Using CAD Effectively

- Limitations of CAD
- Implementations of CAD
- Optimizing CAD for specific problems or problem classes

Limitations

- Complexity: Doubly exponential in the number of variables. Constructing a CAD via Collins' original algorithm takes time $(2n)^{2^{2r+8}} m^{2^{r+6}} d^3$ where $r =$ # of variables, $n =$ max degree of input in any variable, $m =$ # of input polynomials, $d =$ max bitlength of coefficients.

- Practical Observations:
  - random input is very bad!
  - non-random input is often not so bad, since the polynomials encountered in projection tend to factor a lot
  - specializing projection and lifting to specific input types often ameliorates the high completxity

Implementations

- QEPCAD — written mostly by Hoon Hong, but with contributions by many others, including George Collins.
- QEPCAD B — based on QEPCAD but with many extensions and improvements.
- RLCAD — due to Andreas Seidl & Thomas Sturm, part of the Redlog system.
- Mathematica's CAD — written by Adam Strzeboñski.

Using CAD to Solve Problems Efficiently

1. Variable Ordering
2. Prepare input - break into pieces, do trivial eliminations by hand, etc.
3. Partial CAD
4. Special case of Partial CAD: full-dimensional cells only
5. Go beyond $\exists$ and $\forall$: Use the structure of CADs

Variable Ordering

The variable ordering you use can make a big difference!

Project $\{y^2 - x + 1, y^3 - y + x\}$ with order $x \prec y$ and get ...

Variable Ordering

The variable ordering you use can make a big difference!

Project $\{y^2 - x + 1, y^3 - y + x\}$ with order $x \prec y$ and get ...

$$\{x - 1, 27x^2 - 4, x^3 - 6x^2 + 8x - 4\}$$

Variable Ordering

The variable ordering you use can make a big difference!

Project $\{y^2 - x + 1, y^3 - y + x\}$ with order $x \prec y$ and get ...

$$\{x - 1, 27x^2 - 4, x^3 - 6x^2 + 8x - 4\}$$

Project $\{y^2 - x + 1, y^3 - y + x\}$ with order $y \prec x$ and get ...

Variable Ordering

The variable ordering you use can make a big difference!

Project $\{y^2 - x + 1, y^3 - y + x\}$ with order $x \prec y$ and get ...

$$\{x - 1, 27x^2 - 4, x^3 - 6x^2 + 8x - 4\}$$

Project $\{y^2 - x + 1, y^3 - y + x\}$ with order $y \prec x$ and get ...

$$\{y^3 + y^2 - y + 1\}$$

More on Variable Orderings

- The problem may constrain orderings.
- Example of a simple heuristic:
  1. Descending order by degree of variable, breaking ties with
  2. Descending order by highest total-degree term in which the variable appears, breaking ties with
  3. Descending order by number of terms containing the variable
- The technical report "Efficient Projection Orders for CAD", Dolzmann, Seidl & Sturm, examines problem and proposes a greedy algorithm for constructing good projection orders.

Prepare Input: Make Trivial Substitutions

Let C_1 and C_2 be circles whose centers are 10 units apart and whose radii are 1 and 3. Find, with proof, the locus of all points M for which there exist points X on C_1 and Y on C_2 such that M is the midpoint of the line segment XY.
—*Recent Putnam Question*

$$\exists x_1, y_1, x_2, y_2[x_1^2 + y_1^2 - 1 = 0 \wedge (x_2 - 10)^2 + y_2^2 - 9 = 0 \wedge x = \frac{x_1+x_2}{2} \wedge y = \frac{y_1+y_2}{2}]$$

Prepare Input: Make Trivial Substitutions

Let C_1 and C_2 be circles whose centers are 10 units apart and whose radii are 1 and 3. Find, with proof, the locus of all points M for which there exist points X on C_1 and Y on C_2 such that M is the midpoint of the line segment XY.
—*Recent Putnam Question*

$$\exists x_1, y_1, x_2, y_2[x_1^2 + y_1^2 - 1 = 0 \wedge (x_2 - 10)^2 + y_2^2 - 9 = 0 \wedge x = \tfrac{x_1+x_2}{2} \wedge y = \tfrac{y_1+y_2}{2}]$$

Substitute $x_2 = 2x + x_1$ and $y_2 = 2y + y_1$, producing:

$$\exists x_1, y_1[x_1^2 + y_1^2 - 1 = 0 \wedge (2x - x_1 - 10)^2 + (2y - y_1)^2 - 9 = 0].$$

Prepare Input: Make Trivial Substitutions

Let C_1 and C_2 be circles whose centers are 10 units apart and whose radii are 1 and 3. Find, with proof, the locus of all points M for which there exist points X on C_1 and Y on C_2 such that M is the midpoint of the line segment XY.
—*Recent Putnam Question*

$$\exists x_1, y_1, x_2, y_2[x_1^2 + y_1^2 - 1 = 0 \wedge (x_2 - 10)^2 + y_2^2 - 9 = 0 \wedge x = \tfrac{x_1+x_2}{2} \wedge y = \tfrac{y_1+y_2}{2}]$$

Substitute $x_2 = 2x + x_1$ and $y_2 = 2y + y_1$, producing:

$$\exists x_1, y_1[x_1^2 + y_1^2 - 1 = 0 \wedge (2x - x_1 - 10)^2 + (2y - y_1)^2 - 9 = 0].$$

Prepare Input: Break Problems into Pieces

$$\exists c[ab = b + 1 - c^2 \wedge 2(a + b)c^2 - b^2 + c - 1 = 0 \wedge a^2 + b^2 + c^2 \leq 4]$$

Prepare Input: Break Problems into Pieces

$$\exists c[ab = b + 1 - c^2 \wedge 2(a+b)c^2 - b^2 + c - 1 = 0 \wedge a^2 + b^2 + c^2 \leq 4]$$

Solving for a and subsituting requires distinguishing cases $b \neq 0$ and $b = 0$:

$$\exists c \left[\begin{array}{c} b \neq 0 \wedge 2(\frac{b+1-c^2}{b} + b)c^2 - b^2 + c - 1 = 0 \wedge (\frac{b+1-c^2}{b})^2 + b^2 + c^2 \leq 4 \\ \vee \\ b = 0 \wedge 0 = 1 - c^2 \wedge 2ac^2 + c - 1 = 0 \wedge a^2 + c^2 \leq 4 \end{array} \right]$$

Prepare Input: Break Problems into Pieces

$$\exists c[ab = b + 1 - c^2 \wedge 2(a+b)c^2 - b^2 + c - 1 = 0 \wedge a^2 + b^2 + c^2 \leq 4]$$

Solving for a and subsituting requires distinguishing cases $b \neq 0$ and $b = 0$:

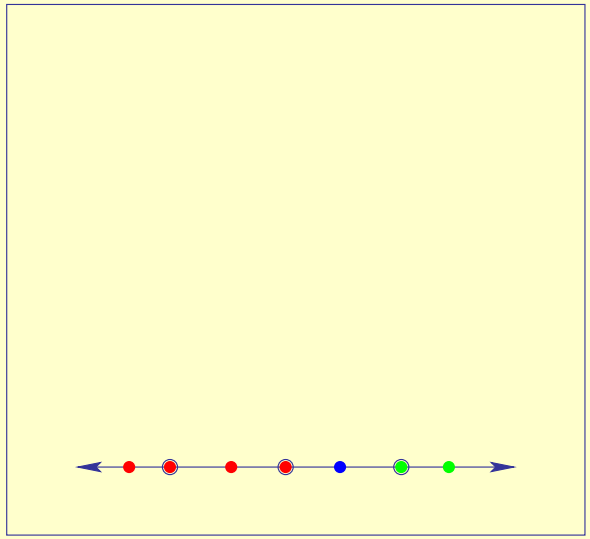
$$\exists c \left[\begin{array}{c} b \neq 0 \wedge 2(\frac{b+1-c^2}{b} + b)c^2 - b^2 + c - 1 = 0 \wedge (\frac{b+1-c^2}{b})^2 + b^2 + c^2 \leq 4 \\ \vee \\ b = 0 \wedge 0 = 1 - c^2 \wedge 2ac^2 + c - 1 = 0 \wedge a^2 + c^2 \leq 4 \end{array} \right]$$

Instead of solving this problem with CAD directly, split it into:

$$\exists c \left[b \neq 0 \wedge 2(\frac{b+1-c^2}{b} + b)c^2 - b^2 + c - 1 = 0 \wedge (\frac{b+1-c^2}{b})^2 + b^2 + c^2 \leq 4 \right]$$
$$\vee$$
$$b = 0 \wedge \exists c \left[0 = 1 - c^2 \wedge 2ac^2 + c - 1 = 0 \wedge a^2 + c^2 \leq 4 \right]$$

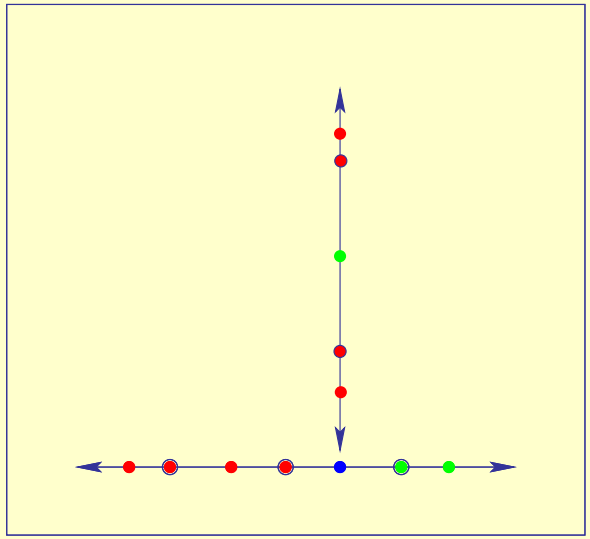
Partial CAD

- Collins & Hong introduced *Partial* CAD.
- Partial CAD is basically a lazy approach to lifting.



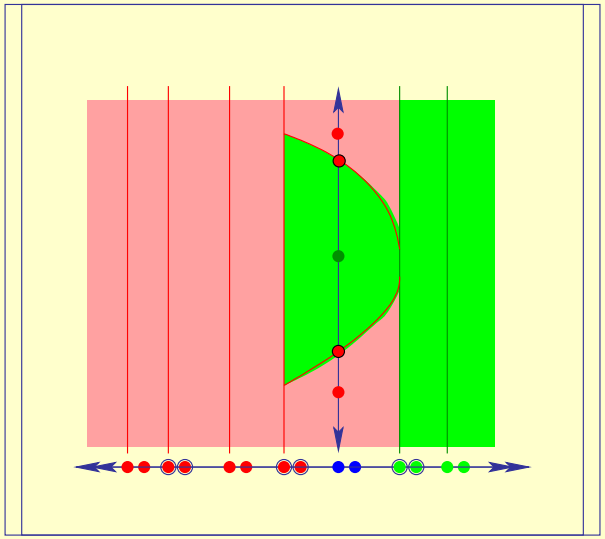
Partial CAD

- Collins & Hong introduced *Partial* CAD.
- Partial CAD is basically a lazy approach to lifting.



Partial CAD

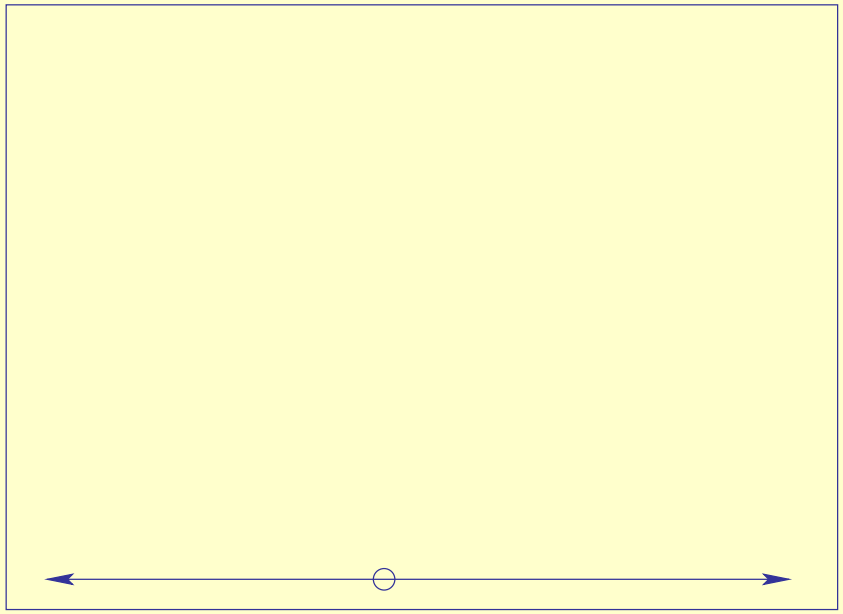
- Collins & Hong introduced *Partial* CAD.
- Partial CAD is basically a lazy approach to lifting.



Partial CAD & Quantifier Elimination

- A CAD data-structure is like a tree. Propagating $\exists$ and $\forall$ is like AI search in that tree. We can consider different search strategies too.

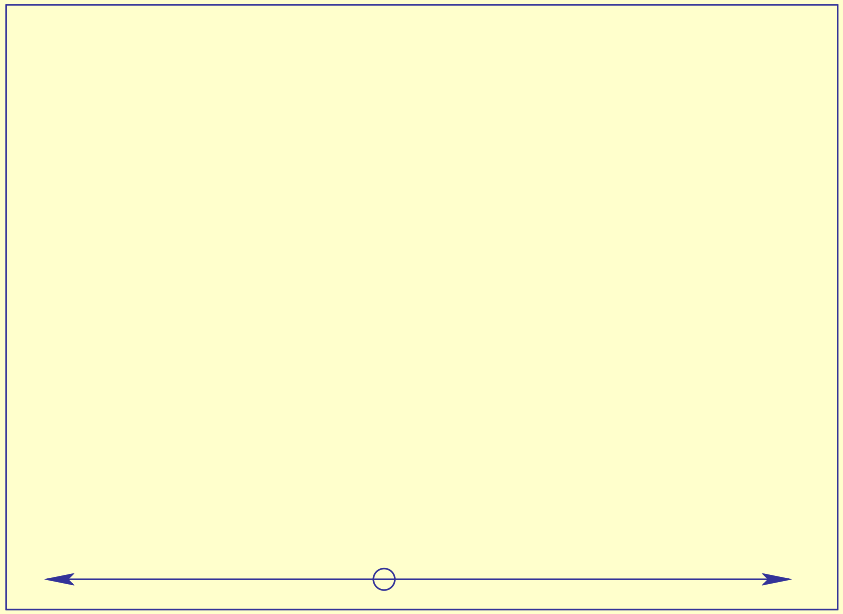
Example: $\exists y \forall z \ldots$



Partial CAD & Quantifier Elimination

- A CAD data-structure is like a tree. Propagating $\exists$ and $\forall$ is like AI search in that tree. We can consider different search strategies too.

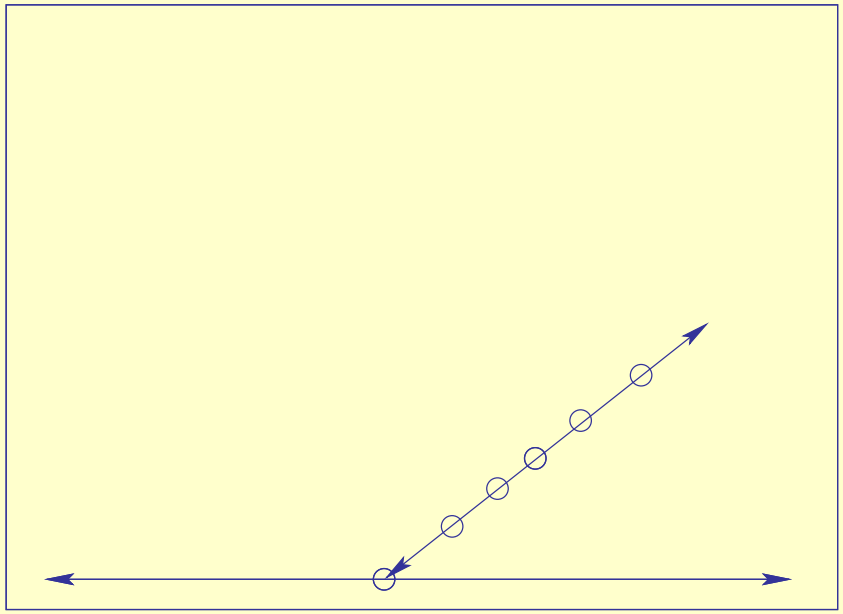
Example: $\exists y \forall z \ldots$



Partial CAD & Quantifier Elimination

- A CAD data-structure is like a tree. Propagating $\exists$ and $\forall$ is like AI search in that tree. We can consider different search strategies too.

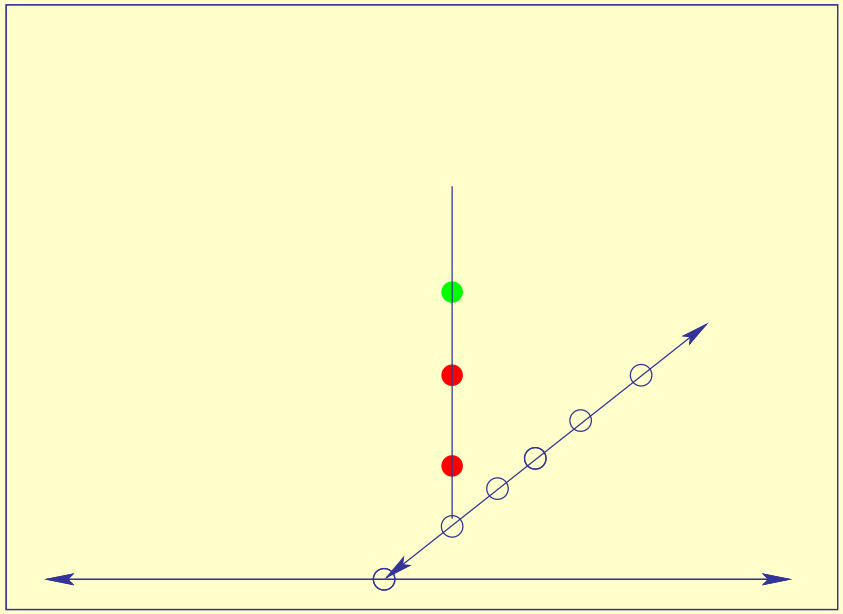
Example: $\exists y \forall z \ldots$



Partial CAD & Quantifier Elimination

- A CAD data-structure is like a tree. Propagating $\exists$ and $\forall$ is like AI search in that tree. We can consider different search strategies too.

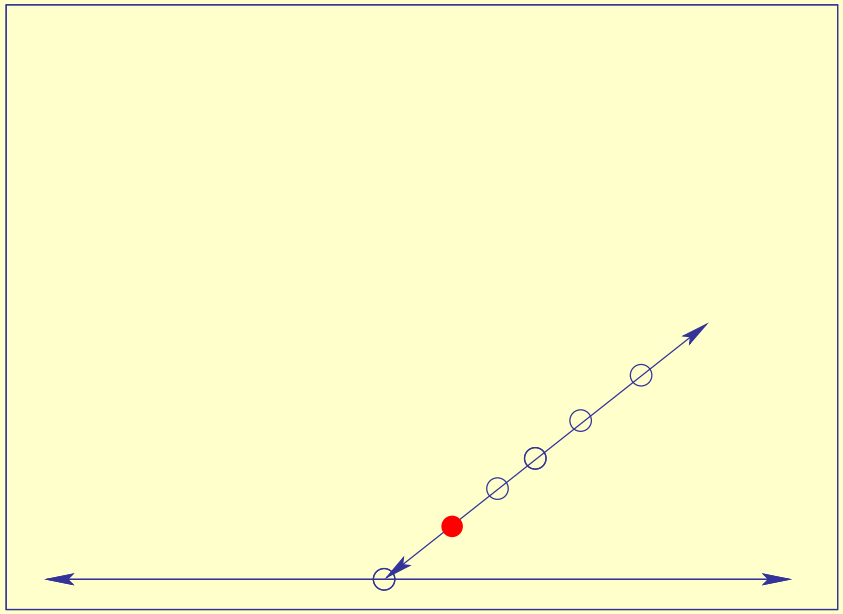
Example: $\exists y \forall z \ldots$



Partial CAD & Quantifier Elimination

- A CAD data-structure is like a tree. Propagating $\exists$ and $\forall$ is like AI search in that tree. We can consider different search strategies too.

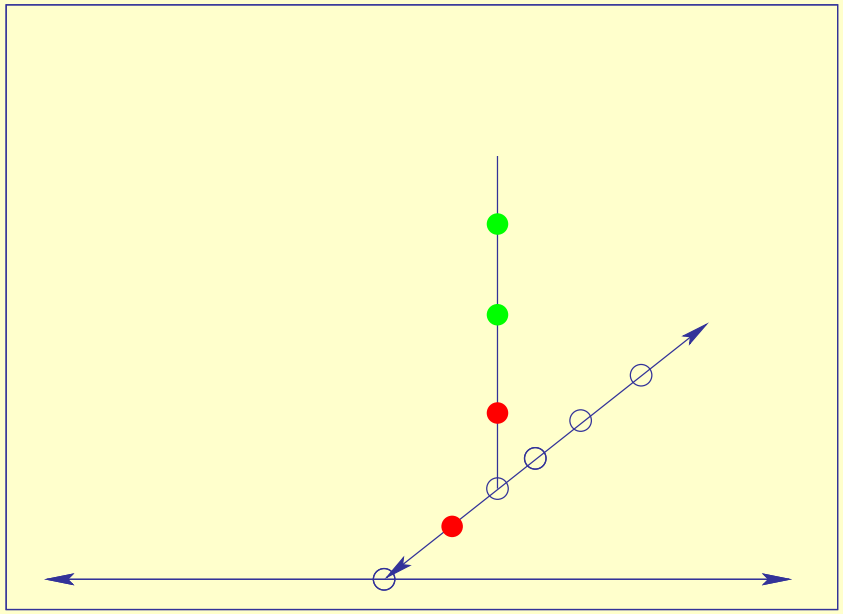
Example: $\exists y \forall z \ldots$



Partial CAD & Quantifier Elimination

- A CAD data-structure is like a tree. Propagating $\exists$ and $\forall$ is like AI search in that tree. We can consider different search strategies too.

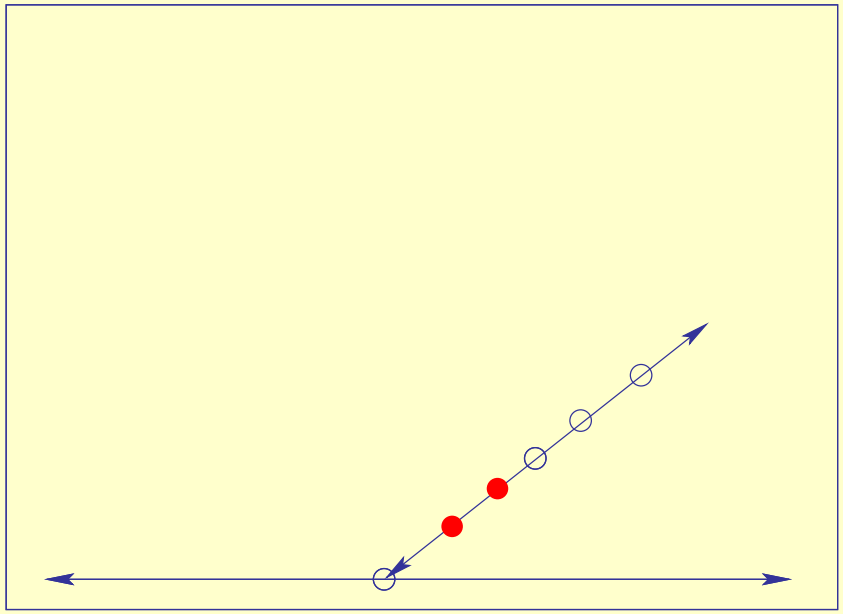
Example: $\exists y \forall z \ldots$



Partial CAD & Quantifier Elimination

- A CAD data-structure is like a tree. Propagating $\exists$ and $\forall$ is like AI search in that tree. We can consider different search strategies too.

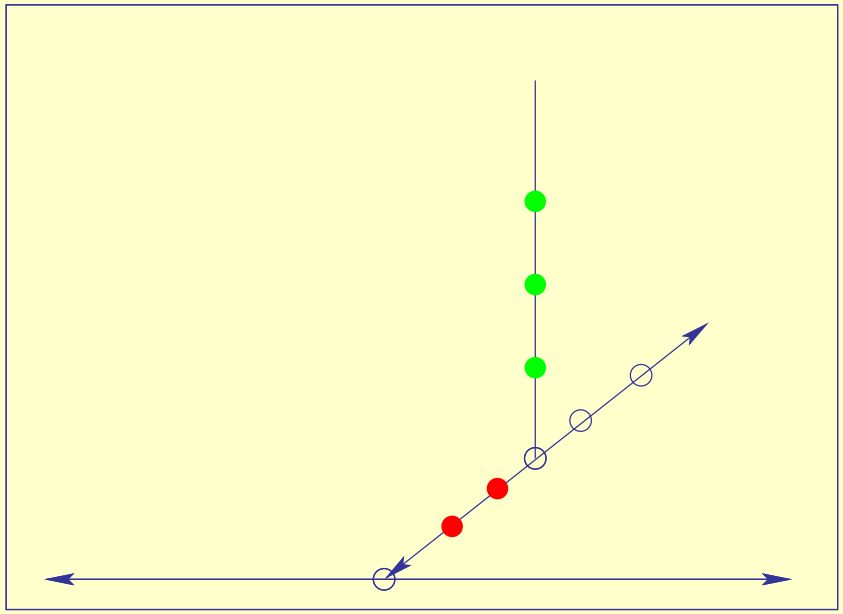
Example: $\exists y \forall z \ldots$



Partial CAD & Quantifier Elimination

- A CAD data-structure is like a tree. Propagating $\exists$ and $\forall$ is like AI search in that tree. We can consider different search strategies too.

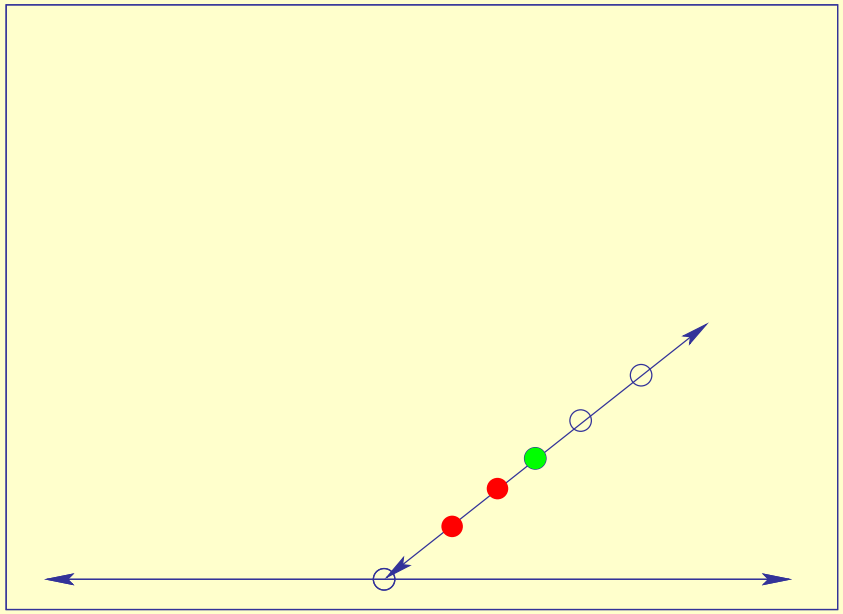
Example: $\exists y \forall z \ldots$



Partial CAD & Quantifier Elimination

- A CAD data-structure is like a tree. Propagating $\exists$ and $\forall$ is like AI search in that tree. We can consider different search strategies too.

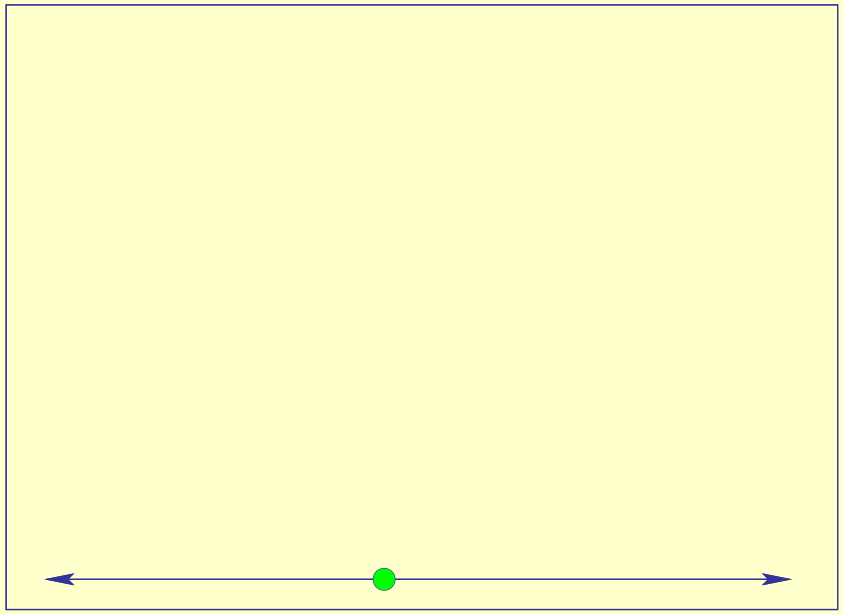
Example: $\exists y \forall z \ldots$



Partial CAD & Quantifier Elimination

- A CAD data-structure is like a tree. Propagating $\exists$ and $\forall$ is like AI search in that tree. We can consider different search strategies too.

Example: $\exists y \forall z \ldots$



Partial CAD: Full dimensional cells

- Special case of partial CAD: only lift over full dimensional cells. Particularly desirable because:

  1. No algebraic number computations.
  2. Projection is simpler.

  Huge reduction in computing time in most cases! Very easy to implement!

- McCallum and Strzebonski both applied this idea to solving systems of strict polynomial inequalities, where the solution set is open.

- Could consider new quantifiers "for all but finitely many" and "exists infinitely many" that can be decided based only on truth values of full dimensional cells.

Full dimensional cells example

- "An effective decision method for semidefinite polynomials", Guangxing & Xiaoning, JSC 2004.

Full dimensional cells example

- "An effective decision method for semidefinite polynomials", Guangxing & Xiaoning, JSC 2004. Their Ex. 4 asks whether the following polynomial is semi-definite:

$$w^6 + 2z^2w^3 + x^4 + y^4 + z^4 + 2x^2w + 2x^2z + 3x^2 + w^2 + 2zw + z^2 + 2z + 2w + 1$$

Full dimensional cells example

- "An effective decision method for semidefinite polynomials", Guangxing & Xiaoning, JSC 2004. Their Ex. 4 asks whether the following polynomial is semi-definite:

$$w^6+2z^2w^3+x^4+y^4+z^4+2x^2w+2x^2z+3x^2+w^2+2zw+z^2+2z+2w+1$$

- $p(x_1,\ldots,x_k)$ is not positive semi-definite if and only if $p<0$ at some point α, in which case $p<0$ for some neighborhood around α.

- Consider a CAD for p. p is not positive semi-definite if and only if $p<0$ in some full dimensional cell.

- QEPCAD B decides Ex. 4 is semi-definite in 0.3 seconds (on this laptop) when only full-dimensional cells are considered. (order $w \prec z \prec x \prec y$)

Full dimensional cells: Part II

- *Approximate*: Perform Q.E. or formula simplification only for full dimensional cells in free variable space. Answer correct up to some measure zero subset of parameter space — i.e. the lower dimensional cells. For parameters with physical meaning this is good enough.

- *Generic* Q.E. (Seidl & Sturm): Lift over all cells except sections of certain projection factors.
  - Choose projection factors whose possible vanishing requires us to increase projection size.
  - Don't lift over sections of chosen factors so projection size kept smaller.
  - Output solution formula with formula stating that the chosen projection factors are assumed not to be zero (the *theory*).
  - Answer is exact given the theory.

Use CAD Properties in Novel Ways

- CADs tell you *lots* about the projection factors. Exploit this! Adapt CAD to new problems rather than try to phrase things as QE problems.

- Example: Characterize the monic quartic polynomials with all four roots real and distinct.
  - As Q.E. problem needs 4 free and 4 bound variables.
  - Use CAD of $x^4 + ax^3 + bx^2 + cx + d$, with order $a \prec b \prec c \prec d \prec x$
  - Over each cell in $\mathbb{R}^4$ count roots & assign truth values.
  - This solution needs only 4 free and 1 bound variable.

- Anai & Parrilo, "*Convex Quantifier Elimination for Semi-Definite Programming*" is a nice example of specializing CAD to a particular problem.