); sys1 := lsas[1]: sys2 := lsas[2]:

  vars := SuggestVariableOrder(sys1): R := PolynomialRing(vars):
  out1 := CylindricalAlgebraicDecompose(sys1, R, output=rootof, precondition='TD',
  partial='true');

  vars := SuggestVariableOrder(sys2): R := PolynomialRing(vars):
  out2 := CylindricalAlgebraicDecompose(sys2, R, output=rootof, precondition='TD',
  partial='true');

  evalb( nops(out1)=0 and nops(out2)=0 ); t1 := time() - t0;
                                          2
                                    out1 := [ ]
                                    out2 := [ ]
                                      true
                                    t1 := 55.197
```

Solving the second challenge

```
> t0 := time():
  f_1 := x_1+x_1/(x_1^2+y_1^2)-(x_2+x_2/(x_2^2+y_2^2)):
  f_2 := y_1-y_1/(x_1^2+y_1^2)-(y_2-y_2/(x_2^2+y_2^2)):
  g_1 := numer(normal(f_1)): g_2 := numer(normal(f_2)):

  sys := &not( (y_1>0) &and (y_2>0) &and (g_1=0) &and (g_2=0) &implies  (  (x_1=
  x_2) &and (y_1=y_2)   )  ):
  lsas := RegularChains:-TRDcadLogicFormulaToLsas(sys):
  nops(lsas); sys1 := lsas[1]: sys2 := lsas[2]:

  vars := SuggestVariableOrder(sys1): R := PolynomialRing(vars):
  out1 := CylindricalAlgebraicDecompose(sys1, R, output=rootof, precondition='TD',
  partial='true');

  vars := SuggestVariableOrder(sys2): R := PolynomialRing(vars):
  out2 := CylindricalAlgebraicDecompose(sys2, R, output=rootof, precondition='TD',
  partial='true');

  evalb( nops(out1)=0 and nops(out2)=0 ); t1 := time() - t0;
                                    2
                              out1 := [ ]
                              out2 := [ ]
                                 true
                              t1 := 21.011
```

Outline

1 Introduction

2 Functionality

3 Underlying theory and technical contribution

4 Experimentation

5 Application

6 Conclusion

Summary and future work

- We have presented the command `CylindricalAlgebraicDecompose` of the `RegularChains` library.
- Yes, we should have better named it `CylindricallyAlgebraicDecompose` but most people who work around at Maplesoft or contribute to it are not native English speakers.
- The Maple library archive `RegularChains.mla` containing the latest version of `CylindricalAlgebraicDecompose` can be downloaded from `www.regularchains.org`
- The $\widehat{\text{RealRootIsolate}}$ command has been improved a lot as well as the CAD algorithm.
- The collaboration with the group at Bath U. has lead to the TTI-CAD option of `CylindricalAlgebraicDecompose` which is very effective.
- These improvements benefit to the `QunatifierElimination`, command, see the experimentation in our ISSAC 2014 paper.
- Further work is required to get simpler output QFF and partial cylindrical algebraic decompositions.