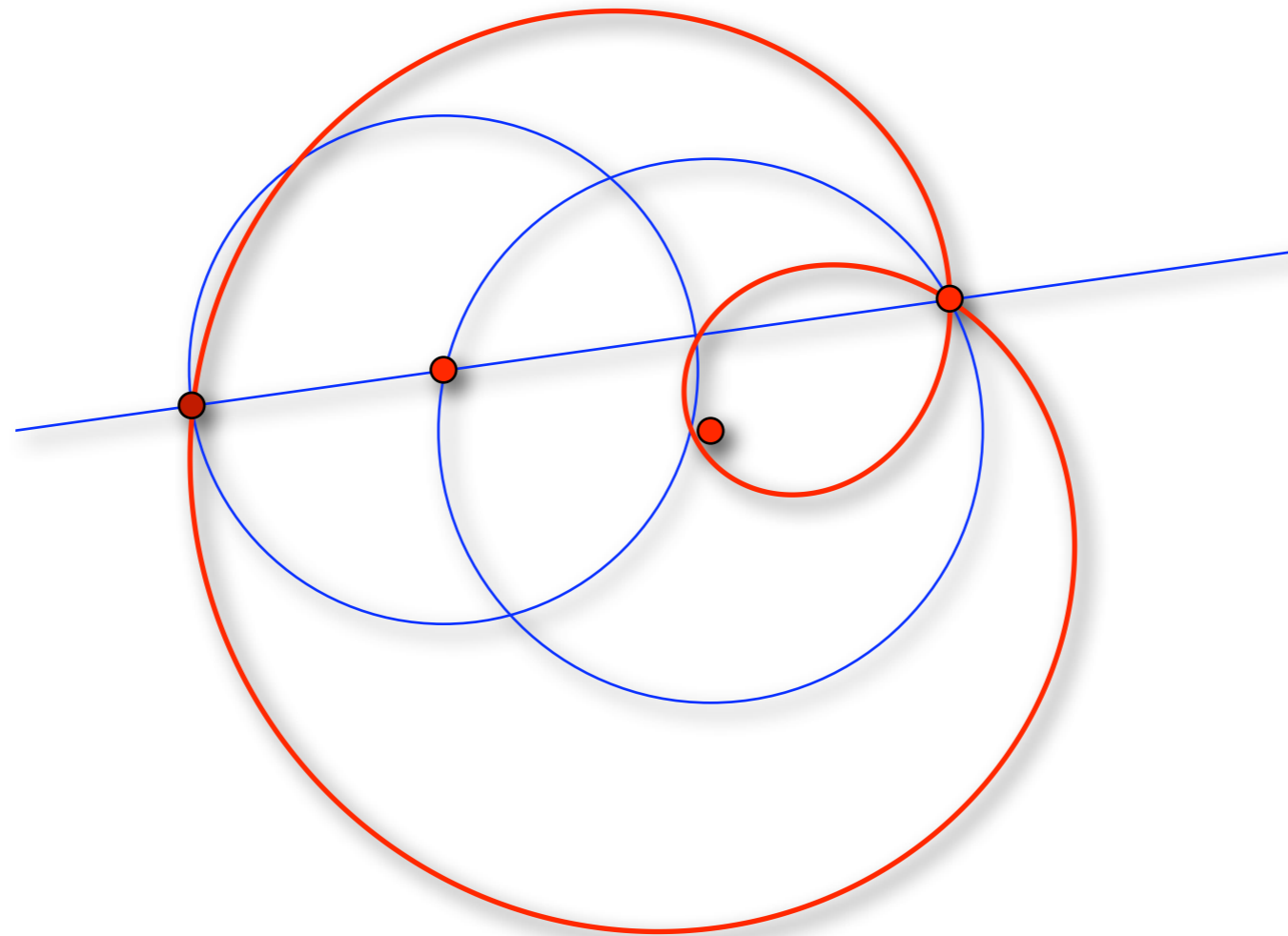


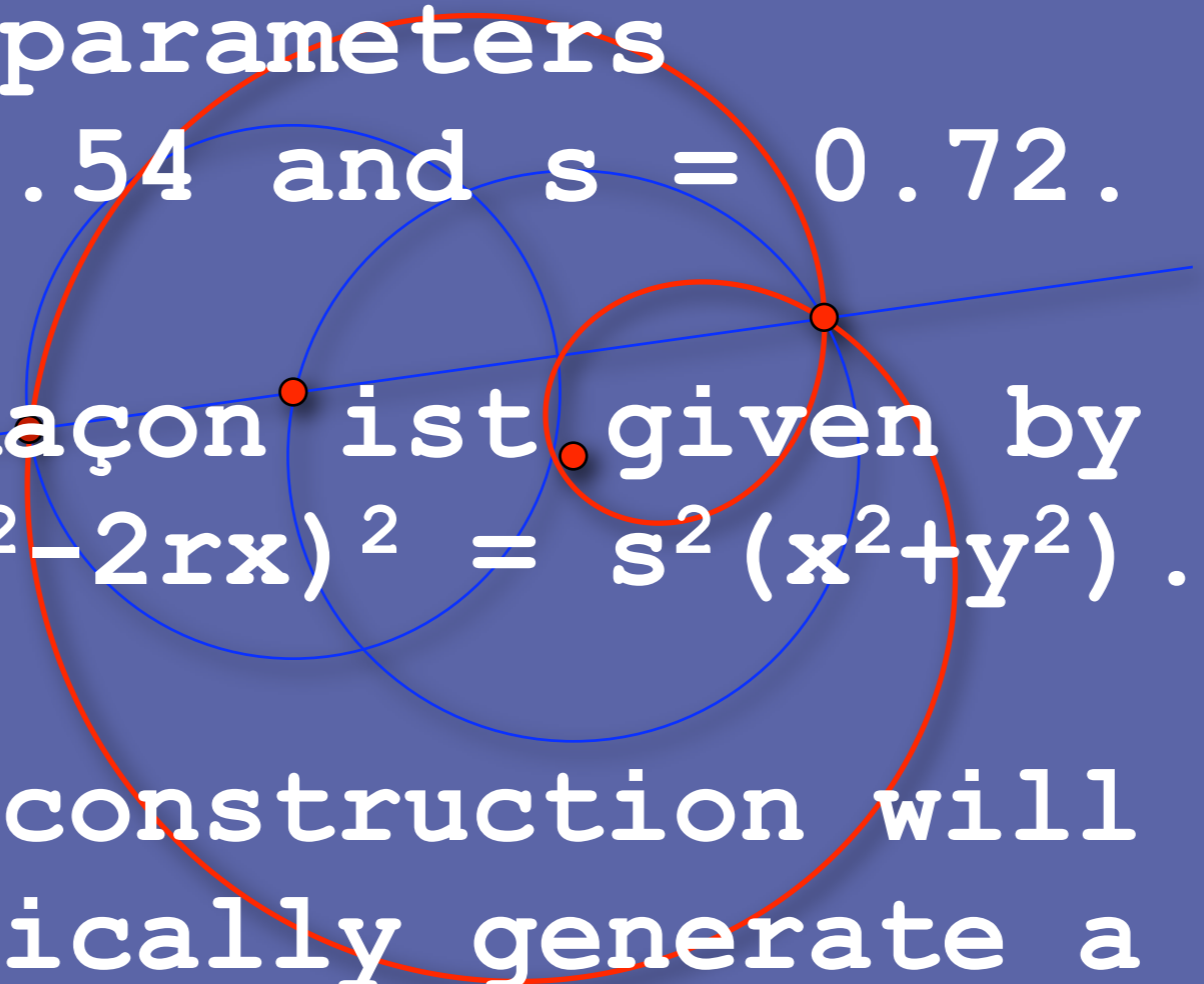
# Recognition of computationally constructed loci

Peter Lebmeir and Jürgen Richter-Gebert

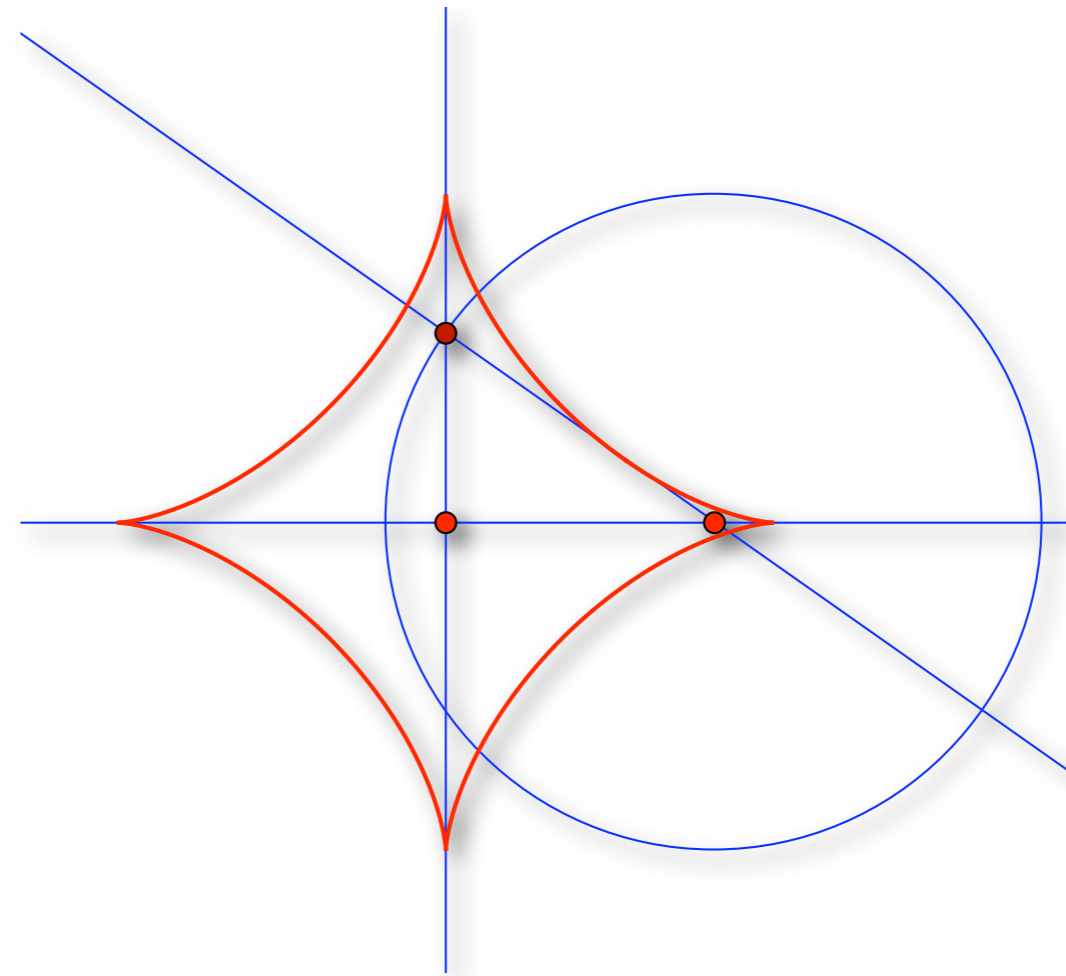
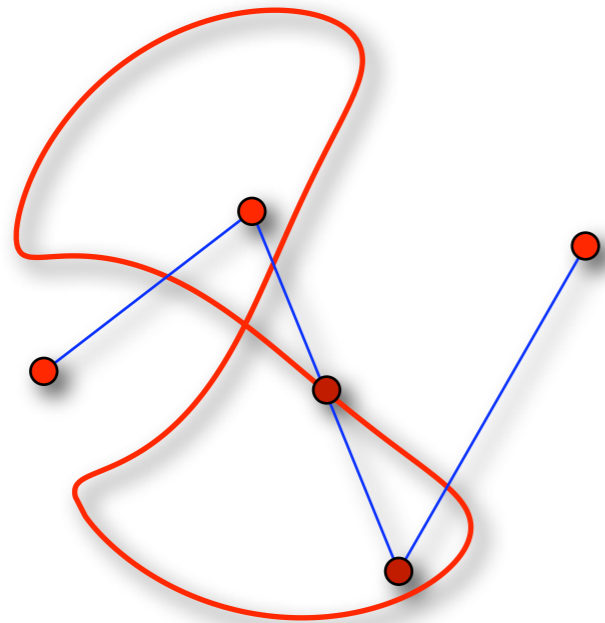


Welcome!

## Optimum:

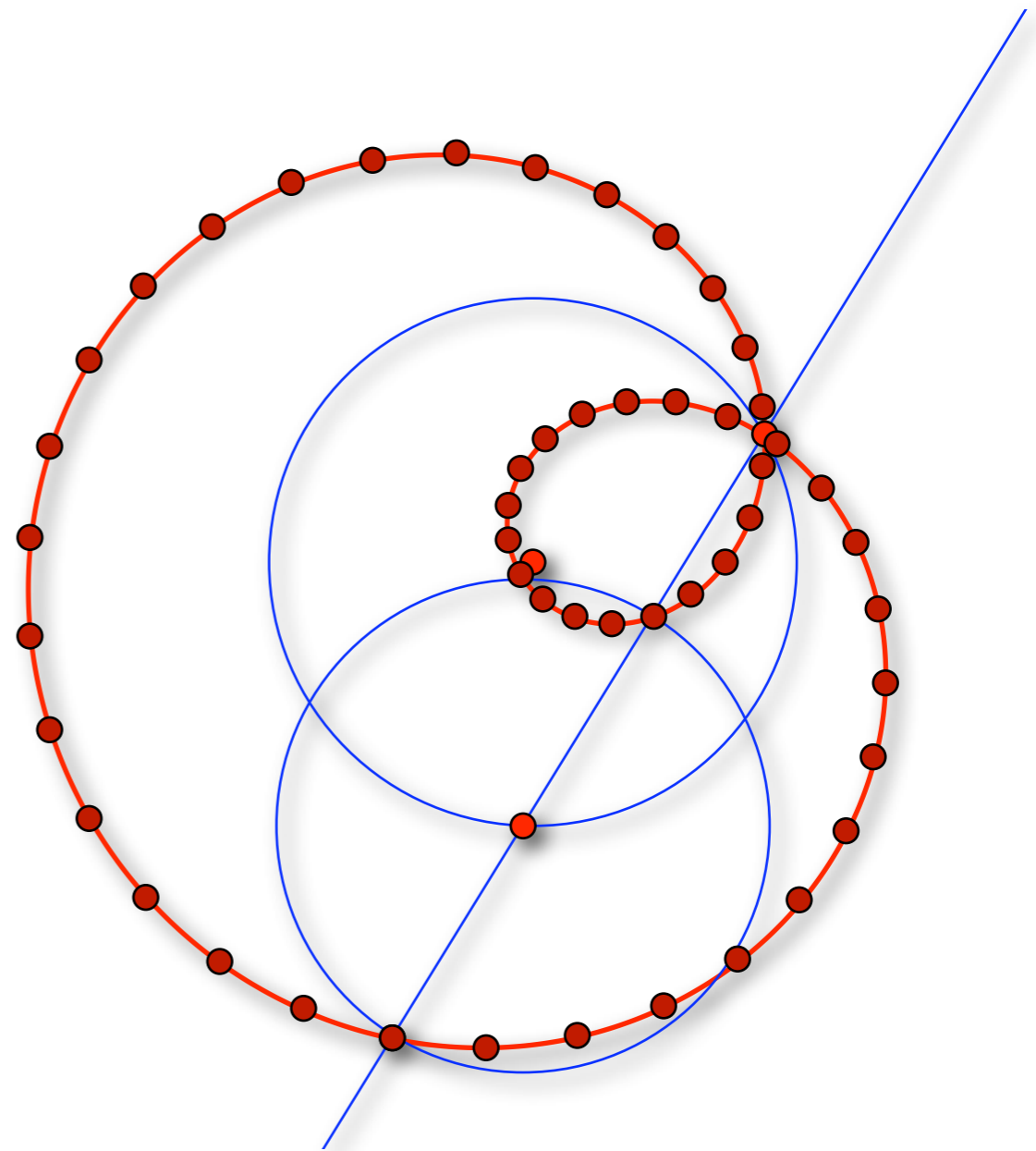
- You constructed a limaçon with parameters  $r = 0.54$  and  $s = 0.72$ .
  - A limaçon ist given by  $(x^2 + y^2 - 2rx)^2 = s^2(x^2 + y^2)$ .
  - Your construction will generically generate a limaçon.
- 
- A geometric diagram illustrating the construction of a limaçon. It shows two blue circles of different radii. A red circle is drawn with its center at the intersection of the two blue circles. A red limaçon is drawn, passing through the intersection point of the two blue circles. A blue line passes through the center of the larger blue circle and the intersection point of the two blue circles, intersecting the red limaçon at two points. The intersection point on the right is marked with a red dot.

## Examples:

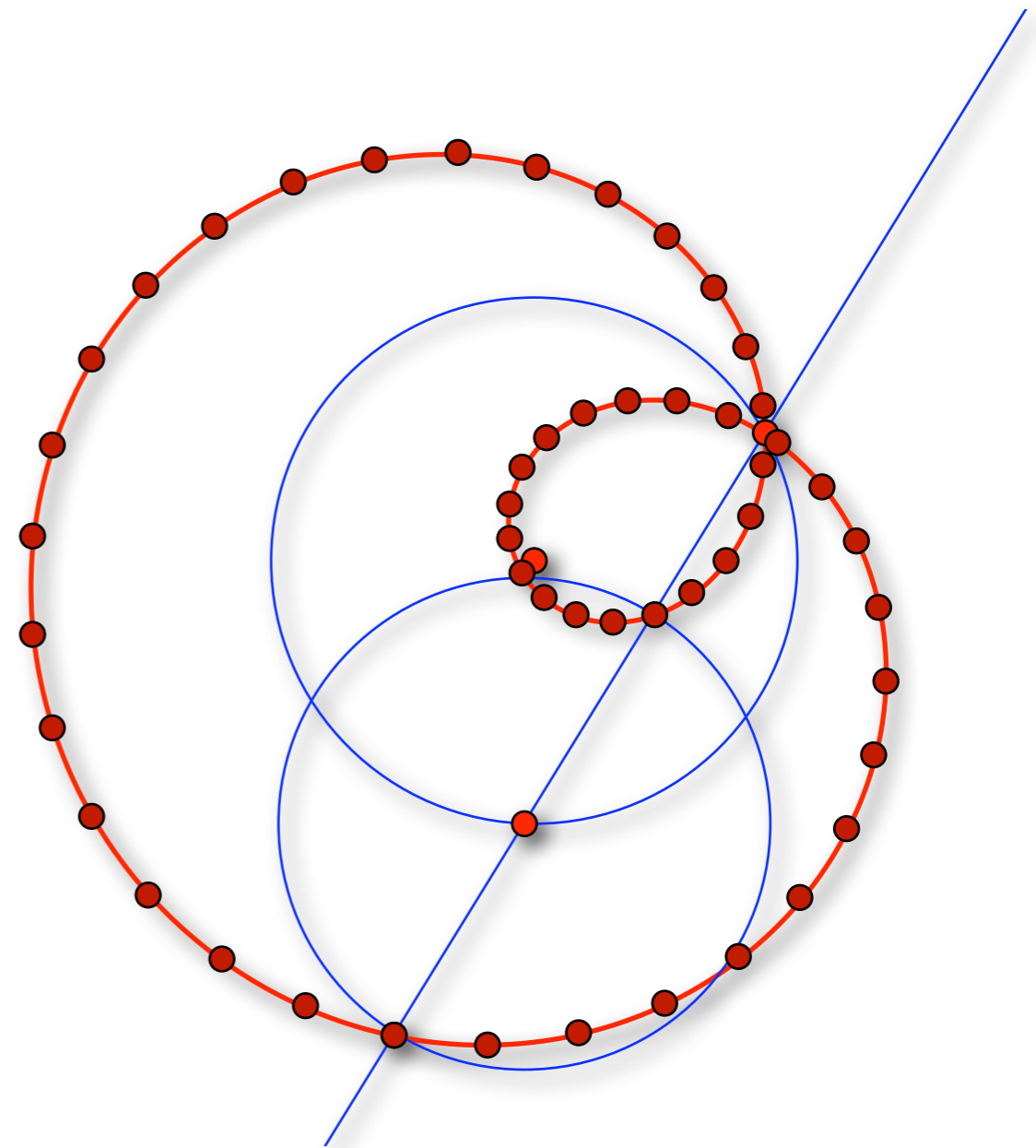


## Locus:

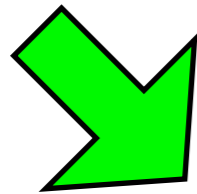
- Mover & Tracer
- Dyn. Geometry Program selects mover-positions
- high-precision tracer-positions  
→ Locus



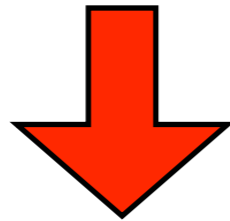
- What's this locus?
- Which loci result from the construction?



mover



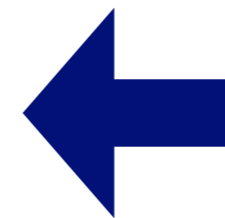
free and



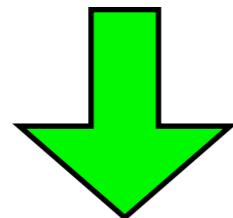
semi-free elements



Black Box:  
Construction

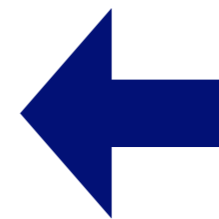


**Ruler & Compass**



precise tracer-positions  $\mathcal{P}$

$$Z(\mathbf{b}) = \{x \in \mathbb{RP}^2 \mid b(x) = 0\}$$



**Algebraic  
Curve**

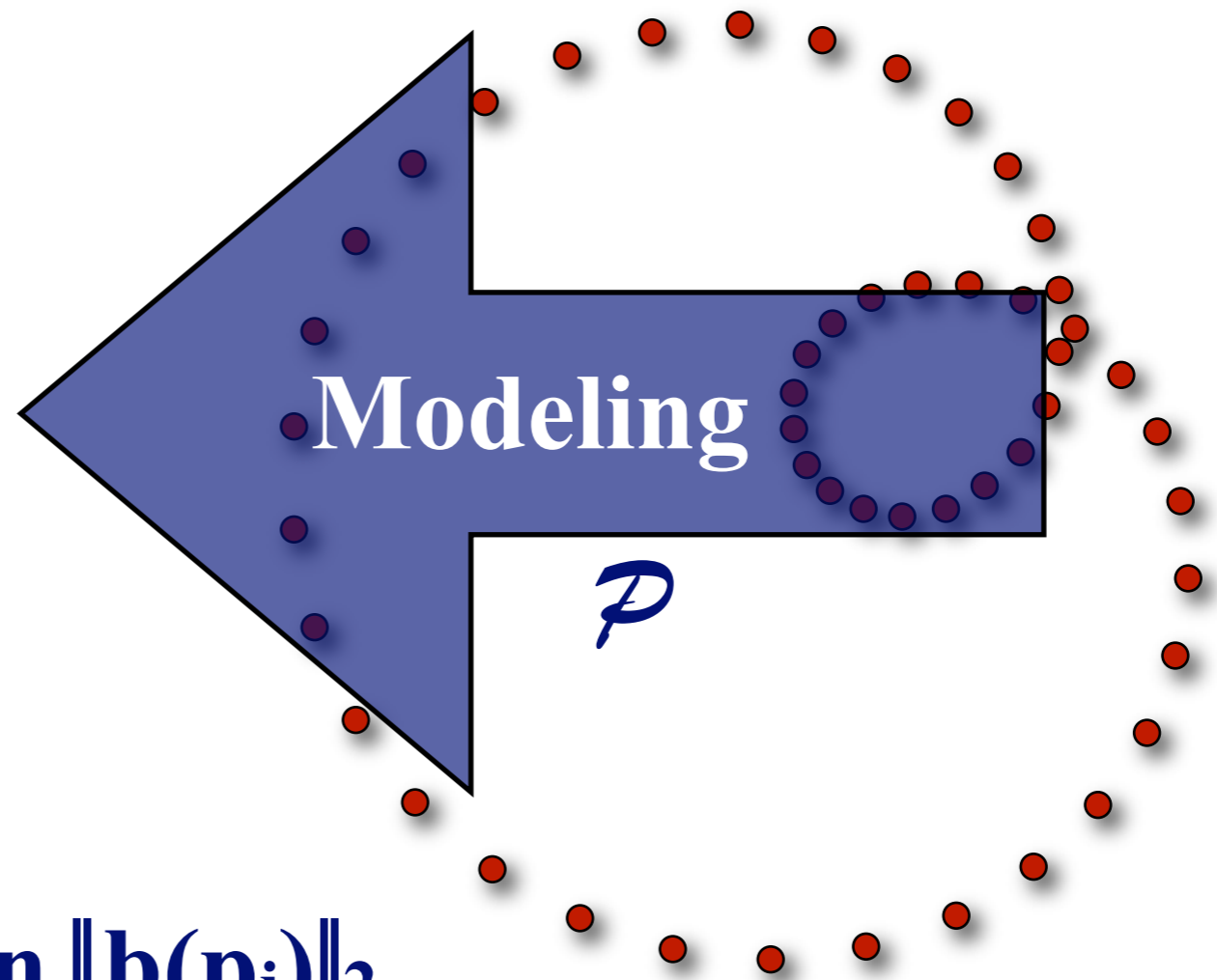
Which algebraic curve  $b(x) = 0$  of minimal degree is “reasonably” fitting  $\mathcal{P} = \{p_1, p_2, \dots, p_m\}$  ?

- $\frac{1}{m} \sum_{i=1}^m \text{dist}(p_i, Z(b))$

- $\min \frac{1}{m} \sum_{i=1}^m b^2(p_i)$

- $\min \sqrt{\sum_{i=1}^m b^2(p_i)}$

$$= \min \|b(p_i)\|_2$$

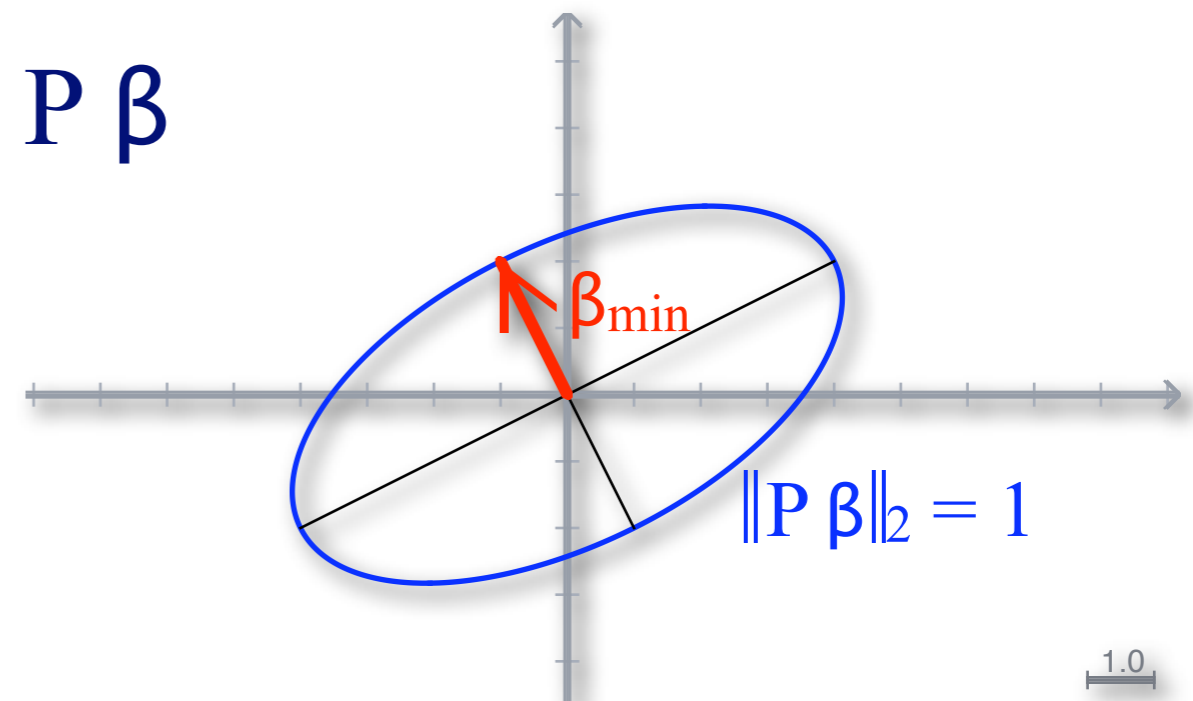


$$b(p) = \beta_1 x^d + \beta_2 x^{d-1} y + \beta_3 x^{d-1} z + \beta_4 x^{d-2} y^2 + \dots + \beta_k z^d = 0$$

$$\min_{\|\beta\|_2 = 1} \|b(p_i)\|_2 = \min_{\|\beta\|_2 = 1} \left\| \begin{bmatrix} x_1^d & x_1^{d-1} y_1 & \dots & z_1^d \\ x_2^d & x_2^{d-1} y_2 & \dots & z_2^d \\ \vdots & \vdots & \ddots & \vdots \\ x_m^d & x_m^{d-1} y_m & \dots & z_m^d \end{bmatrix} \beta \right\|_2$$

$$= \min_{\|\beta\|_2 = 1} \|P \beta\|_2 = \min_{\|\beta\|_2 = 1} \beta^T P^T P \beta$$

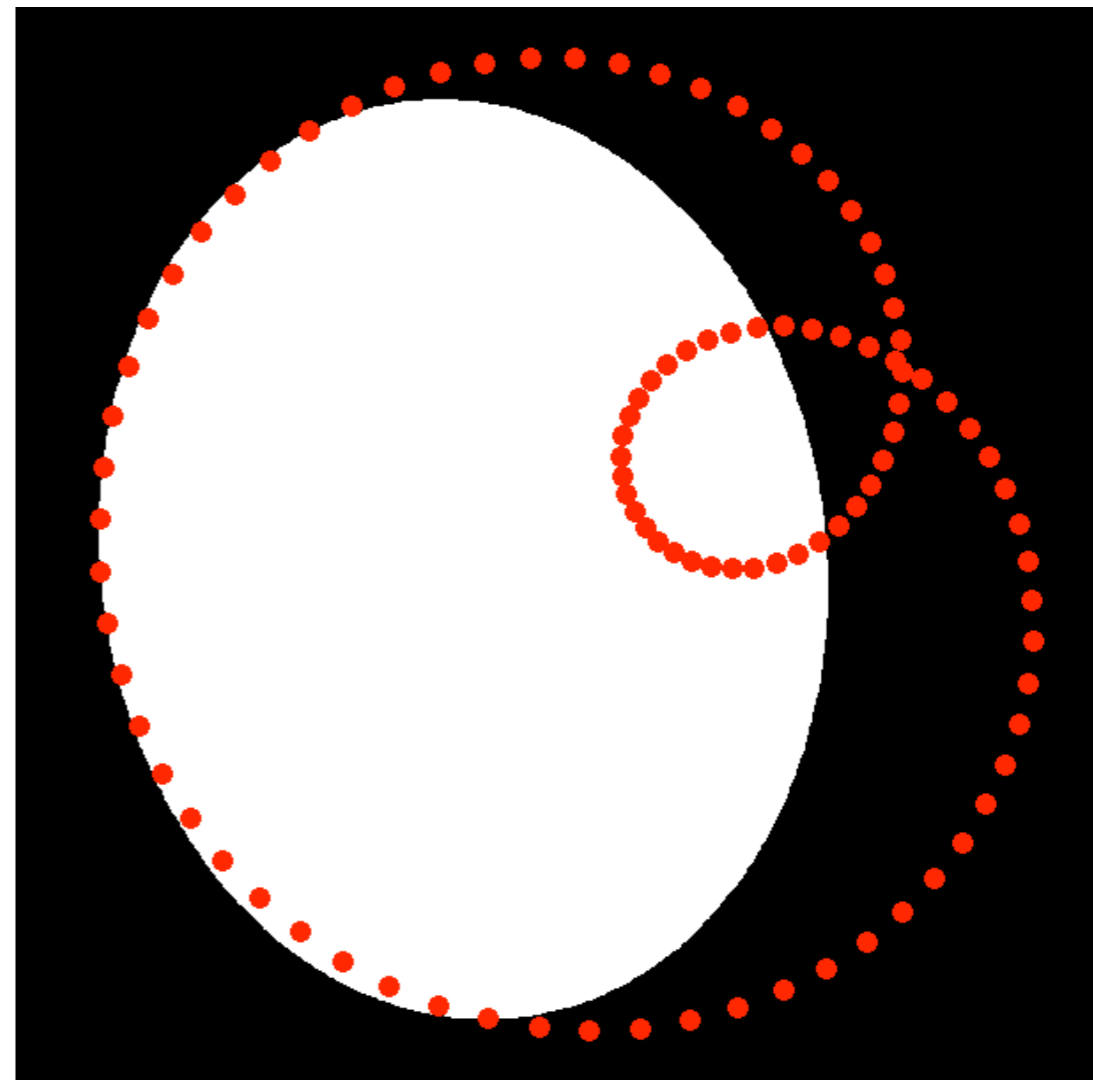
$$= \min_{\lambda \in \sigma(P^T P)} |\lambda|$$



## Algorithm:

- Given:  $\mathcal{P}$
- degree-assumption  
 $\deg(\mathbf{b}) = 2$
- $\min_{\lambda \in \sigma(\mathbf{P}^T \mathbf{P})} |\lambda|$

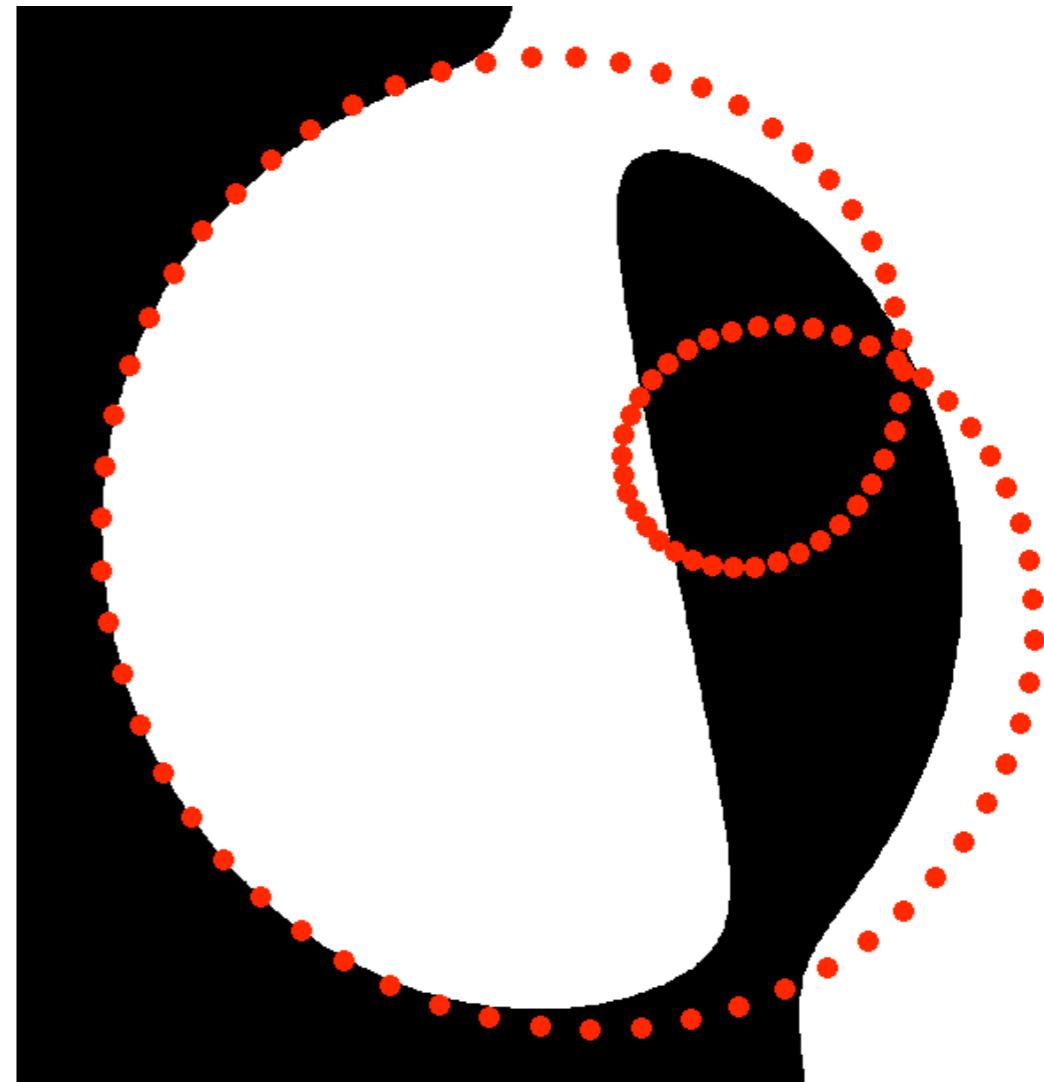
➔  $|\lambda| \approx 0.018$



## Algorithm:

- Given:  $\mathcal{P}$
- degree-assumption  
 $\deg(\mathbf{b}) = 3$
- $\min_{\lambda \in \sigma(\mathbf{P}^T \mathbf{P})} |\lambda|$

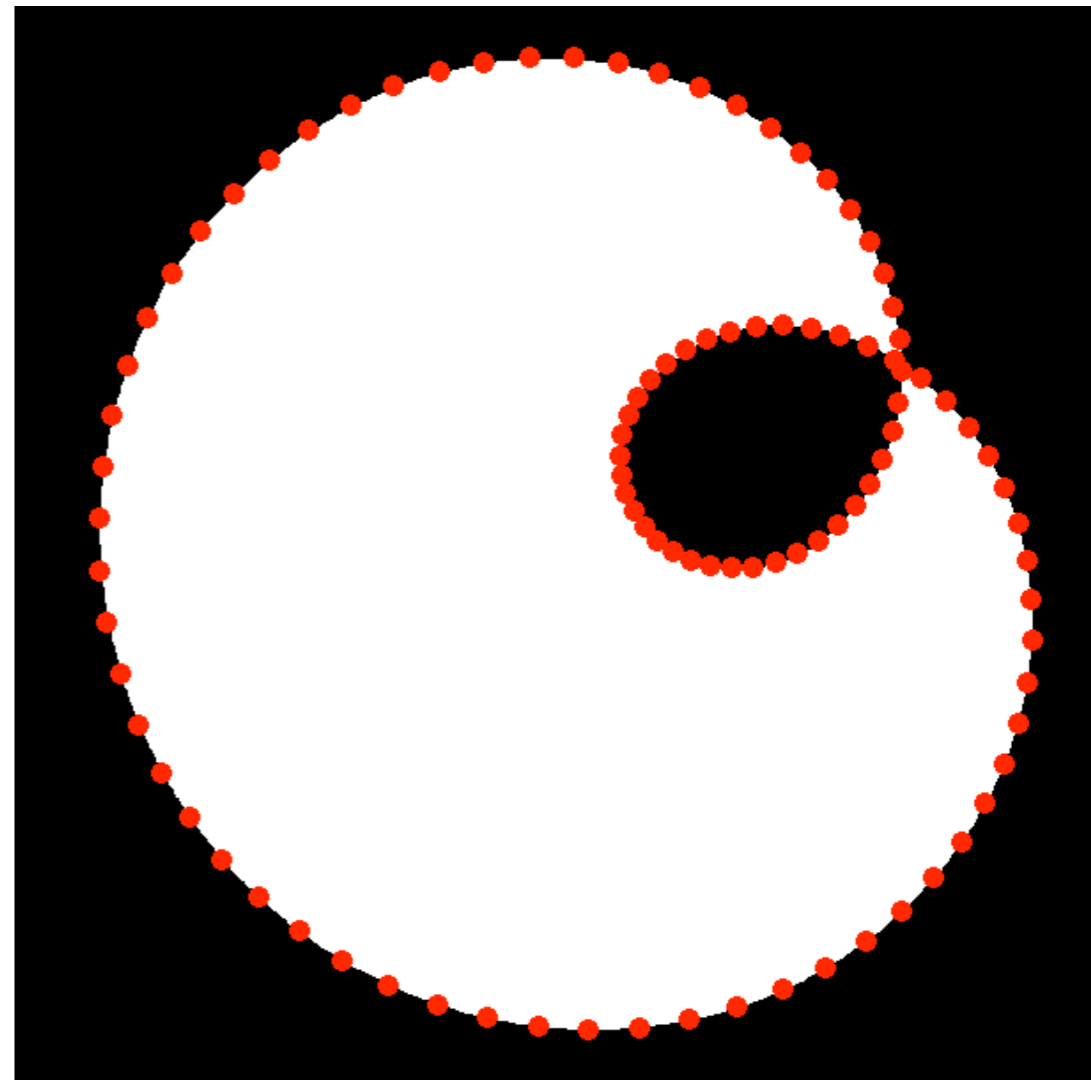
➔  $|\lambda| \approx 4.4 \cdot 10^{-6}$



## Algorithm:

- Given:  $\mathcal{P}$
- degree-assumption  
 $\deg(\mathbf{b}) = 4$
- $\min_{\lambda \in \sigma(\mathbf{P}^T \mathbf{P})} |\lambda|$

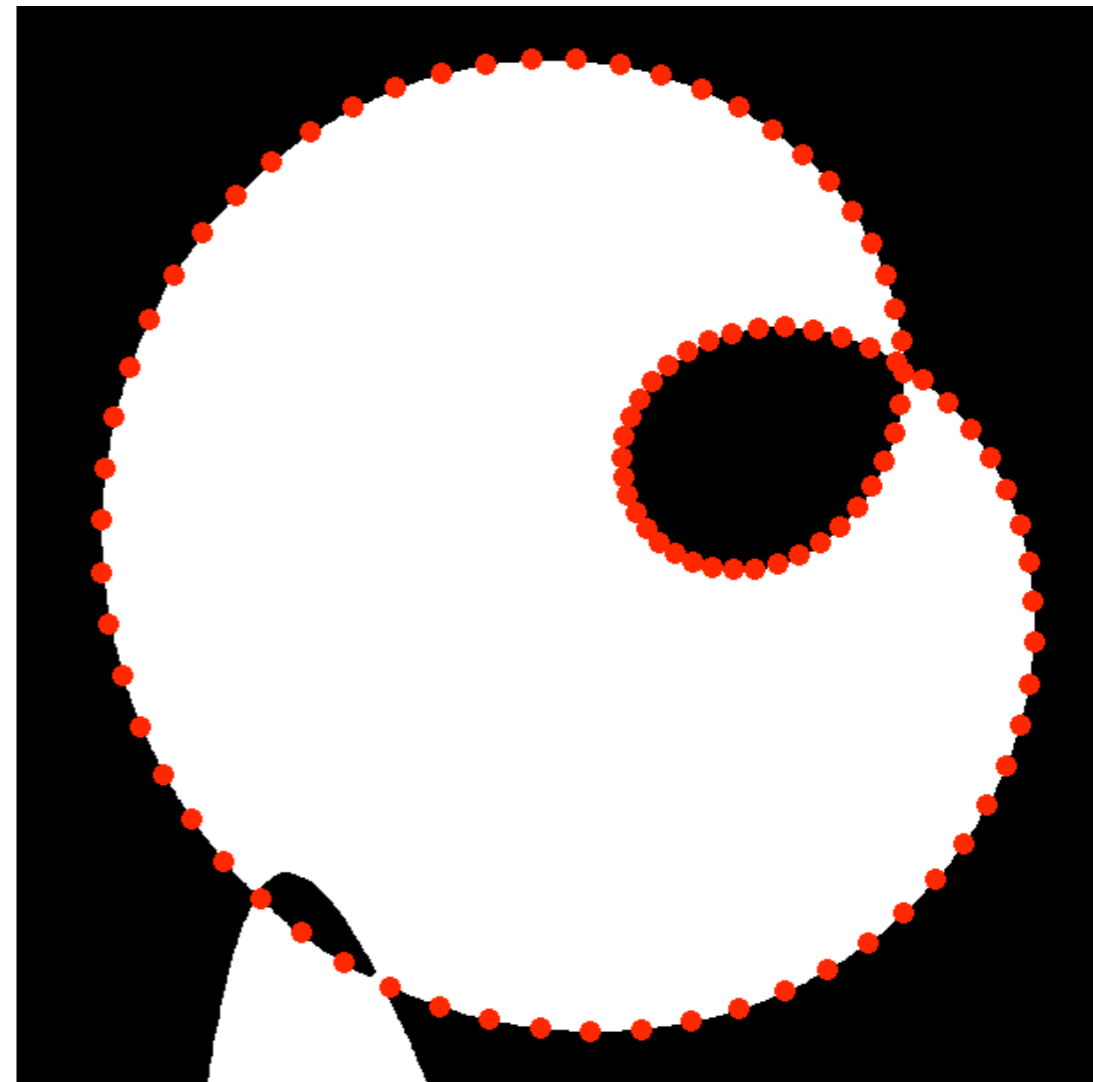
  $|\lambda| \approx 3.1 \cdot 10^{-17}$



degree-assumption too high:

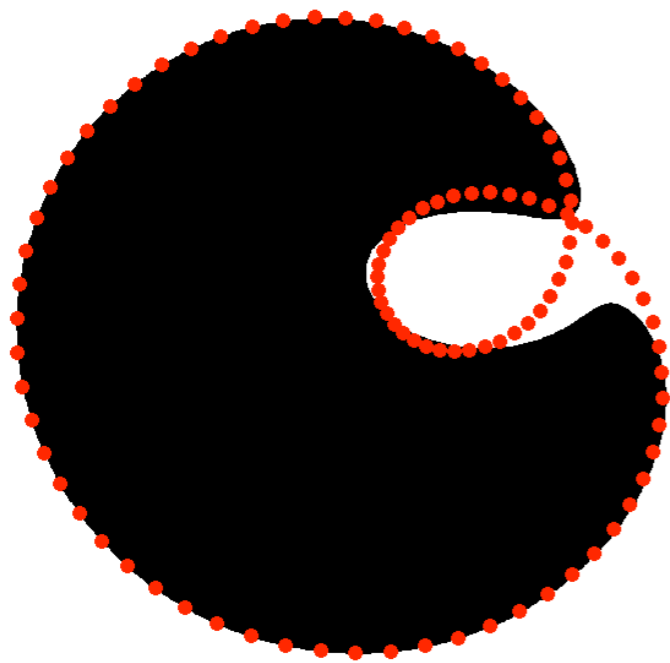


$d = 5$

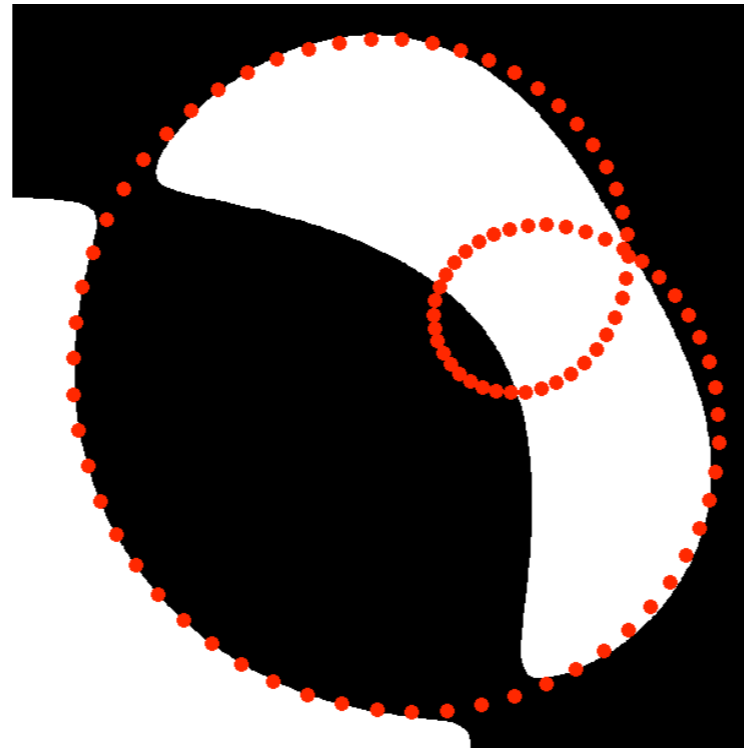


$d = 6$

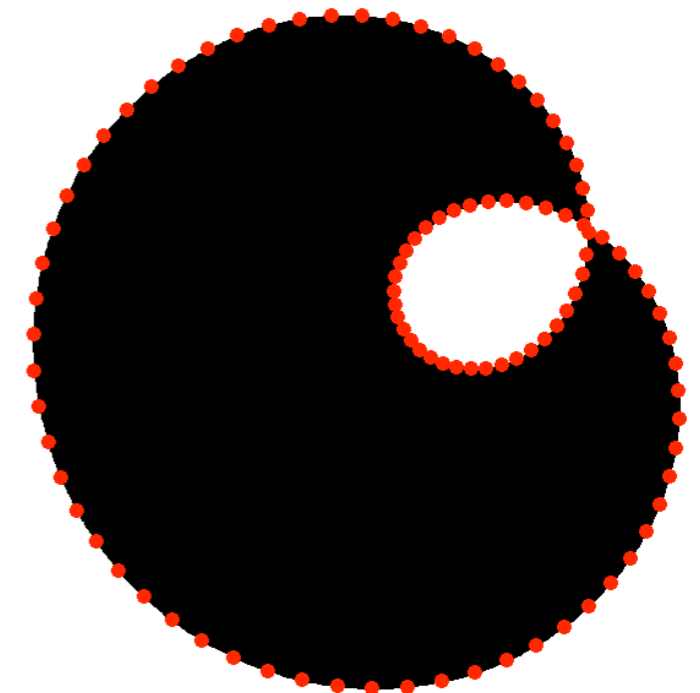
## Implications of matrix notation:



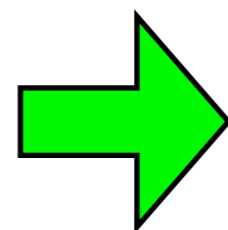
$$p_i \mapsto p_i + (10, 10)$$



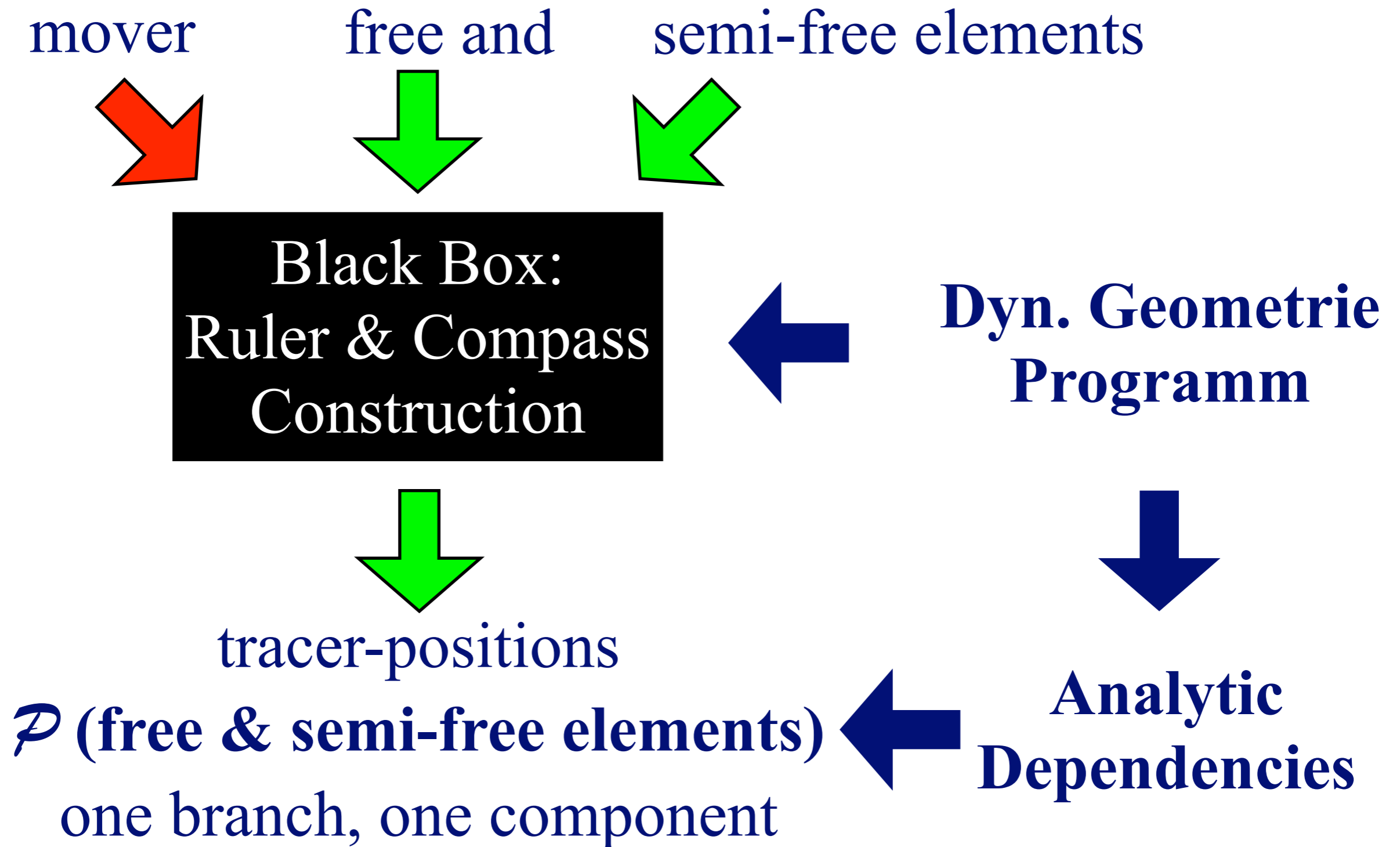
$$p_i \mapsto p_i + (30, 30)$$



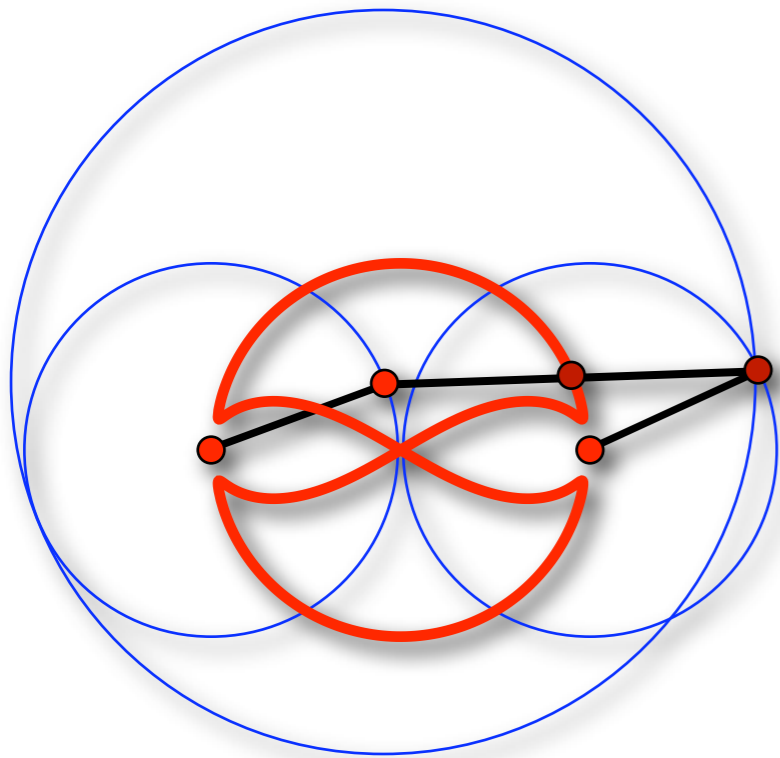
$$p_i \mapsto 0.01 (p_i + (30, 30))$$



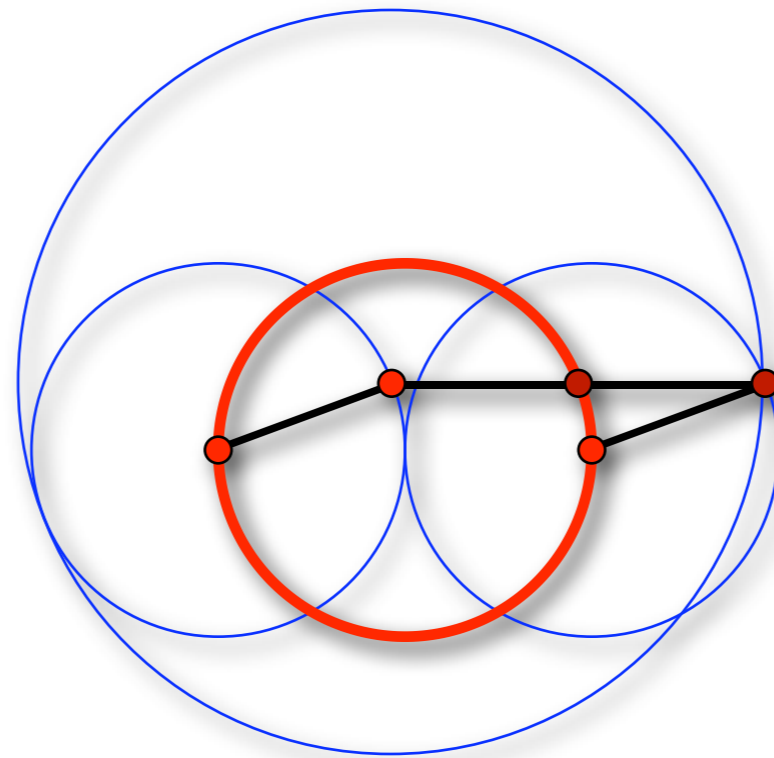
Preprocessing !



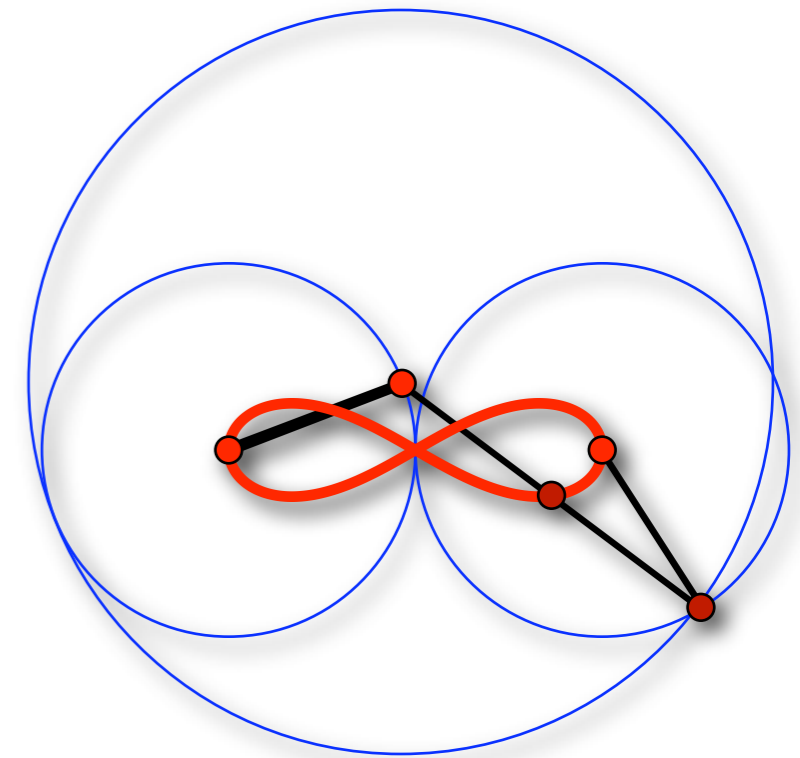
Degenerate curve:  $\mathcal{P}$  is contained in one branch, only



degree 6



degree 2

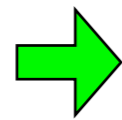
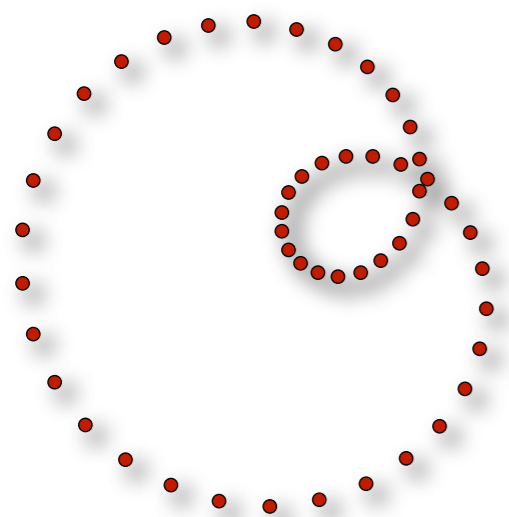


degree 4

Degenerate curve:  $\mathcal{P}$  is contained in one branch, only

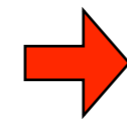
- Set of degenerate curves:  
zero-set / all
- Almost all curves are of the same degree  $d$
- A generic curve yields  $d$

Orthogonalspace:

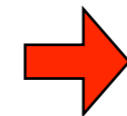


$\beta =$

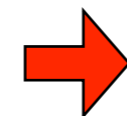
$$\begin{bmatrix} 1 \\ -9.4 \cdot 10^{-8} \\ 2 \\ -8.8 \cdot 10^{-8} \\ 1 \\ -28 \\ -0.56 \\ -28 \\ -0.56 \\ 237 \\ 7.9 \\ 40 \\ -423 \\ 66 \\ -1396 \end{bmatrix}$$



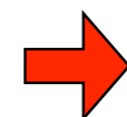
$$2\beta_1 = \beta_3 = 2\beta_5$$



$$\beta_2 = \beta_4 = 0$$



$$\beta_6 = \beta_8$$



$$\beta_7 = \beta_9$$

## Orthogonalspace:

$$\begin{aligned} (2,0,-1,0,0,0,0,0,0,0,0,0,0,0) \\ (0,0,-1,0,2,0,0,0,0,0,0,0,0,0) \end{aligned} \quad \leftarrow \quad 2\beta_1 = \beta_3 = 2\beta_5$$

$$\begin{aligned} (0,1,0,0,0,0,0,0,0,0,0,0,0,0) \\ (0,0,0,1,0,0,0,0,0,0,0,0,0,0) \end{aligned} \quad \leftarrow \quad \beta_2 = \beta_4 = 0$$

$$(0,0,0,0,0,1,0,-1,0,0,0,0,0,0) \quad \leftarrow \quad \beta_6 = \beta_8$$

$$(0,0,0,0,0,0,1,0,-1,0,0,0,0,0) \quad \leftarrow \quad \beta_7 = \beta_9$$

## Orthogonalspace:

- Some generic curve parameters  $\rightarrow B$
- Orthogonalspace =  $\ker(B^T B)$
- Compare with database entries

## Problems:

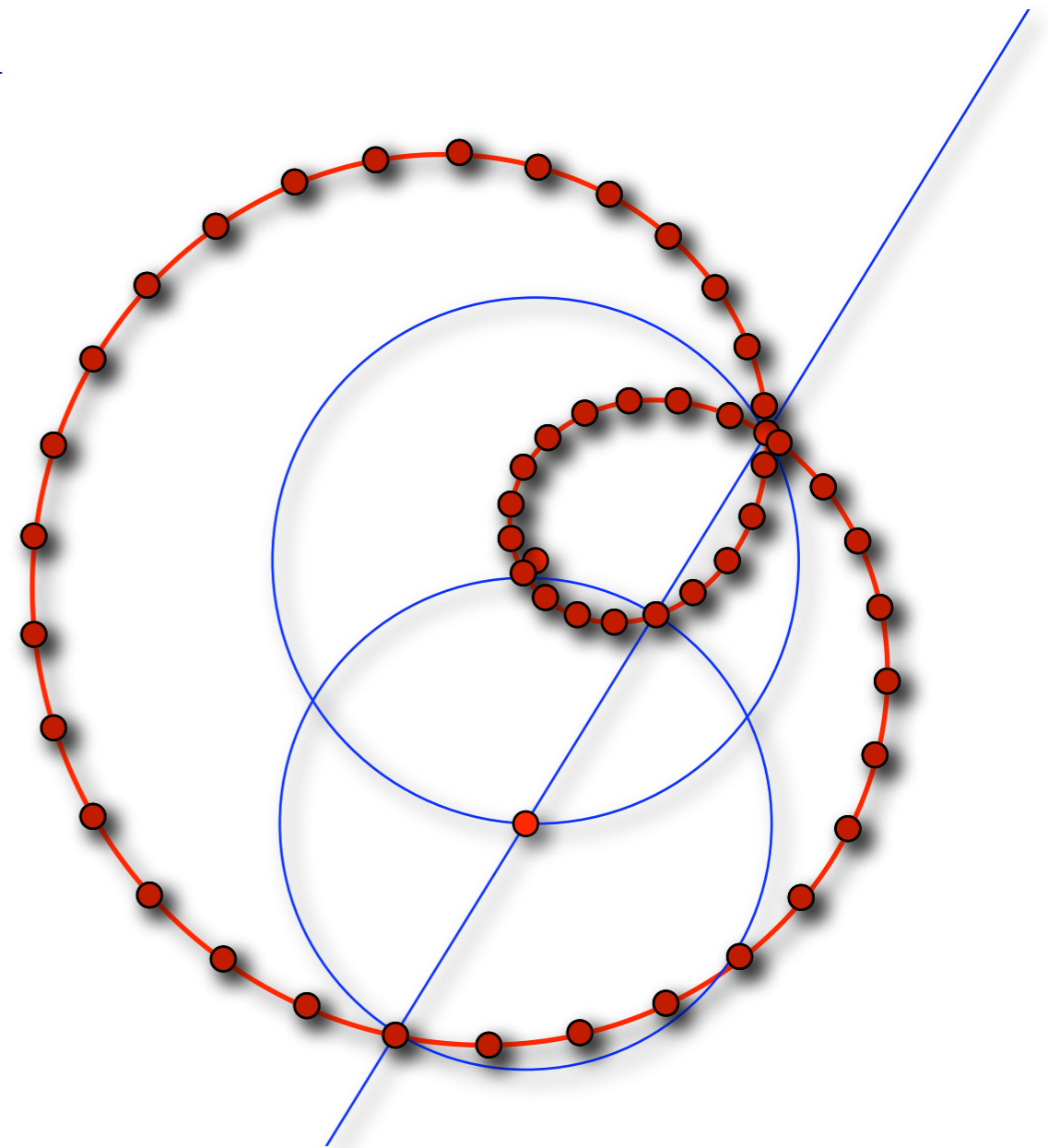
- More complex curves with parameters
- Generic methods

Constructed curves and curve invariants are recognisable !

Still open:

- Optimal preprocessing
- Recognition of more complex curves
- Benefit of randomization techniques

- You constructed a limaçon with parameters  $r = 0,54$  and  $s = 0,72$ .
- A limaçon is given by  $(x^2+y^2-2rx)^2 = s^2(x^2+y^2)$ .
- Your construction will generically generate a limaçon.



To illustrate the meaning of a “construction” and a “locus”, a Cinderella-demonstration was made during the talk. The following slides try to explain the technical terms which were introduced.

(You can download the dynamic geometry software Cinderella from: [www.cinderella.de](http://www.cinderella.de))

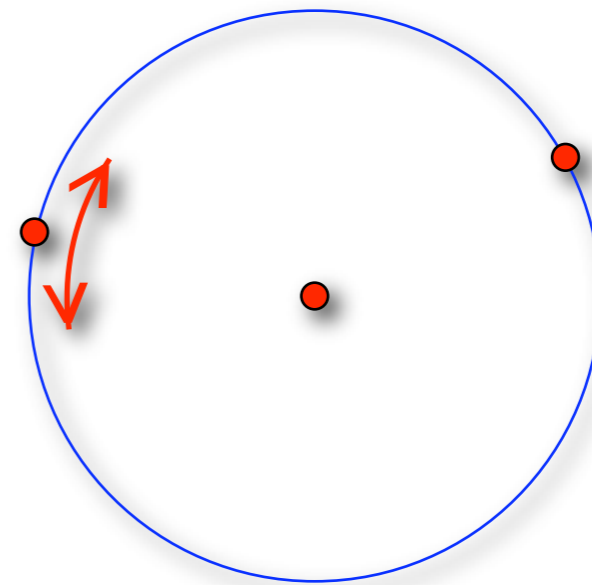
## Construction:

- Free points



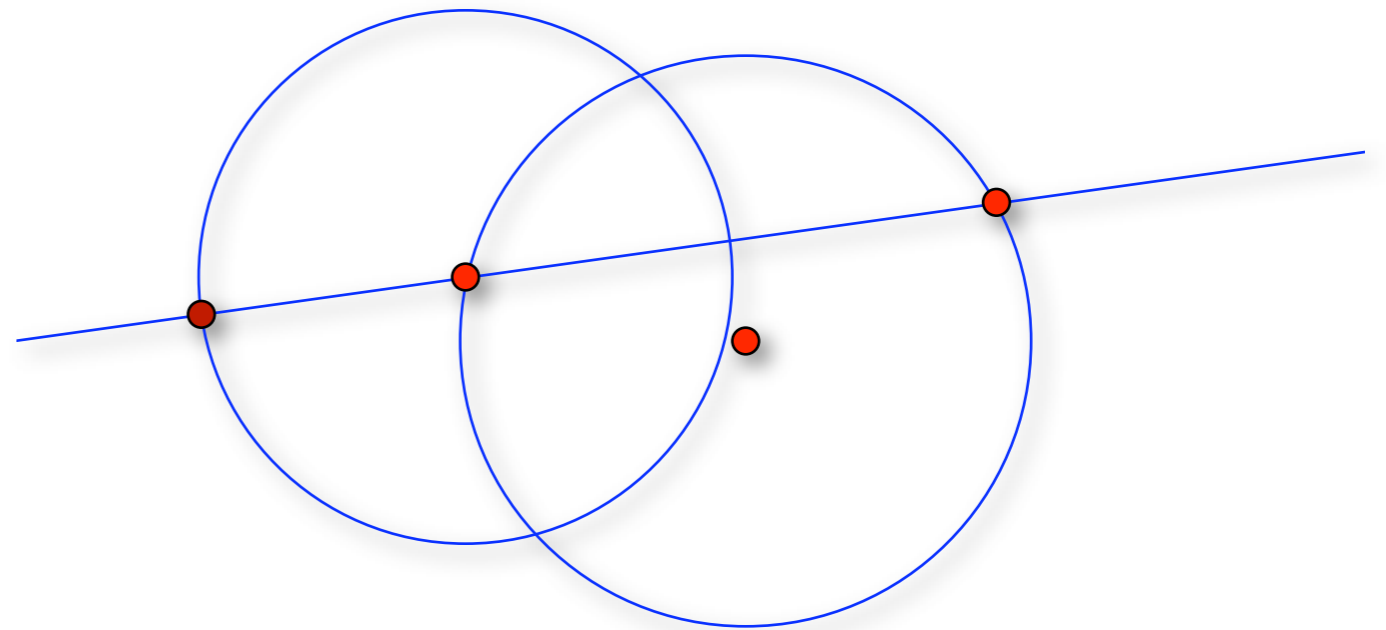
## Construction:

- Free points
- Semi-free points



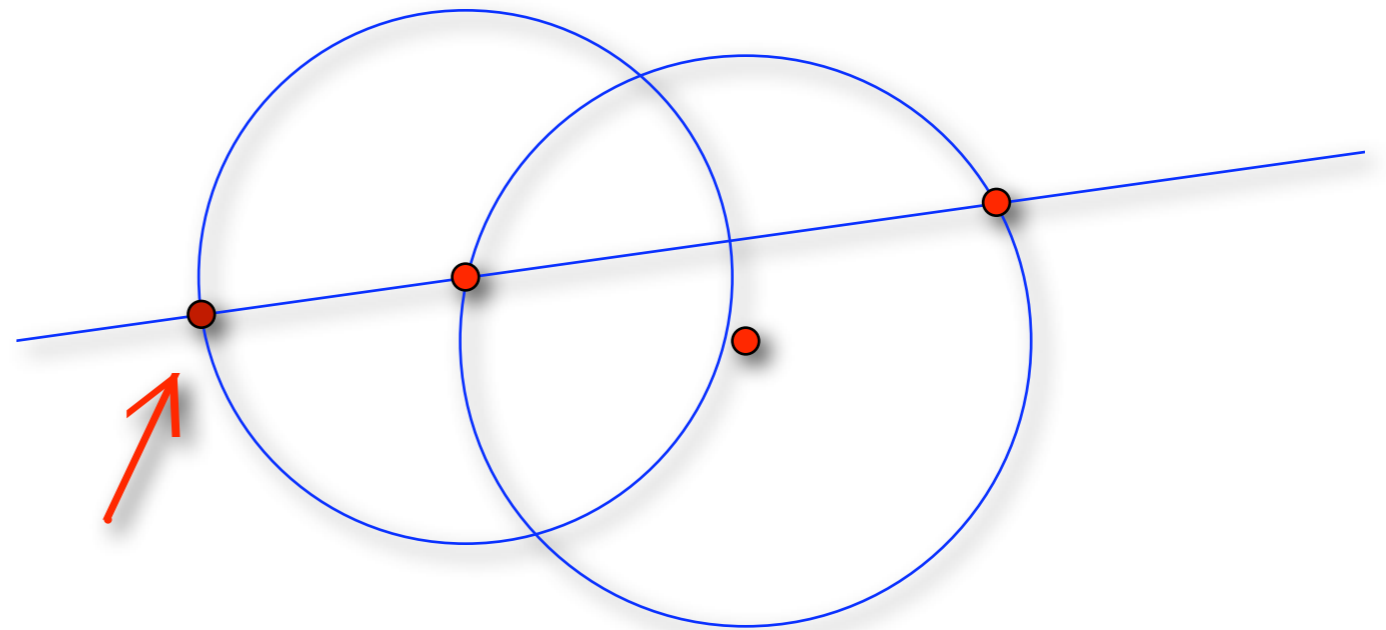
## Construction:

- Free points
- Semi-free points
- Ruler & compass construction



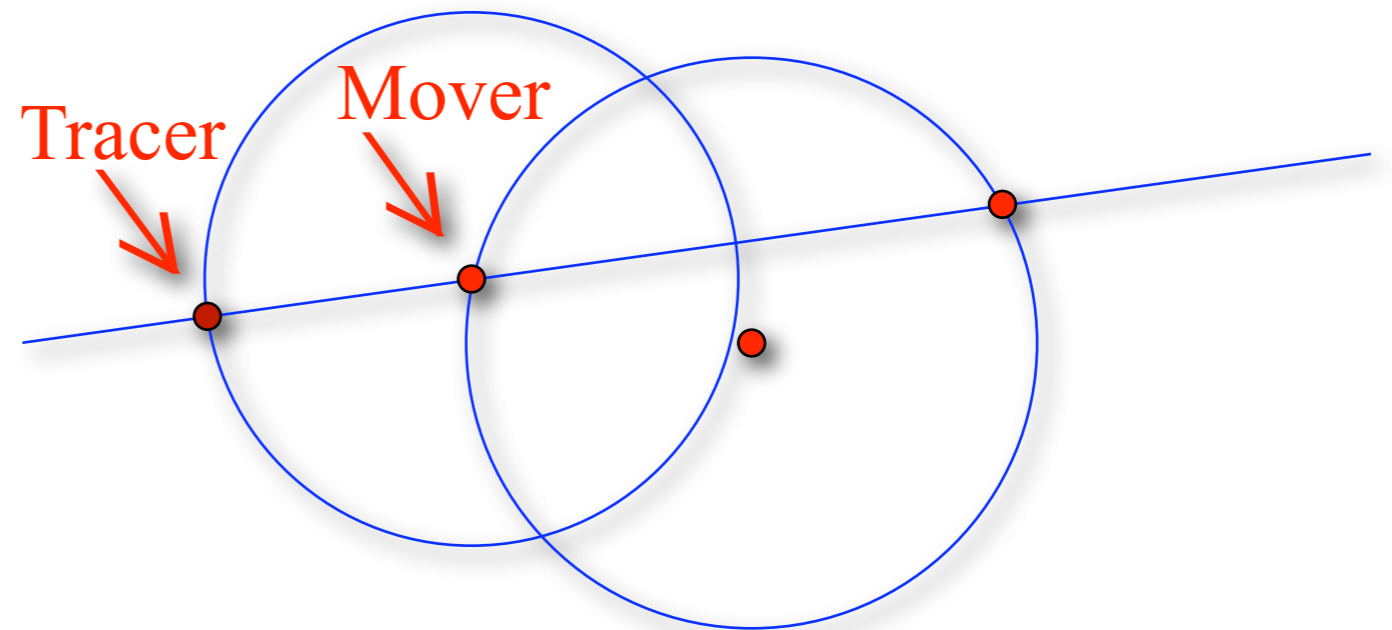
## Construction:

- Free points
- Semi-free points
- Ruler & compass construction
- Dependent points



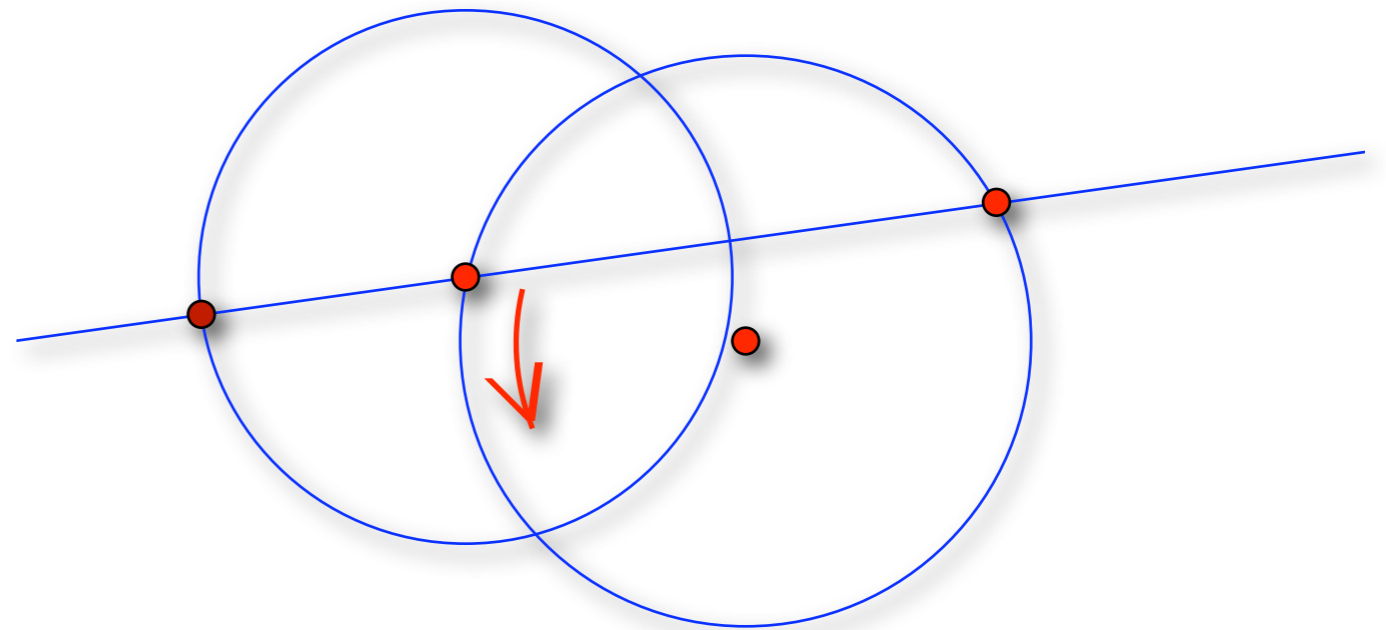
Locus:

- Mover & Tracer



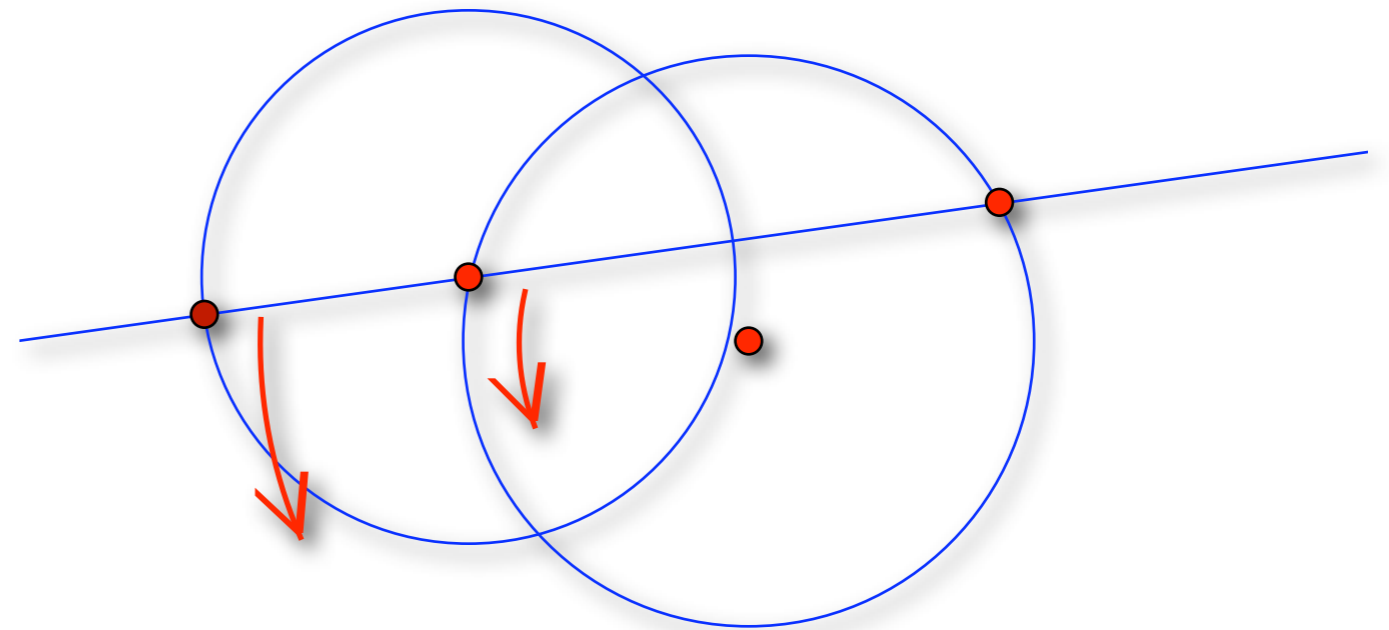
Locus:

- Mover & Tracer



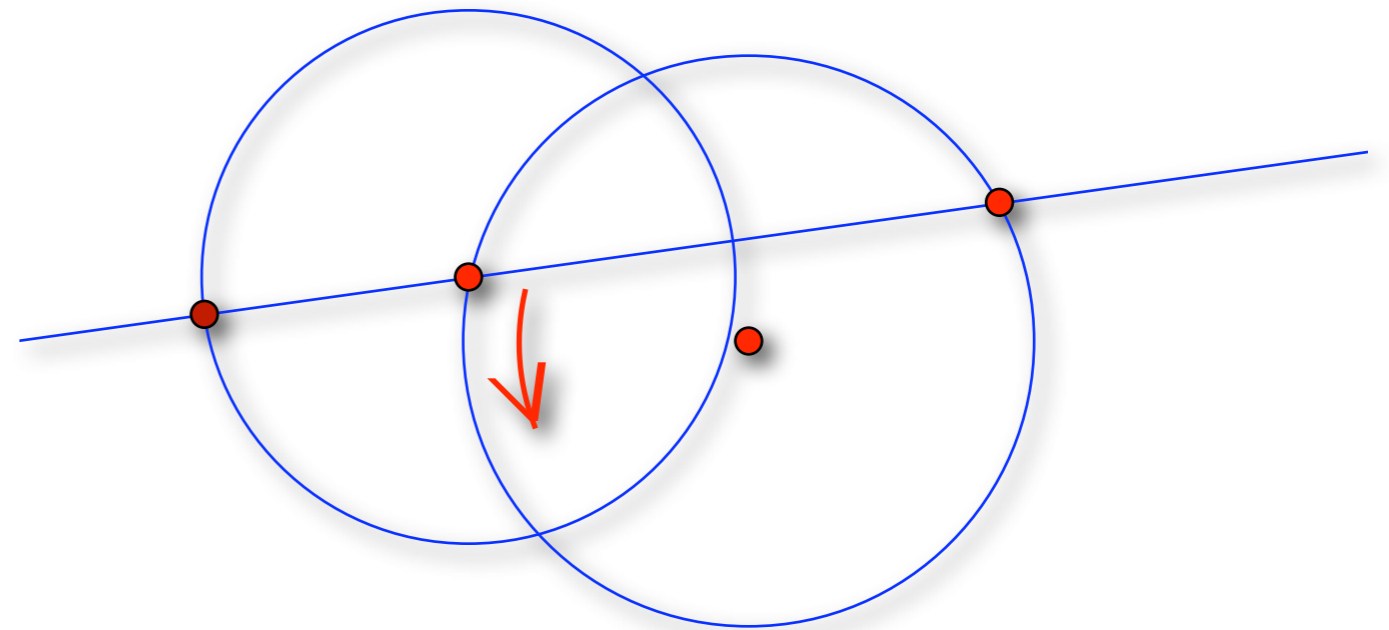
Locus:

- Mover & Tracer



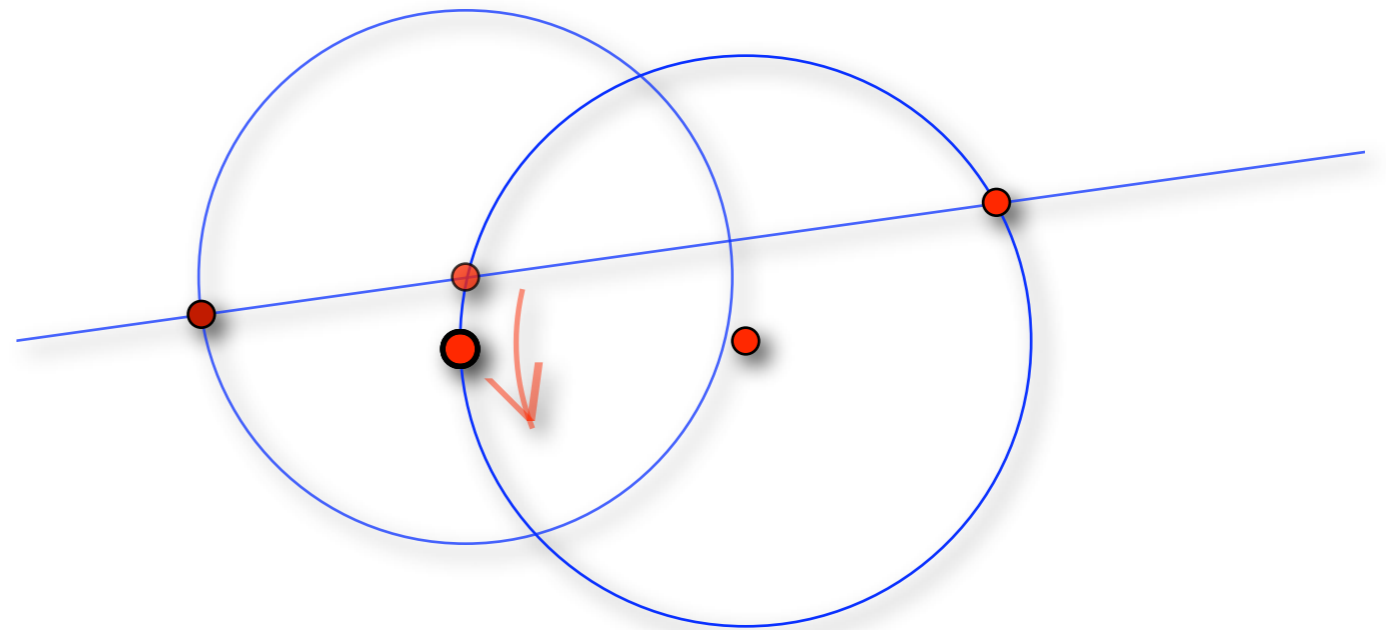
## Locus:

- Mover & Tracer
- Dyn. geometry program selects mover-positions



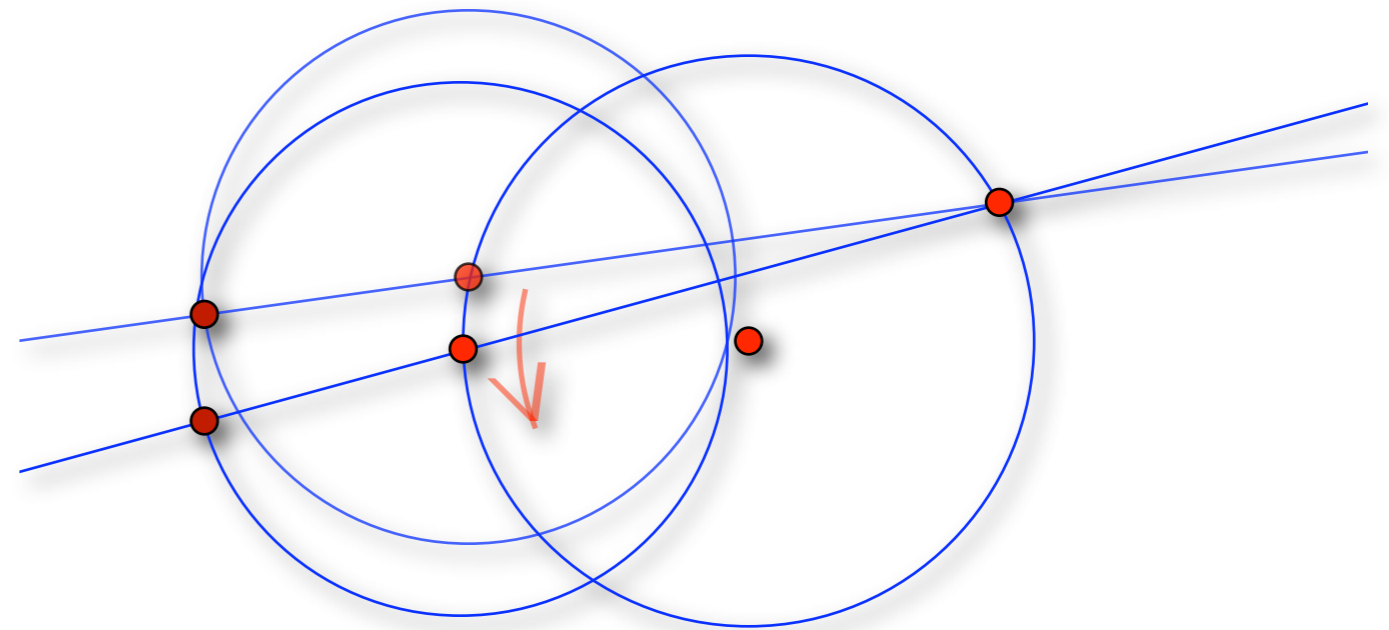
## Locus:

- Mover & Tracer
- Dyn. geometry program selects mover-positions



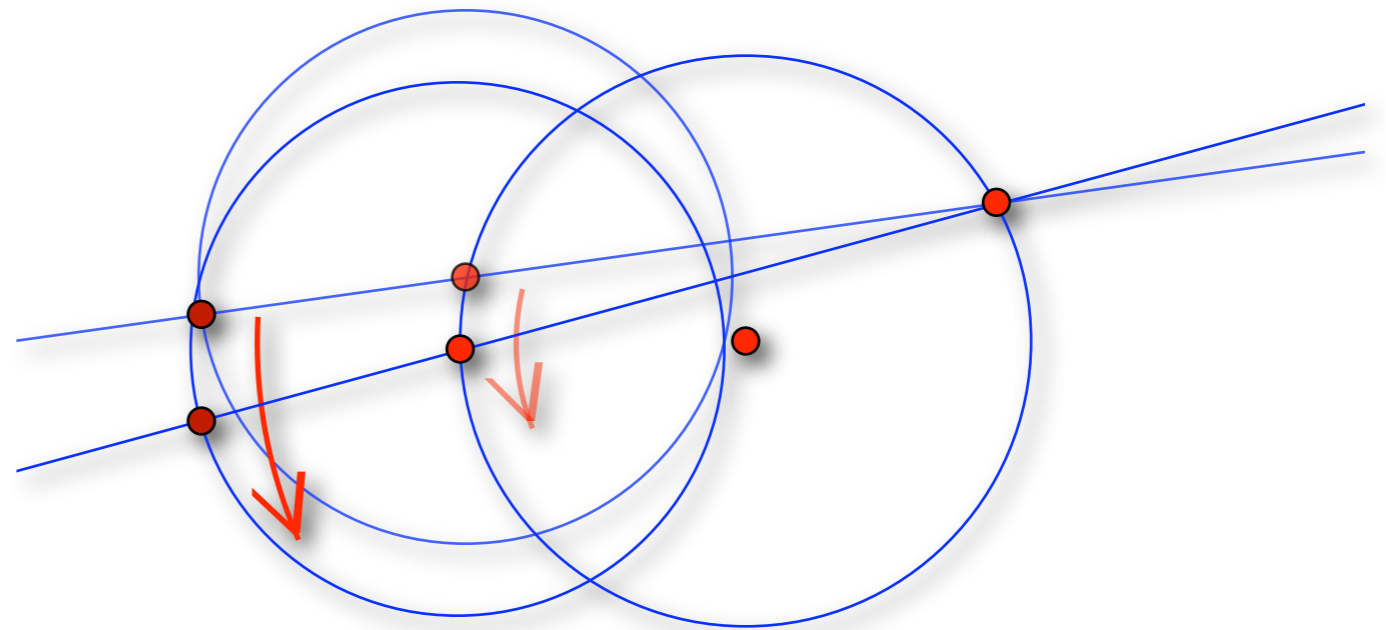
## Locus:

- Mover & Tracer
- Dyn. geometry program selects mover-positions



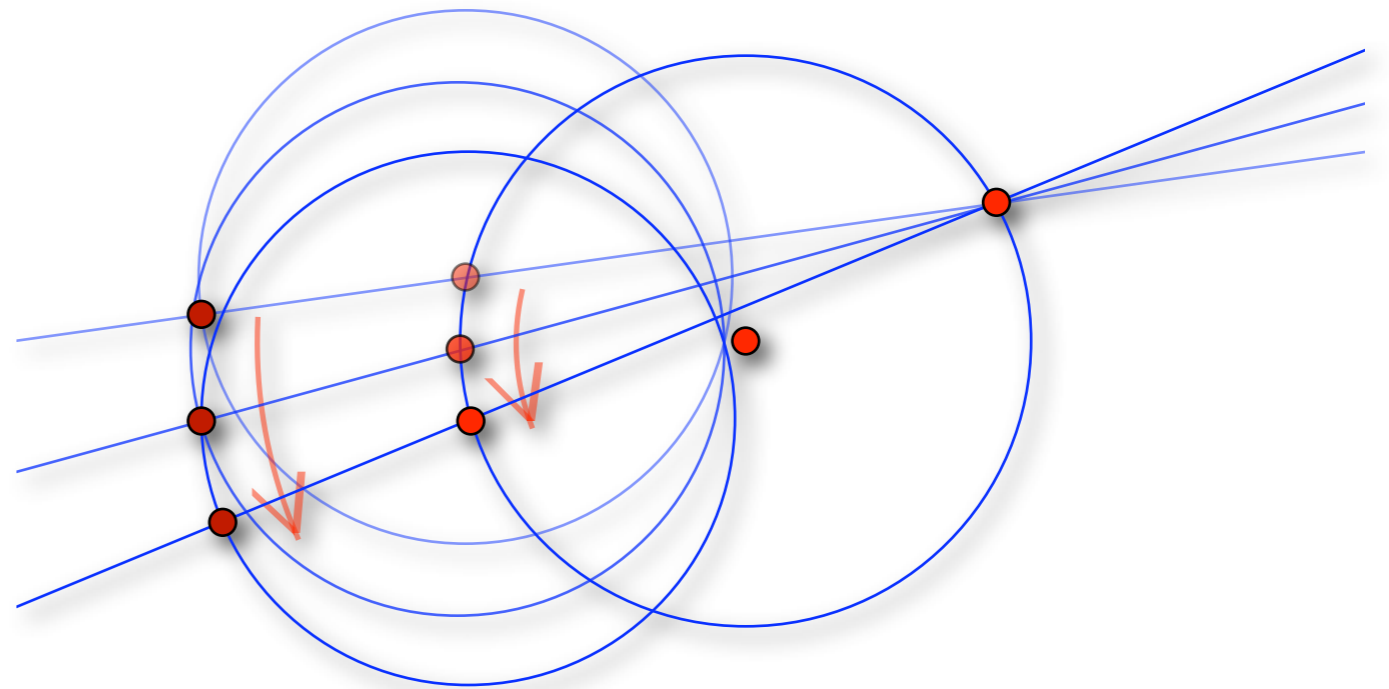
## Locus:

- Mover & Tracer
- Dyn. geometry program selects mover-positions
- precise tracer-positions



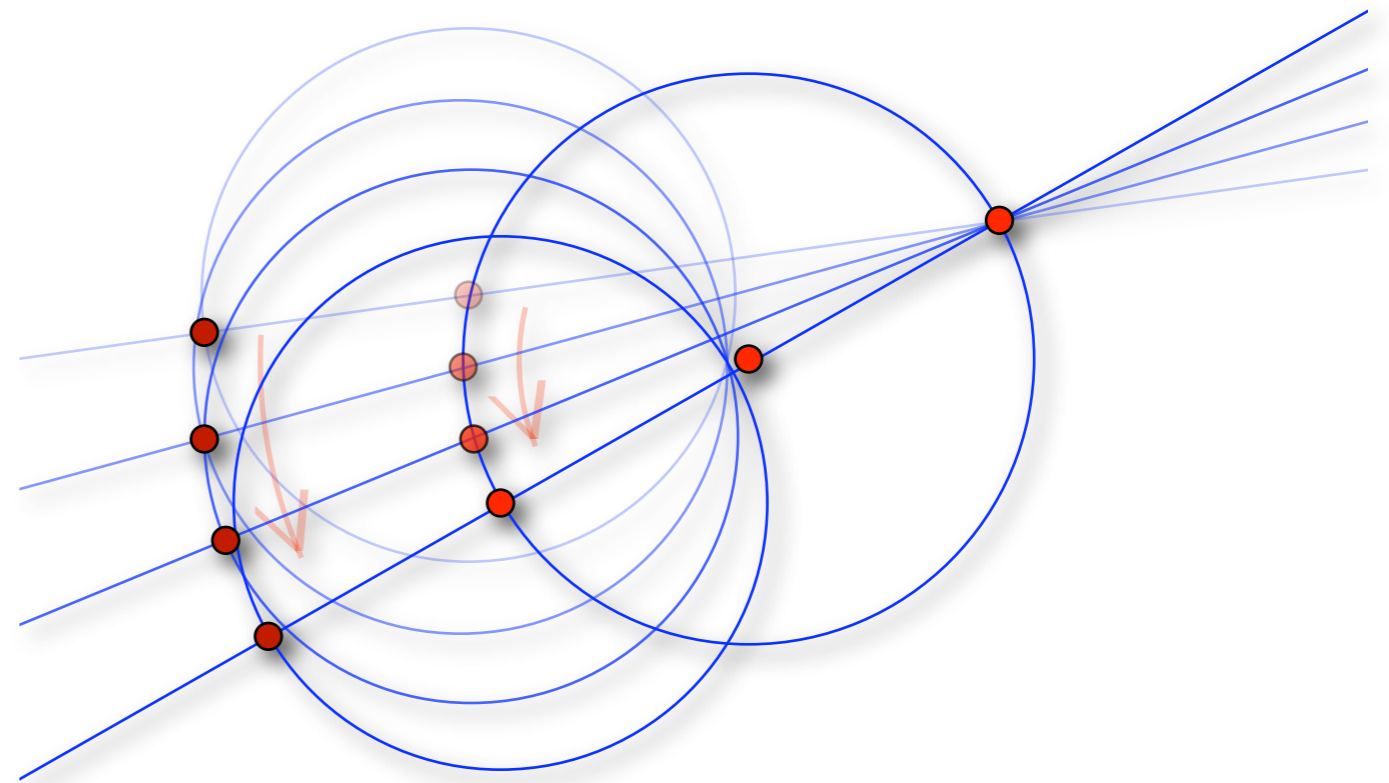
## Locus:

- Mover & Tracer
- Dyn. geometry program selects mover-positions
- precise tracer-positions



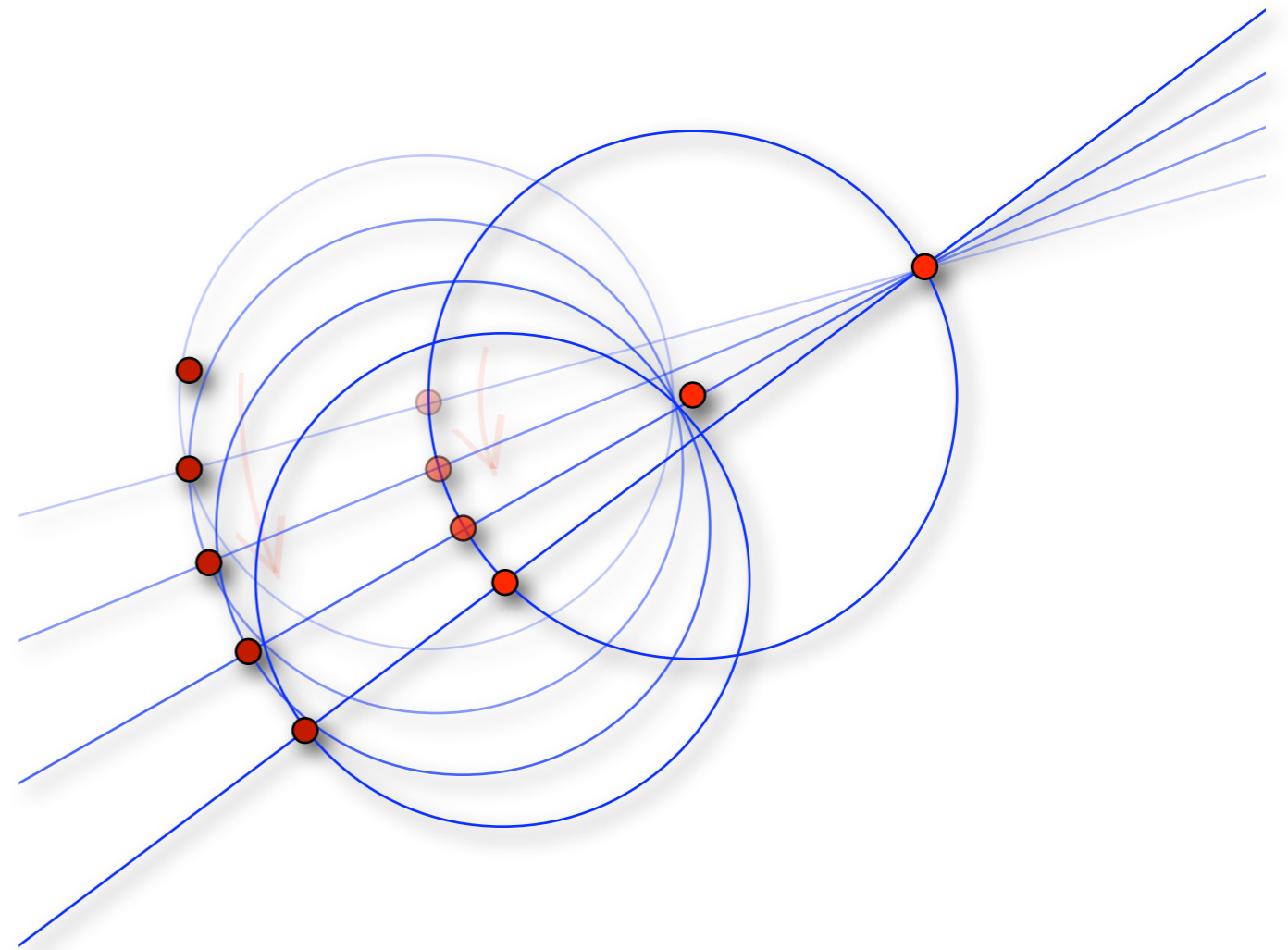
## Locus:

- Mover & Tracer
- Dyn. geometry program selects mover-positions
- precise tracer-positions



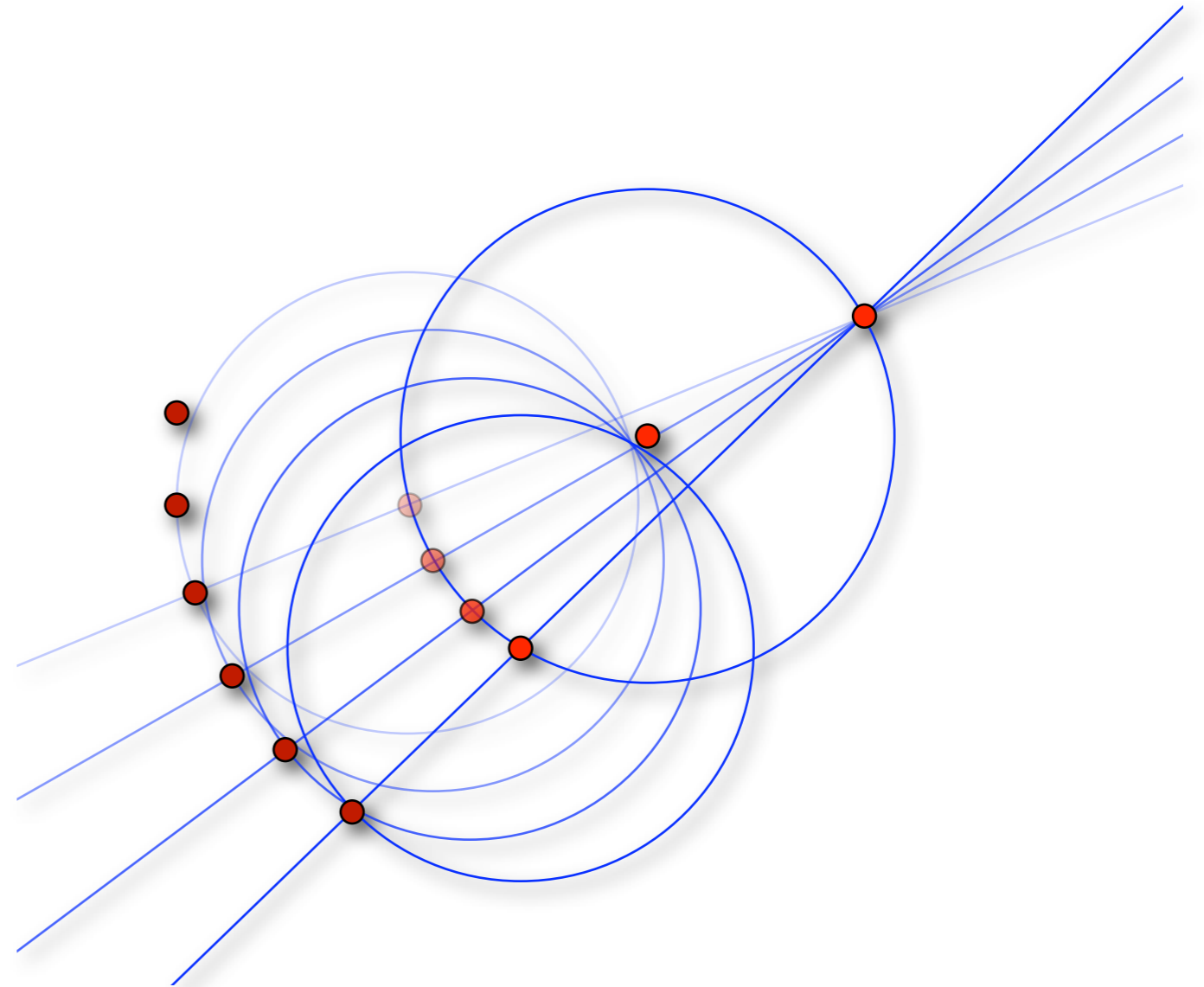
## Locus:

- Mover & Tracer
- Dyn. geometry program selects mover-positions
- precise tracer-positions



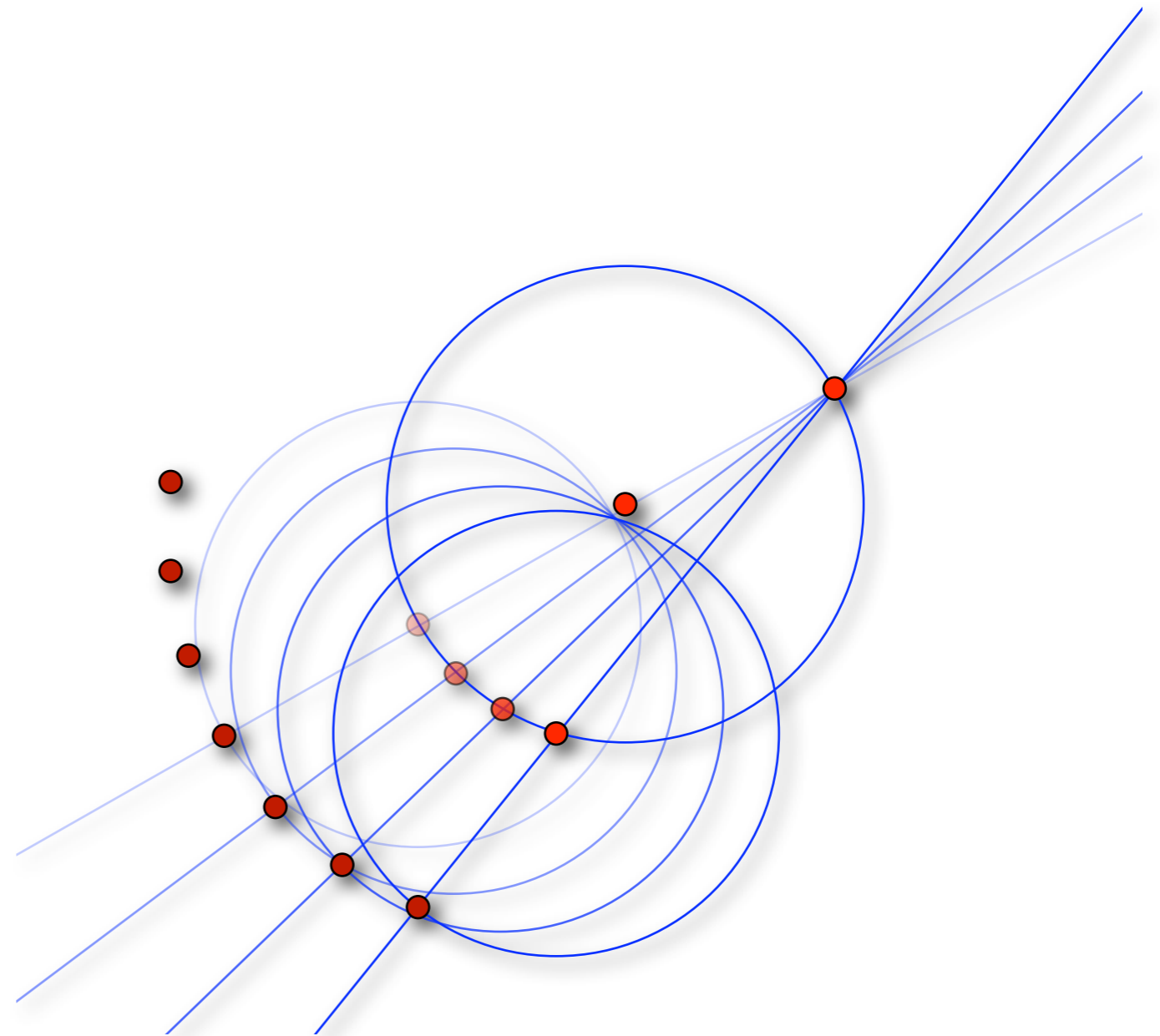
## Locus:

- Mover & Tracer
- Dyn. geometry program selects mover-positions
- precise tracer-positions



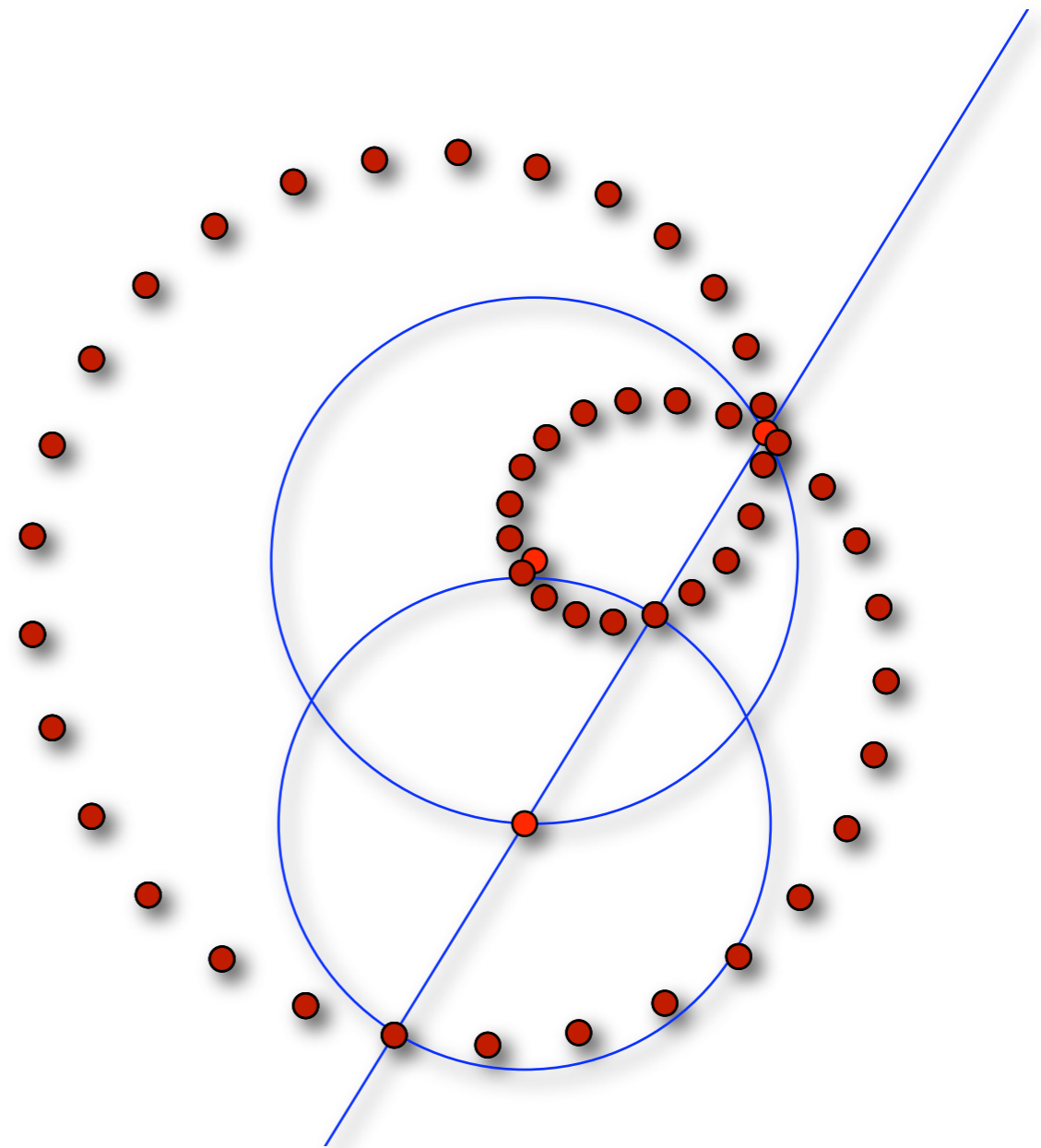
## Locus:

- Mover & Tracer
- Dyn. geometry program selects mover-positions
- precise tracer-positions



## Locus:

- Mover & Tracer
- Dyn. geometry program selects mover-positions
- precise tracer-positions



## Locus:

- Mover & Tracer
- Dyn. geometry program selects mover-positions
- precise tracer-positions  
→ Locus

