

# Zero Problems in Theory and Applications

1

Chee Yap

KIAS, and

Courant Institute of Mathematical Sciences

Department of Computer Science

New York University

# TALK OVERVIEW

2

- Is There a Problem with Zero?
- Why Care About Nothing (Zero)?
- On Zero Bounds and Adaptivity
- On Foundations of Real Computation
- Conclusions

# PART I.

## Is There a Problem with Zero?

*“The history of the **zero recognition** problem is somewhat confused by the fact that many people do not recognize it as a problem at all.”*

— Daniel Richardson (1996)

# What Appears to be the Problem?

- Is a number equal to zero?
  - \* Decision Problem – YES/NO answers
  
- Why is any effort needed at all?
  - \* Numbers have canonical names
  - \* E.g., zero, one, two, half, negative ten, square-root two, pi, etc
  - \* In symbols,  $0$ ,  $1$ ,  $2$ ,  $\frac{1}{2}$ ,  $-10$ ,  $\sqrt{2}$ ,  $\pi$ , ...
  
- Numerical Expressions (non-canonical!)
  - \*  $1 - 1 + 1 - 1$ ,
  - \*  $2^2 + 5 - 3^2$ ,
  - \*  $1 - \sum_{n=1}^{\infty} 2^{-n}$ ,
  - \*  $\sqrt{2} + \sqrt{3} - \sqrt{5 + 2\sqrt{6}}$
  - \* These are all 0

# What Appears to be the Problem?

- Is a number equal to zero?
  - \* Decision Problem – YES/NO answers
  
- Why is any effort needed at all?
  - \* Numbers have canonical names
  - \* E.g., zero, one, two, half, negative ten, square-root two, pi, etc
  - \* In symbols,  $0$ ,  $1$ ,  $2$ ,  $\frac{1}{2}$ ,  $-10$ ,  $\sqrt{2}$ ,  $\pi$ , ...
  
- Numerical Expressions (non-canonical!)
  - \*  $1 - 1 + 1 - 1$ ,
  - \*  $2^2 + 5 - 3^2$ ,
  - \*  $1 - \sum_{n=1}^{\infty} 2^{-n}$ ,
  - \*  $\sqrt{2} + \sqrt{3} - \sqrt{5 + 2\sqrt{6}}$
  - \* These are all 0

# What Appears to be the Problem?

- Is a number equal to zero?
  - \* Decision Problem – YES/NO answers
  
- Why is any effort needed at all?
  - \* Numbers have canonical names
  - \* E.g., zero, one, two, half, negative ten, square-root two, pi, etc
  - \* In symbols,  $0$ ,  $1$ ,  $2$ ,  $\frac{1}{2}$ ,  $-10$ ,  $\sqrt{2}$ ,  $\pi$ , ...
  
- Numerical Expressions (non-canonical!)
  - \*  $1 - 1 + 1 - 1$ ,
  - \*  $2^2 + 5 - 3^2$ ,
  - \*  $1 - \sum_{n=1}^{\infty} 2^{-n}$ ,
  - \*  $\sqrt{2} + \sqrt{3} - \sqrt{5 + 2\sqrt{6}}$
  - \* These are all 0

# What Appears to be the Problem?

- Is a number equal to zero?
  - \* Decision Problem – YES/NO answers
  
- Why is any effort needed at all?
  - \* Numbers have canonical names
  - \* E.g., zero, one, two, half, negative ten, square-root two, pi, etc
  - \* In symbols,  $0, 1, 2, \frac{1}{2}, -10, \sqrt{2}, \pi, \dots$
  
- Numerical Expressions (non-canonical!)
  - \*  $1 - 1 + 1 - 1,$
  - \*  $2^2 + 5 - 3^2,$
  - \*  $1 - \sum_{n=1}^{\infty} 2^{-n},$
  - \*  $\sqrt{2} + \sqrt{3} - \sqrt{5 + 2\sqrt{6}}$
  - \* These are all 0

## But is it REALLY zero?

- Two ways to decide zero:
  - \* (A) Algebraically. E.g., repeated squaring
  - \* (B) Numerically. E.g., by approximation

- We are interested in (B):

\*

$$\begin{aligned}
 \sqrt{2} + \sqrt{3} - \sqrt{5 + 2\sqrt{6}} &= 1.4142 + 1.7320 - \sqrt{5 + 2 \times 2.4494} \\
 &= 3.1462 - \sqrt{9.8989} \\
 &= 3.1462 - 3.1462 \\
 &= 0 \quad ??
 \end{aligned}$$



## But is it REALLY zero?

- Two ways to decide zero:
  - \* (A) Algebraically. E.g., repeated squaring
  - \* (B) Numerically. E.g., by approximation
- We are interested in (B):

\*

$$\begin{aligned}
 \sqrt{2} + \sqrt{3} - \sqrt{5 + 2\sqrt{6}} &= 1.4142 + 1.7320 - \sqrt{5 + 2 \times 2.4494} \\
 &= 3.1462 - \sqrt{9.8989} \\
 &= 3.1462 - 3.1462 \\
 &= 0 \quad ??
 \end{aligned}$$

## But is it REALLY zero?

- Two ways to decide zero:
  - \* (A) Algebraically. E.g., repeated squaring
  - \* (B) Numerically. E.g., by approximation
- We are interested in (B):

\*

$$\begin{aligned}
 \sqrt{2} + \sqrt{3} - \sqrt{5 + 2\sqrt{6}} &= 1.4142 + 1.7320 - \sqrt{5 + 2 \times 2.4494} \\
 &= 3.1462 - \sqrt{9.8989} \\
 &= 3.1462 - 3.1462 \\
 &= 0 \quad ??
 \end{aligned}$$

# Apparent Zeros

- Folklore:

- \*  $\left(3 \cdot \log(640320) / \sqrt{163}\right) - \pi < 10^{-15}$

- R.Graham:

- \*

$$\begin{aligned} & \left( \sqrt{1000001} + \sqrt{1000025} + \sqrt{1000031} + \sqrt{1000084} + \sqrt{1000087} \right. \\ & \quad \left. + \sqrt{1000134} + \sqrt{1000158} + \sqrt{1000182} + \sqrt{1000198} \right) \\ & - \left( \sqrt{1000002} + \sqrt{1000018} + \sqrt{1000042} + \sqrt{1000066} + \sqrt{1000113} \right. \\ & \quad \left. + \sqrt{1000116} + \sqrt{1000169} + \sqrt{1000175} + \sqrt{1000199} \right) \\ & < 10^{-36} \end{aligned}$$

- Many more...

- Richardson (2005)

- \* Let  $F(x) = (1 + x)^{1/2} - 2(1 + 3x/4)^{1/3} + 1$

- \* Then  $F(F(10^{-126})) < 10^{-1141}$

# Apparent Zeros

- Folklore:

$$* \left( 3 \cdot \log(640320) / \sqrt{163} \right) - \pi < 10^{-15}$$

- R.Graham:

\*

$$\begin{aligned} & \left( \sqrt{1000001} + \sqrt{1000025} + \sqrt{1000031} + \sqrt{1000084} + \sqrt{1000087} \right. \\ & \quad \left. + \sqrt{1000134} + \sqrt{1000158} + \sqrt{1000182} + \sqrt{1000198} \right) \\ & - \left( \sqrt{1000002} + \sqrt{1000018} + \sqrt{1000042} + \sqrt{1000066} + \sqrt{1000113} \right. \\ & \quad \left. + \sqrt{1000116} + \sqrt{1000169} + \sqrt{1000175} + \sqrt{1000199} \right) \\ & < 10^{-36} \end{aligned}$$

- Many more...

- Richardson (2005)

$$* \text{ Let } F(x) = (1 + x)^{1/2} - 2(1 + 3x/4)^{1/3} + 1$$

$$* \text{ Then } F(F(10^{-126})) < 10^{-1141}$$

# Apparent Zeros

- Folklore:

$$* \left( 3 \cdot \log(640320) / \sqrt{163} \right) - \pi < 10^{-15}$$

- R.Graham:

\*

$$\begin{aligned} & \left( \sqrt{1000001} + \sqrt{1000025} + \sqrt{1000031} + \sqrt{1000084} + \sqrt{1000087} \right. \\ & \quad \left. + \sqrt{1000134} + \sqrt{1000158} + \sqrt{1000182} + \sqrt{1000198} \right) \\ & - \left( \sqrt{1000002} + \sqrt{1000018} + \sqrt{1000042} + \sqrt{1000066} + \sqrt{1000113} \right. \\ & \quad \left. + \sqrt{1000116} + \sqrt{1000169} + \sqrt{1000175} + \sqrt{1000199} \right) \\ & < 10^{-36} \end{aligned}$$

- Many more...

- Richardson (2005)

$$* \text{ Let } F(x) = (1 + x)^{1/2} - 2(1 + 3x/4)^{1/3} + 1$$

$$* \text{ Then } F(F(10^{-126})) < 10^{-1141}$$

# Apparent Zeros

- Folklore:

$$* \left( 3 \cdot \log(640320) / \sqrt{163} \right) - \pi < 10^{-15}$$

- R.Graham:

\*

$$\begin{aligned} & \left( \sqrt{1000001} + \sqrt{1000025} + \sqrt{1000031} + \sqrt{1000084} + \sqrt{1000087} \right. \\ & \quad \left. + \sqrt{1000134} + \sqrt{1000158} + \sqrt{1000182} + \sqrt{1000198} \right) \\ & - \left( \sqrt{1000002} + \sqrt{1000018} + \sqrt{1000042} + \sqrt{1000066} + \sqrt{1000113} \right. \\ & \quad \left. + \sqrt{1000116} + \sqrt{1000169} + \sqrt{1000175} + \sqrt{1000199} \right) \\ & < 10^{-36} \end{aligned}$$

- Many more...

- Richardson (2005)

$$* \text{ Let } F(x) = (1 + x)^{1/2} - 2(1 + 3x/4)^{1/3} + 1$$

$$* \text{ Then } F(F(10^{-126})) < 10^{-1141}$$

# Apparent Zeros

- Folklore:

$$* \left( 3 \cdot \log(640320) / \sqrt{163} \right) - \pi < 10^{-15}$$

- R.Graham:

\*

$$\begin{aligned} & \left( \sqrt{1000001} + \sqrt{1000025} + \sqrt{1000031} + \sqrt{1000084} + \sqrt{1000087} \right. \\ & \quad \left. + \sqrt{1000134} + \sqrt{1000158} + \sqrt{1000182} + \sqrt{1000198} \right) \\ & - \left( \sqrt{1000002} + \sqrt{1000018} + \sqrt{1000042} + \sqrt{1000066} + \sqrt{1000113} \right. \\ & \quad \left. + \sqrt{1000116} + \sqrt{1000169} + \sqrt{1000175} + \sqrt{1000199} \right) \\ & < 10^{-36} \end{aligned}$$

- Many more...

- Richardson (2005)

$$* \text{ Let } F(x) = (1 + x)^{1/2} - 2(1 + 3x/4)^{1/3} + 1$$

$$* \text{ Then } F(F(10^{-126})) < 10^{-1141}$$

# The Zero Problems

- $\Omega$  = set of algebraic operators
  - \* E.g.,  $\Omega = \{+, \times, \div\} \cup \mathbb{Z}$
- Let  $\text{Expr}(\Omega)$  denote expressions over  $\Omega$
- Evaluation function:  
 $\text{Val} : \text{Expr}(\Omega) \dashrightarrow \mathbb{C}$ 
  - \* Say  $e$  is invalid if  $\text{Val}(e) = \uparrow$
- Zero problem,  $ZERO(\Omega)$ :
  - \* Given  $e \in \text{Expr}(\Omega)$ , is  $\text{Val}(e) = 0$  ?



# The Zero Problems

- $\Omega$  = set of algebraic operators
  - \* E.g.,  $\Omega = \{\pm, \times, \div\} \cup \mathbb{Z}$
- Let  $\text{Expr}(\Omega)$  denote **expressions** over  $\Omega$
- Evaluation function:  
 $\text{Val} : \text{Expr}(\Omega) \dashrightarrow \mathbb{C}$ 
  - \* Say  $e$  is invalid if  $\text{Val}(e) = \uparrow$
- Zero problem,  $ZERO(\Omega)$ :
  - \* Given  $e \in \text{Expr}(\Omega)$ , is  $\text{Val}(e) = 0$  ?

# The Zero Problems

- $\Omega$  = set of algebraic operators
  - \* E.g.,  $\Omega = \{\pm, \times, \div\} \cup \mathbb{Z}$
- Let  $\text{Expr}(\Omega)$  denote expressions over  $\Omega$
- Evaluation function:  
 $\text{Val} : \text{Expr}(\Omega) \dashrightarrow \mathbb{C}$ 
  - \* Say  $e$  is invalid if  $\text{Val}(e) = \uparrow$
- Zero problem,  $ZERO(\Omega)$ :
  - \* Given  $e \in \text{Expr}(\Omega)$ , is  $\text{Val}(e) = 0$  ?

# The Zero Problems

- $\Omega =$  set of algebraic operators
  - \* E.g.,  $\Omega = \{\pm, \times, \div\} \cup \mathbb{Z}$
- Let  $\text{Expr}(\Omega)$  denote expressions over  $\Omega$
- Evaluation function:  
 $\text{Val} : \text{Expr}(\Omega) \dashrightarrow \mathbb{C}$ 
  - \* Say  $e$  is invalid if  $\text{Val}(e) = \uparrow$
- Zero problem,  $ZERO(\Omega)$ :
  - \* Given  $e \in \text{Expr}(\Omega)$ , is  $\text{Val}(e) = 0$  ?

# The Zero Problems

- $\Omega$  = set of algebraic operators
  - \* E.g.,  $\Omega = \{\pm, \times, \div\} \cup \mathbb{Z}$
- Let  $\text{Expr}(\Omega)$  denote expressions over  $\Omega$
- Evaluation function:  
 $\text{Val} : \text{Expr}(\Omega) \dashrightarrow \mathbb{C}$ 
  - \* Say  $e$  is invalid if  $\text{Val}(e) = \uparrow$
- Zero problem,  $ZERO(\Omega)$ :
  - \* Given  $e \in \text{Expr}(\Omega)$ , is  $\text{Val}(e) = 0$  ?

# Error Notation

- NOTATION:  $\tilde{x} = x \pm h$ 
  - \* means  $|\tilde{x} - x| \leq h$
- Absolute Error
  - \*  $\tilde{x}$  is a absolute  $p$ -bit approximation of  $x$  if  $\tilde{x} = x \pm 2^{-p}$
- Relative Error
  - \*  $\tilde{x}$  is a relative  $p$ -bit approximation of  $x$  if  $\tilde{x} = x(1 \pm 2^{-p}|x|)$
- Precision vs. Error
  - \* (1) “precision” is a priori error
  - \* (2) “error” is a posteriori error

# Error Notation

- NOTATION:  $\tilde{x} = x \pm h$ 
  - \* means  $|\tilde{x} - x| \leq h$
- Absolute Error
  - \*  $\tilde{x}$  is a absolute  $p$ -bit approximation of  $x$  if  $\tilde{x} = x \pm 2^{-p}$
- Relative Error
  - \*  $\tilde{x}$  is a relative  $p$ -bit approximation of  $x$  if  $\tilde{x} = x(1 \pm 2^{-p}|x|)$
- Precision vs. Error
  - \* (1) “precision” is a priori error
  - \* (2) “error” is a posteriori error

# Error Notation

- NOTATION:  $\tilde{x} = x \pm h$ 
  - \* means  $|\tilde{x} - x| \leq h$
- Absolute Error
  - \*  $\tilde{x}$  is a absolute  $p$ -bit approximation of  $x$  if  $\tilde{x} = x \pm 2^{-p}$
- Relative Error
  - \*  $\tilde{x}$  is a relative  $p$ -bit approximation of  $x$  if  $\tilde{x} = x(1 \pm 2^{-p}|x|)$
- Precision vs. Error
  - \* (1) “precision” is a priori error
  - \* (2) “error” is a posteriori error

# Error Notation

- NOTATION:  $\tilde{x} = x \pm h$ 
  - \* means  $|\tilde{x} - x| \leq h$
- Absolute Error
  - \*  $\tilde{x}$  is a absolute  $p$ -bit approximation of  $x$  if  $\tilde{x} = x \pm 2^{-p}$
- Relative Error
  - \*  $\tilde{x}$  is a relative  $p$ -bit approximation of  $x$  if  $\tilde{x} = x(1 \pm 2^{-p}|x|)$
- Precision vs. Error
  - \* (1) “precision” is a priori error
  - \* (2) “error” is a posteriori error



# Error Notation

- NOTATION:  $\tilde{x} = x \pm h$ 
  - \* means  $|\tilde{x} - x| \leq h$
- Absolute Error
  - \*  $\tilde{x}$  is a absolute  $p$ -bit approximation of  $x$  if  $\tilde{x} = x \pm 2^{-p}$
- Relative Error
  - \*  $\tilde{x}$  is a relative  $p$ -bit approximation of  $x$  if  $\tilde{x} = x(1 \pm 2^{-p}|x|)$
- Precision vs. Error
  - \* (1) “precision” is a priori error
  - \* (2) “error” is a posteriori error

# Partial Functions

- Partial function,  $f : S \dashrightarrow T$ 
  - \* Nominal domain:  $S$
  - \* Proper domain:  $\text{dom}(f) = \{w \in S : f(w) = \downarrow\}$
  - \* If  $S = \text{dom}(f)$ , then  $f$  is total, written  $f : S \rightarrow \Sigma^*$
  
- Two cases of interest:
  - \* (Discrete computation) Turing Machines for computing  $f : S \subseteq \Sigma^* \dashrightarrow \Sigma^*$
  - \* (Continuous computation) Real functions  $f : S \subseteq \mathbb{R} \dashrightarrow \mathbb{R}$
  
- Why have both  $S$  and partial  $f$ ?
  - \* In analysis: we often choose  $S$  to be nice
  - \* In algebra: we have no choice about  $S$ , and  $f$  may be partial (E.g.,  $\div$ )

# Partial Functions

- Partial function,  $f : S \dashrightarrow T$ 
  - \* Nominal domain:  $S$
  - \* Proper domain:  $\text{dom}(f) = \{w \in S : f(w) = \downarrow\}$
  - \* If  $S = \text{dom}(f)$ , then  $f$  is total, written  $f : S \rightarrow \Sigma^*$
  
- Two cases of interest:
  - \* (Discrete computation) Turing Machines for computing  $f : S \subseteq \Sigma^* \dashrightarrow \Sigma^*$
  - \* (Continuous computation) Real functions  $f : S \subseteq \mathbb{R} \dashrightarrow \mathbb{R}$
  
- Why have both  $S$  and partial  $f$ ?
  - \* In analysis: we often choose  $S$  to be nice
  - \* In algebra: we have no choice about  $S$ , and  $f$  may be partial (E.g.,  $\div$ )

# Partial Functions

- Partial function,  $f : S \dashrightarrow T$ 
  - \* Nominal domain:  $S$
  - \* Proper domain:  $\text{dom}(f) = \{w \in S : f(w) = \downarrow\}$
  - \* If  $S = \text{dom}(f)$ , then  $f$  is total, written  $f : S \rightarrow \Sigma^*$
  
- Two cases of interest:
  - \* (Discrete computation) Turing Machines for computing  $f : S \subseteq \Sigma^* \dashrightarrow \Sigma^*$
  - \* (Continuous computation) Real functions  $f : S \subseteq \mathbb{R} \dashrightarrow \mathbb{R}$
  
- Why have both  $S$  and partial  $f$ ?
  - \* In analysis: we often choose  $S$  to be nice
  - \* In algebra: we have no choice about  $S$ , and  $f$  may be partial (E.g.,  $\div$ )

# Partial Functions

- Partial function,  $f : S \dashrightarrow T$ 
  - \* Nominal domain:  $S$
  - \* Proper domain:  $\text{dom}(f) = \{w \in S : f(w) = \downarrow\}$
  - \* If  $S = \text{dom}(f)$ , then  $f$  is total, written  $f : S \rightarrow \Sigma^*$
  
- Two cases of interest:
  - \* (Discrete computation) Turing Machines for computing  $f : S \subseteq \Sigma^* \dashrightarrow \Sigma^*$
  - \* (Continuous computation) Real functions  $f : S \subseteq \mathbb{R} \dashrightarrow \mathbb{R}$
  
- Why have both  $S$  and partial  $f$ ?
  - \* In analysis: we often choose  $S$  to be nice
  - \* In algebra: we have no choice about  $S$ , and  $f$  may be partial (E.g.,  $\div$ )

# Classic (Discrete) Computability Theory

- $f$  is **partial recursive** if there is a Turing machine  $M$  :
  - \* (1) For all  $w \in \text{dom}(f)$ ,  $M$  halts with outputs  $f(w)$
  - \* (2) For all  $w \notin \text{dom}(f)$ ,  $M$  does not halt
  
- $f$  is **recursive** if ...
  - \* (1) For all  $w \in \text{dom}(f)$ , ...
  - \* (2) For all  $w \notin \text{dom}(f)$ ,  $M$  halts in a state  $q_{\uparrow}$
  
- E.g.,  $\div : \mathbb{Q} \times \mathbb{Q} \dashrightarrow \mathbb{Q}$  a recursive partial (!) function
  
- $Rec =$  Recursive functions;  
 $Prec =$  Partial Recursive functions

# Classic (Discrete) Computability Theory

10

- $f$  is partial recursive if there is a Turing machine  $M$  :
  - \* (1) For all  $w \in \text{dom}(f)$ ,  $M$  halts with outputs  $f(w)$
  - \* (2) For all  $w \notin \text{dom}(f)$ ,  $M$  does not halt
- $f$  is **recursive** if ...
  - \* (1) For all  $w \in \text{dom}(f)$ , ...
  - \* (2) For all  $w \notin \text{dom}(f)$ ,  $M$  halts in a state  $q_{\uparrow}$
- E.g.,  $\div : \mathbb{Q} \times \mathbb{Q} \dashrightarrow \mathbb{Q}$  a recursive partial (!) function
- $Rec =$  Recursive functions;  
 $Prec =$  Partial Recursive functions

# Classic (Discrete) Computability Theory

10

- $f$  is partial recursive if there is a Turing machine  $M$  :
  - \* (1) For all  $w \in \text{dom}(f)$ ,  $M$  halts with outputs  $f(w)$
  - \* (2) For all  $w \notin \text{dom}(f)$ ,  $M$  does not halt
- $f$  is recursive if ...
  - \* (1) For all  $w \in \text{dom}(f)$ , ...
  - \* (2) For all  $w \notin \text{dom}(f)$ ,  $M$  halts in a state  $q_{\uparrow}$
- E.g.,  $\div : \mathbb{Q} \times \mathbb{Q} \dashrightarrow \mathbb{Q}$  a recursive partial (!) function
- $Rec =$  Recursive functions;  
 $Prec =$  Partial Recursive functions



# Classic (Discrete) Computability Theory

10

- $f$  is partial recursive if there is a Turing machine  $M$  :
  - \* (1) For all  $w \in \text{dom}(f)$ ,  $M$  halts with outputs  $f(w)$
  - \* (2) For all  $w \notin \text{dom}(f)$ ,  $M$  does not halt
- $f$  is recursive if ...
  - \* (1) For all  $w \in \text{dom}(f)$ , ...
  - \* (2) For all  $w \notin \text{dom}(f)$ ,  $M$  halts in a state  $q_{\uparrow}$
- E.g.,  $\div : \mathbb{Q} \times \mathbb{Q} \dashrightarrow \mathbb{Q}$  a recursive partial (!) function
- $Rec =$  Recursive functions;  
 $Prec =$  Partial Recursive functions

# Classic (Discrete) Computability Theory

- $f$  is partial recursive if there is a Turing machine  $M$  :
  - \* (1) For all  $w \in \text{dom}(f)$ ,  $M$  halts with outputs  $f(w)$
  - \* (2) For all  $w \notin \text{dom}(f)$ ,  $M$  does not halt
  
- $f$  is recursive if ...
  - \* (1) For all  $w \in \text{dom}(f)$ , ...
  - \* (2) For all  $w \notin \text{dom}(f)$ ,  $M$  halts in a state  $q_{\uparrow}$
  
- E.g.,  $\div : \mathbb{Q} \times \mathbb{Q} \dashrightarrow \mathbb{Q}$  a recursive partial (!) function
  
- $Rec =$  Recursive functions;  
 $Prec =$  Partial Recursive functions

# Classic (Discrete) Computability Theory

11

- $Rec \subseteq Prec$ 
  - \* There exists a function  $HALT$  in  $Prec \setminus Rec$
  - \* Every  $f \in Prec$  is reducible to  $HALT$
- What happens to the Halting Problem in the continuous domain?

# Classic (Discrete) Computability Theory

11

- $Rec \subseteq Prec$ 
  - \* There exists a function  $HALT$  in  $Prec \setminus Rec$
  - \* Every  $f \in Prec$  is reducible to  $HALT$
- What happens to the Halting Problem in the continuous domain?

# Classic (Discrete) Computability Theory

11

- $Rec \subseteq Prec$ 
  - \* There exists a function  $HALT$  in  $Prec \setminus Rec$
  - \* Every  $f \in Prec$  is reducible to  $HALT$
- What happens to the Halting Problem in the continuous domain?

# Real Approximations

- A set  $\mathbb{F}$  of **base reals** is any subset  $\mathbb{F} \subseteq \mathbb{R}$  such that
  - \* (1)  $\mathbb{F}$  is a ring extension of  $\mathbb{Z}$
  - \* (2)  $\mathbb{F}$  is countably dense in  $\mathbb{R}$
  - \* (3) Ring operations,  $x \mapsto x/2$  and comparisons of  $\mathbb{F}$  are efficient
  
- E.g.,  $\mathbb{F} = \mathbb{Q}$     or     $\mathbb{F} = \{n2^m : n, m \in \mathbb{Z}\}$
  
- Let  $f : S \subseteq \mathbb{R} \dashrightarrow \mathbb{R}$     and     $\tilde{f} : \mathbb{F} \times \mathbb{Z} \dashrightarrow \mathbb{F}$
  
- $\tilde{f}$  is an  $\mathcal{A}$ -approximation of  $f$  if:
  - for all  $x \in \mathbb{F}$ ,  $\tilde{f}(x; p) \equiv f(x) \pm 2^{-p}$ .
  - \* Similarly for  $\mathcal{R}$ -approximable.

# Real Approximations

- A set  $\mathbb{F}$  of base reals is any subset  $\mathbb{F} \subseteq \mathbb{R}$  such that
  - \* (1)  $\mathbb{F}$  is a ring extension of  $\mathbb{Z}$
  - \* (2)  $\mathbb{F}$  is countably dense in  $\mathbb{R}$
  - \* (3) Ring operations,  $x \mapsto x/2$  and comparisons of  $\mathbb{F}$  are efficient
  
- E.g.,  $\mathbb{F} = \mathbb{Q}$     **or**     $\mathbb{F} = \{n2^m : n, m \in \mathbb{Z}\}$
  
- Let  $f : S \subseteq \mathbb{R} \dashrightarrow \mathbb{R}$     and     $\tilde{f} : \mathbb{F} \times \mathbb{Z} \dashrightarrow \mathbb{F}$
  
- $\tilde{f}$  is an  $\mathcal{A}$ -approximation of  $f$  if:
  - for all  $x \in \mathbb{F}$ ,  $\tilde{f}(x; p) \equiv f(x) \pm 2^{-p}$ .
  - \* Similarly for  $\mathcal{R}$ -approximable.

# Real Approximations

- A set  $\mathbb{F}$  of base reals is any subset  $\mathbb{F} \subseteq \mathbb{R}$  such that
  - \* (1)  $\mathbb{F}$  is a ring extension of  $\mathbb{Z}$
  - \* (2)  $\mathbb{F}$  is countably dense in  $\mathbb{R}$
  - \* (3) Ring operations,  $x \mapsto x/2$  and comparisons of  $\mathbb{F}$  are efficient
- E.g.,  $\mathbb{F} = \mathbb{Q}$     or     $\mathbb{F} = \{n2^m : n, m \in \mathbb{Z}\}$
- Let  $f : S \subseteq \mathbb{R} \dashrightarrow \mathbb{R}$     and     $\tilde{f} : \mathbb{F} \times \mathbb{Z} \dashrightarrow \mathbb{F}$
- $\tilde{f}$  is an  $\mathcal{A}$ -approximation of  $f$  if:
  - for all  $x \in \mathbb{F}$ ,  $\tilde{f}(x; p) \equiv f(x) \pm 2^{-p}$ .
  - \* Similarly for  $\mathcal{R}$ -approximable.



# Real Approximations

- A set  $\mathbb{F}$  of base reals is any subset  $\mathbb{F} \subseteq \mathbb{R}$  such that
  - \* (1)  $\mathbb{F}$  is a ring extension of  $\mathbb{Z}$
  - \* (2)  $\mathbb{F}$  is countably dense in  $\mathbb{R}$
  - \* (3) Ring operations,  $x \mapsto x/2$  and comparisons of  $\mathbb{F}$  are efficient
- E.g.,  $\mathbb{F} = \mathbb{Q}$     or     $\mathbb{F} = \{n2^m : n, m \in \mathbb{Z}\}$
- Let  $f : S \subseteq \mathbb{R} \dashrightarrow \mathbb{R}$     and     $\tilde{f} : \mathbb{F} \times \mathbb{Z} \dashrightarrow \mathbb{F}$
- $\tilde{f}$  is an  **$\mathcal{A}$ -approximation** of  $f$  if:
  - for all  $x \in \mathbb{F}$ ,  $\tilde{f}(x; p) \equiv f(x) \pm 2^{-p}$ .
  - \* Similarly for  $\mathcal{R}$ -approximable.

# Real Approximations

- A set  $\mathbb{F}$  of base reals is any subset  $\mathbb{F} \subseteq \mathbb{R}$  such that
  - \* (1)  $\mathbb{F}$  is a ring extension of  $\mathbb{Z}$
  - \* (2)  $\mathbb{F}$  is countably dense in  $\mathbb{R}$
  - \* (3) Ring operations,  $x \mapsto x/2$  and comparisons of  $\mathbb{F}$  are efficient
- E.g.,  $\mathbb{F} = \mathbb{Q}$     or     $\mathbb{F} = \{n2^m : n, m \in \mathbb{Z}\}$
- Let  $f : S \subseteq \mathbb{R} \dashrightarrow \mathbb{R}$     and     $\tilde{f} : \mathbb{F} \times \mathbb{Z} \dashrightarrow \mathbb{F}$
- $\tilde{f}$  is an  $\mathcal{A}$ -approximation of  $f$  if:
  - for all  $x \in \mathbb{F}$ ,  $\tilde{f}(x; p) \equiv f(x) \pm 2^{-p}$ .
  - \* Similarly for  $\mathcal{R}$ -approximable.

- Let  $\mathcal{A}_f$  denote (an arbitrary member of) the set of all  $\mathcal{A}$ -approximations of  $f$ .<sup>13</sup>
- $f$  is  $\mathcal{A}$ -approxible if some  $f \in \mathcal{A}_f$  is computable by a halting Turing machine
  - \* Similarly for  $\mathcal{R}$ -approximable and the set  $\mathcal{R}_f$ .
- E.g., the field operations  $\pm, \times, \div$  and  $\exp, \log$  are all  $\mathcal{R}$ -approximable.

- Let  $\mathcal{A}_f$  denote (an arbitrary member of) the set of all  $\mathcal{A}$ -approximations of  $f$ .
- $f$  is  **$\mathcal{A}$ -approxible** if some  $f \in \mathcal{A}_f$  is computable by a halting Turing machine
  - \* Similarly for  $\mathcal{R}$ -approximable and the set  $\mathcal{R}_f$ .
- E.g., the field operations  $\pm, \times, \div$  and  $\exp, \log$  are all  $\mathcal{R}$ -approximable.

- Let  $\mathcal{A}_f$  denote (an arbitrary member of) the set of all  $\mathcal{A}$ -approximations of  $f$ .
- $f$  is  $\mathcal{A}$ -approxible if some  $f \in \mathcal{A}_f$  is computable by a halting Turing machine
  - \* Similarly for  $\mathcal{R}$ -approximable and the set  $\mathcal{R}_f$ .
- E.g., the field operations  $\pm, \times, \div$  and  $\exp, \log$  are all  $\mathcal{R}$ -approximable.



# Sign Lemma

- SIGN LEMMA:  
 $\text{sign}(f(x)) = \text{sign}(\mathcal{R}_f(x; 1))$
- Proof:  $\mathcal{R}_f(x; 1) = f(x)(1 \pm 2^{-1})$ .

# Zero Problem in Real Approximations

- $Zero(f)$  is the problem of deciding for  $x \in \mathbb{F}$ , whether  $f(x) = 0$
- THEOREM: The following is equivalent:
  - \*  $f$  is  $\mathcal{R}$ -approximable
  - \*  $f$  is  $\mathcal{A}$ -approximable and  $Zero(f)$  is decidable
- Proof: ( $\Rightarrow$ ) Given  $x$  and  $p$ , we want to compute  $\mathcal{A}_f(x; p)$ .
  - \* (1) compute  $c = \mathcal{R}_f(x; 1)$
  - \* (2)  $x \in Zero(f)$  iff  $c = 0$  (by SIGN LEMMA)
  - \* (3) If  $c \neq 0$ , OUTPUT  $\mathcal{R}_f(x, p + 1 + \lceil \lg |c| \rceil)$
- Proof: ( $\Leftarrow$ ) Given  $x$  and  $p$ , we want to compute  $\mathcal{R}_f(x; p)$ .
  - \* (1) If  $f(x) = 0$ , OUTPUT 0
  - \* (2) Find the first  $n$  such that  $|\mathcal{A}_f(x; n)| \geq 2^{1-n}$
  - \* (3) So  $|f(x)| \geq 2^{-n}$ . OUTPUT  $\mathcal{A}_f(x; n + p)$



# Zero Problem in Real Approximations

- $Zero(f)$  is the problem of deciding for  $x \in \mathbb{F}$ , whether  $f(x) = 0$
- THEOREM: The following is equivalent:
  - \*  $f$  is  $\mathcal{R}$ -approximable
  - \*  $f$  is  $\mathcal{A}$ -approximable and  $Zero(f)$  is decidable
- Proof: ( $\Rightarrow$ ) Given  $x$  and  $p$ , we want to compute  $\mathcal{A}_f(x; p)$ .
  - \* (1) compute  $c = \mathcal{R}_f(x; 1)$
  - \* (2)  $x \in Zero(f)$  iff  $c = 0$  (by SIGN LEMMA)
  - \* (3) If  $c \neq 0$ , OUTPUT  $\mathcal{R}_f(x, p + 1 + \lceil \lg |c| \rceil)$
- Proof: ( $\Leftarrow$ ) Given  $x$  and  $p$ , we want to compute  $\mathcal{R}_f(x; p)$ .
  - \* (1) If  $f(x) = 0$ , OUTPUT 0
  - \* (2) Find the first  $n$  such that  $|\mathcal{A}_f(x; n)| \geq 2^{1-n}$
  - \* (3) So  $|f(x)| \geq 2^{-n}$ . OUTPUT  $\mathcal{A}_f(x; n + p)$

# Zero Problem in Real Approximations

- $Zero(f)$  is the problem of deciding for  $x \in \mathbb{F}$ , whether  $f(x) = 0$
- THEOREM: The following is equivalent:
  - \*  $f$  is  $\mathcal{R}$ -approximable
  - \*  $f$  is  $\mathcal{A}$ -approximable and  $Zero(f)$  is decidable
- Proof: ( $\Rightarrow$ ) Given  $x$  and  $p$ , we want to compute  $\mathcal{A}_f(x; p)$ .
  - \* (1) compute  $c = \mathcal{R}_f(x; 1)$
  - \* (2)  $x \in Zero(f)$  iff  $c = 0$  (by SIGN LEMMA)
  - \* (3) If  $c \neq 0$ , OUTPUT  $\mathcal{R}_f(x, p + 1 + \lceil \lg |c| \rceil)$
- Proof: ( $\Leftarrow$ ) Given  $x$  and  $p$ , we want to compute  $\mathcal{R}_f(x; p)$ .
  - \* (1) If  $f(x) = 0$ , OUTPUT 0
  - \* (2) Find the first  $n$  such that  $|\mathcal{A}_f(x; n)| \geq 2^{1-n}$
  - \* (3) So  $|f(x)| \geq 2^{-n}$ . OUTPUT  $\mathcal{A}_f(x; n + p)$

# Zero Problem in Real Approximations

- $Zero(f)$  is the problem of deciding for  $x \in \mathbb{F}$ , whether  $f(x) = 0$
- THEOREM: The following is equivalent:
  - \*  $f$  is  $\mathcal{R}$ -approximable
  - \*  $f$  is  $\mathcal{A}$ -approximable and  $Zero(f)$  is decidable
- Proof: ( $\Rightarrow$ ) Given  $x$  and  $p$ , we want to compute  $\mathcal{A}_f(x; p)$ .
  - \* (1) compute  $c = \mathcal{R}_f(x; 1)$
  - \* (2)  $x \in Zero(f)$  iff  $c = 0$  (by SIGN LEMMA)
  - \* (3) If  $c \neq 0$ , OUTPUT  $\mathcal{R}_f(x, p + 1 + \lceil \lg |c| \rceil)$
- Proof: ( $\Leftarrow$ ) Given  $x$  and  $p$ , we want to compute  $\mathcal{R}_f(x; p)$ .
  - \* (1) If  $f(x) = 0$ , OUTPUT 0
  - \* (2) Find the first  $n$  such that  $|\mathcal{A}_f(x; n)| \geq 2^{1-n}$
  - \* (3) So  $|f(x)| \geq 2^{-n}$ . OUTPUT  $\mathcal{A}_f(x; n + p)$

# Zero Problem in Real Approximations

- $Zero(f)$  is the problem of deciding for  $x \in \mathbb{F}$ , whether  $f(x) = 0$
- THEOREM: The following is equivalent:
  - \*  $f$  is  $\mathcal{R}$ -approximable
  - \*  $f$  is  $\mathcal{A}$ -approximable and  $Zero(f)$  is decidable
- Proof: ( $\Rightarrow$ ) Given  $x$  and  $p$ , we want to compute  $\mathcal{A}_f(x; p)$ .
  - \* (1) compute  $c = \mathcal{R}_f(x; 1)$
  - \* (2)  $x \in Zero(f)$  iff  $c = 0$  (by SIGN LEMMA)
  - \* (3) If  $c \neq 0$ , OUTPUT  $\mathcal{R}_f(x, p + 1 + \lceil \lg |c| \rceil)$
- Proof: ( $\Leftarrow$ ) Given  $x$  and  $p$ , we want to compute  $\mathcal{R}_f(x; p)$ .
  - \* (1) If  $f(x) = 0$ , OUTPUT 0
  - \* (2) Find the first  $n$  such that  $|\mathcal{A}_f(x; n)| \geq 2^{1-n}$
  - \* (3) So  $|f(x)| \geq 2^{-n}$ . OUTPUT  $\mathcal{A}_f(x; n + p)$

## Separation of *Abs* and *Rel*

- THEOREM: There is an  $\mathcal{A}$ -approximable function that is not  $\mathcal{R}$ -approximable.
- Proof: Let  $H : \mathbb{N} \rightarrow \mathbb{F}$  where  $H(i) = \begin{cases} 0 & \text{if } i\text{-th TM on } 0 \text{ halts} \\ 2^{-k} & \text{if } i\text{-th TM on } 0 \text{ halts in } k \text{ steps} \end{cases}$ 
  - \*  $H$  is not  $\mathcal{R}$ -approximable: otherwise  $HALT$  is recursive.
  - \*  $H$  is  $\mathcal{A}$ -approximable: on  $i, j$  we can compute  $\tilde{H}(i; j)$  such that  $|\tilde{H}(i; j) - H(i)| \leq 2^{-j}$
- OPEN PROBLEM: Find a Natural Example

# Separation of *Abs* and *Rel*

- THEOREM: There is an  $\mathcal{A}$ -approximable function that is not  $\mathcal{R}$ -approximable.
- Proof: Let  $H : \mathbb{N} \rightarrow \mathbb{F}$  where  $H(i) = \begin{cases} 0 & \text{if } i\text{-th TM on } 0 \text{ halts} \\ 2^{-k} & \text{if } i\text{-th TM on } 0 \text{ halts in } k \text{ steps} \end{cases}$ 
  - \*  $H$  is not  $\mathcal{R}$ -approximable: otherwise  $HALT$  is recursive.
  - \*  $H$  is  $\mathcal{A}$ -approximable: on  $i, j$  we can compute  $\tilde{H}(i; j)$  such that  $|\tilde{H}(i; j) - H(i)| \leq 2^{-j}$
- OPEN PROBLEM: Find a Natural Example

## Separation of *Abs* and *Rel*

- THEOREM: There is an  $\mathcal{A}$ -approximable function that is not  $\mathcal{R}$ -approximable.
- Proof: Let  $H : \mathbb{N} \rightarrow \mathbb{F}$  where  $H(i) = \begin{cases} 0 & \text{if } i\text{-th TM on } 0 \text{ halts} \\ 2^{-k} & \text{if } i\text{-th TM on } 0 \text{ halts in } k \text{ steps} \end{cases}$ 
  - \*  $H$  is not  $\mathcal{R}$ -approximable: otherwise  $HALT$  is recursive.
  - \*  $H$  is  $\mathcal{A}$ -approximable: on  $i, j$  we can compute  $\tilde{H}(i; j)$  such that  $|\tilde{H}(i; j) - H(i)| \leq 2^{-j}$
- OPEN PROBLEM: Find a Natural Example

## Separation of *Abs* and *Rel*

- THEOREM: There is an  $\mathcal{A}$ -approximable function that is not  $\mathcal{R}$ -approximable.
- Proof: Let  $H : \mathbb{N} \rightarrow \mathbb{F}$  where  $H(i) = \begin{cases} 0 & \text{if } i\text{-th TM on } 0 \text{ halts} \\ 2^{-k} & \text{if } i\text{-th TM on } 0 \text{ halts in } k \text{ steps} \end{cases}$ 
  - \*  $H$  is not  $\mathcal{R}$ -approximable: otherwise  $HALT$  is recursive.
  - \*  $H$  is  $\mathcal{A}$ -approximable: on  $i, j$  we can compute  $\tilde{H}(i; j)$  such that  $|\tilde{H}(i; j) - H(i)| \leq 2^{-j}$
- OPEN PROBLEM: Find a Natural Example



# Complexity of Zero Problems

- Useful complexity classification
- Zero Hierarchy
  - \* Polynomial:  $\Omega_0 := \{+, -, \times\} \cup \mathbb{Z}$
  - \* Rational:  $\Omega_1 := \Omega_0 \cup \{\div\}$
  - \* Radical:  $\Omega_2 := \Omega_1 \cup \{\sqrt[k]{\cdot} : k \geq 2\}$
  - \* Algebraic:  $\Omega_3 := \Omega_2 \cup \{\text{RootOf}\}$
  - \* Elementary:  $\Omega_4 = \Omega_3 \cup \{\exp, \log\}$
- Complexity
  - \*  $\text{ZERO}(\Omega_1)$  is in *PSPACE* and *P*-complete [Mehlhorn-Schmitt-Yap]
  - \*  $\text{ZERO}(\Omega_3)$  is decidable in Single Exponential Time [Tarski, Grigoriev, etc]
  - \*  $\text{ZERO}(\Omega_4)$  is decidable provided Schanuel's conjecture holds [Richardson]
- OPEN PROBLEM: Is  $\text{ZERO}(\Omega_4)$  decidable?

# Complexity of Zero Problems

- Useful complexity classification
- Zero Hierarchy
  - \* Polynomial:  $\Omega_0 := \{+, -, \times\} \cup \mathbb{Z}$
  - \* Rational:  $\Omega_1 := \Omega_0 \cup \{\div\}$
  - \* Radical:  $\Omega_2 := \Omega_1 \cup \{\sqrt[k]{\cdot} : k \geq 2\}$
  - \* Algebraic:  $\Omega_3 := \Omega_2 \cup \{\text{RootOf}\}$
  - \* Elementary:  $\Omega_4 = \Omega_3 \cup \{\exp, \log\}$
- Complexity
  - \*  $\text{ZERO}(\Omega_1)$  is in *PSPACE* and *P*-complete [Mehlhorn-Schmitt-Yap]
  - \*  $\text{ZERO}(\Omega_3)$  is decidable in Single Exponential Time [Tarski, Grigoriev, etc]
  - \*  $\text{ZERO}(\Omega_4)$  is decidable provided Schanuel's conjecture holds [Richardson]
- OPEN PROBLEM: Is  $\text{ZERO}(\Omega_4)$  decidable?

# Complexity of Zero Problems

- Useful complexity classification
- Zero Hierarchy
  - \* Polynomial:  $\Omega_0 := \{+, -, \times\} \cup \mathbb{Z}$
  - \* Rational:  $\Omega_1 := \Omega_0 \cup \{\div\}$
  - \* Radical:  $\Omega_2 := \Omega_1 \cup \{\sqrt[k]{\cdot} : k \geq 2\}$
  - \* Algebraic:  $\Omega_3 := \Omega_2 \cup \{\text{RootOf}\}$
  - \* Elementary:  $\Omega_4 = \Omega_3 \cup \{\exp, \log\}$
- Complexity
  - \*  $ZERO(\Omega_1)$  is in *PSPACE* and *P*-complete [Mehlhorn-Schmitt-Yap]
  - \*  $ZERO(\Omega_3)$  is decidable in Single Exponential Time [Tarski, Grigoriev, etc]
  - \*  $ZERO(\Omega_4)$  is decidable provided Schanuel's conjecture holds [Richardson]
- OPEN PROBLEM: Is  $ZERO(\Omega_4)$  decidable?

# Complexity of Zero Problems

- Useful complexity classification
- Zero Hierarchy
  - \* Polynomial:  $\Omega_0 := \{+, -, \times\} \cup \mathbb{Z}$
  - \* Rational:  $\Omega_1 := \Omega_0 \cup \{\div\}$
  - \* Radical:  $\Omega_2 := \Omega_1 \cup \{\sqrt[k]{\cdot} : k \geq 2\}$
  - \* Algebraic:  $\Omega_3 := \Omega_2 \cup \{\text{RootOf}\}$
  - \* Elementary:  $\Omega_4 = \Omega_3 \cup \{\exp, \log\}$
- Complexity
  - \*  $\text{ZERO}(\Omega_1)$  is in *PSPACE* and *P*-complete [Mehlhorn-Schmitt-Yap]
  - \*  $\text{ZERO}(\Omega_3)$  is decidable in Single Exponential Time [Tarski, Grigoriev, etc]
  - \*  $\text{ZERO}(\Omega_4)$  is decidable provided Schanuel's conjecture holds [Richardson]
- OPEN PROBLEM: Is  $\text{ZERO}(\Omega_4)$  decidable?

# Complexity of Zero Problems

- Useful complexity classification
- Zero Hierarchy
  - \* Polynomial:  $\Omega_0 := \{+, -, \times\} \cup \mathbb{Z}$
  - \* Rational:  $\Omega_1 := \Omega_0 \cup \{\div\}$
  - \* Radical:  $\Omega_2 := \Omega_1 \cup \{\sqrt[k]{\cdot} : k \geq 2\}$
  - \* Algebraic:  $\Omega_3 := \Omega_2 \cup \{\text{RootOf}\}$
  - \* Elementary:  $\Omega_4 = \Omega_3 \cup \{\exp, \log\}$
- Complexity
  - \*  $\text{ZERO}(\Omega_1)$  is in *PSPACE* and *P*-complete [Mehlhorn-Schmitt-Yap]
  - \*  $\text{ZERO}(\Omega_3)$  is decidable in Single Exponential Time [Tarski, Grigoriev, etc]
  - \*  $\text{ZERO}(\Omega_4)$  is decidable provided Schanuel's conjecture holds [Richardson]
- OPEN PROBLEM: Is  $\text{ZERO}(\Omega_4)$  decidable?

## PART II.

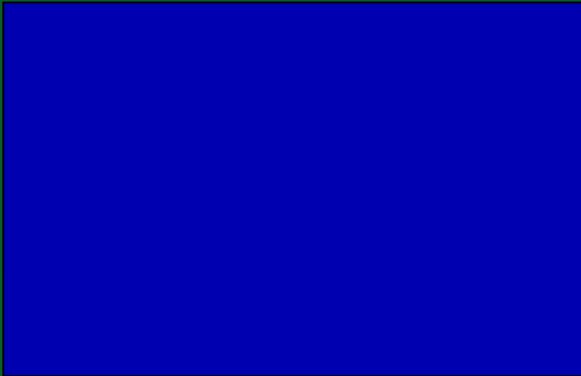
# Why Care About Nothing (Zero)?

*Much Ado About Nothing*

– Shakespeare (1600)

# Points and Lines

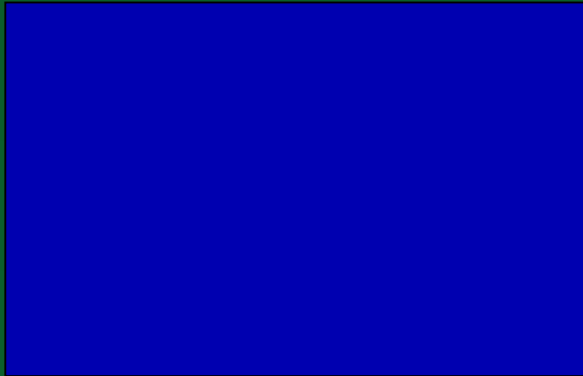
- Simple geometric test:
  - \* Is a point  $P$  on a line  $L$ ?



- $P$  is on  $L \iff ax_0 + by_0 + c = 0$ 
  - \*  $L : aX + bY + c = 0$
  - \*  $P : (x_0, y_0)$
- In Meshing Applications
  - \* Point Classification Problem: Is  $P$  IN/OUT/ON a given triangle?
  - \* Sign determination
- Upshot
  - \* Knowing zero (and sign) is necessary for computing correct geometry

# Points and Lines

- Simple geometric test:
  - \* Is a point  $P$  on a line  $L$ ?

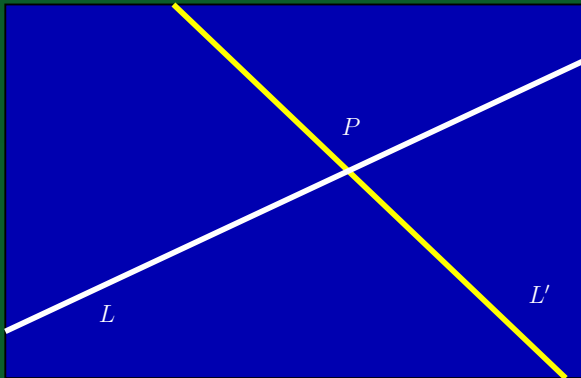


- $P$  is on  $L \iff ax_0 + by_0 + c = 0$ 
  - \*  $L : aX + bY + c = 0$
  - \*  $P : (x_0, y_0)$
- In Meshing Applications
  - \* Point Classification Problem: Is  $P$  IN/OUT/ON a given triangle?
  - \* Sign determination
- Upshot
  - \* Knowing zero (and sign) is necessary for computing correct geometry



# Points and Lines

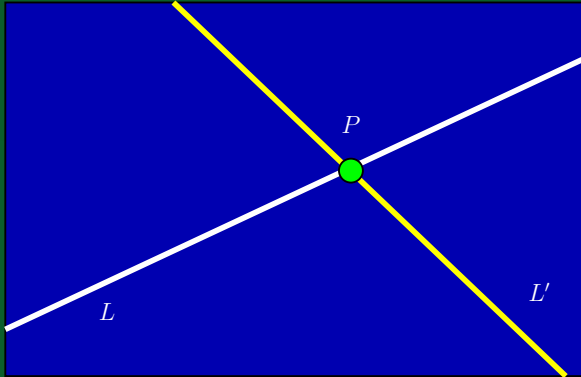
- Simple geometric test:
  - \* Is a point  $P$  on a line  $L$ ?



- $P$  is on  $L \iff ax_0 + by_0 + c = 0$ 
  - \*  $L : aX + bY + c = 0$
  - \*  $P : (x_0, y_0)$
- In Meshing Applications
  - \* Point Classification Problem: Is  $P$  IN/OUT/ON a given triangle?
  - \* Sign determination
- Upshot
  - \* Knowing zero (and sign) is necessary for computing correct geometry

# Points and Lines

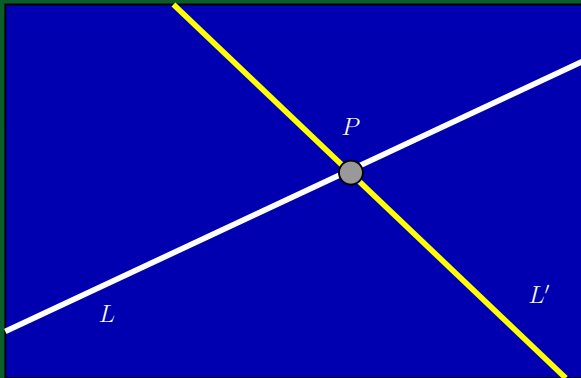
- Simple geometric test:
  - \* Is a point  $P$  on a line  $L$ ?



- $P$  is on  $L \iff ax_0 + by_0 + c = 0$ 
  - \*  $L : aX + bY + c = 0$
  - \*  $P : (x_0, y_0)$
- In Meshing Applications
  - \* Point Classification Problem: Is  $P$  IN/OUT/ON a given triangle?
  - \* Sign determination
- Upshot
  - \* Knowing zero (and sign) is necessary for computing correct geometry

# Points and Lines

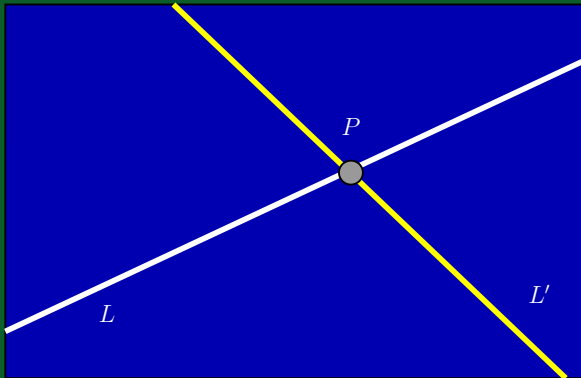
- Simple geometric test:
  - \* Is a point  $P$  on a line  $L$ ?



- $P$  is on  $L \iff ax_0 + by_0 + c = 0$ 
  - \*  $L : aX + bY + c = 0$
  - \*  $P : (x_0, y_0)$
- In Meshing Applications
  - \* Point Classification Problem: Is  $P$  IN/OUT/ON a given triangle?
  - \* Sign determination
- Upshot
  - \* Knowing zero (and sign) is necessary for computing correct geometry

# Points and Lines

- Simple geometric test:
  - \* Is a point  $P$  on a line  $L$ ?



- $P$  is on  $L \iff ax_0 + by_0 + c = 0$ 
  - \*  $L : aX + bY + c = 0$
  - \*  $P : (x_0, y_0)$

- In Meshing Applications

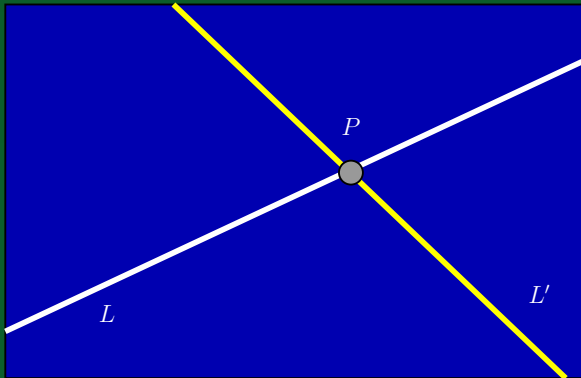
- \* Point Classification Problem: Is  $P$  IN/OUT/ON a given triangle?
- \* Sign determination

- Upshot

- \* Knowing zero (and sign) is necessary for computing correct geometry

# Points and Lines

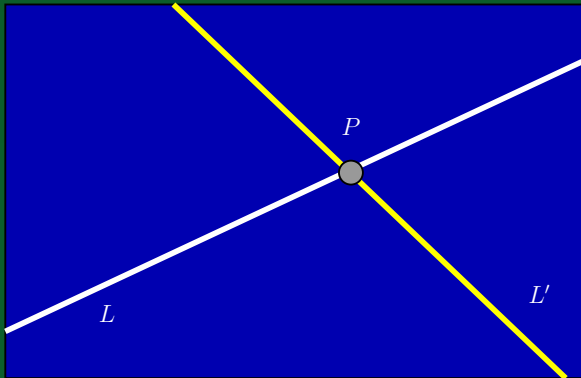
- Simple geometric test:
  - \* Is a point  $P$  on a line  $L$ ?



- $P$  is on  $L \iff ax_0 + by_0 + c = 0$ 
  - \*  $L : aX + bY + c = 0$
  - \*  $P : (x_0, y_0)$
- In Meshing Applications
  - \* Point Classification Problem: Is  $P$  IN/OUT/ON a given triangle?
  - \* Sign determination
- Upshot
  - \* Knowing zero (and sign) is necessary for computing correct geometry

# Points and Lines

- Simple geometric test:
  - \* Is a point  $P$  on a line  $L$ ?



- $P$  is on  $L \iff ax_0 + by_0 + c = 0$ 
  - \*  $L : aX + bY + c = 0$
  - \*  $P : (x_0, y_0)$
- In Meshing Applications
  - \* Point Classification Problem: Is  $P$  IN/OUT/ON a given triangle?
  - \* Sign determination
- Upshot
  - \* Knowing zero (and sign) is necessary for computing correct geometry



# Again, What is Geometry?

- Geometric vein permeates all of Mathematics
  - \* Number theory, algebraic & differential geometry, topology, probability, ...
- What is Geometry?
  - \* Euclid: Axiomatic Approach
  - \* Descartes: Algebraization of Geometry
  - \* Klein: Transformation Groups
  - \* Hilbert: Logical Foundations
  - \* Tarski: Elementary Geometry and Algebra
  - \* Erdos: Combinatorial Vein
- Computational Perspective:
  - \* (1) Geometry is comprised of discrete relations among geometric objects
  - \* (2) Computational Geometry computes these relations, by deciding zero & sign of expressions
- Exact Geometric Computation (EGC):
  - an approach requiring error-free zero & sign computation



# Again, What is Geometry?

- Geometric vein permeates all of Mathematics
  - \* Number theory, algebraic & differential geometry, topology, probability, ...
- What is Geometry?
  - \* Euclid: **Axiomatic Approach**
  - \* Descartes: **Algebraization of Geometry**
  - \* Klein: **Transformation Groups**
  - \* Hilbert: **Logical Foundations**
  - \* Tarski: **Elementary Geometry and Algebra**
  - \* Erdos: **Combinatorial Vein**
- Computational Perspective:
  - \* (1) Geometry is comprised of discrete relations among geometric objects
  - \* (2) Computational Geometry computes these relations, by deciding zero & sign of expressions
- Exact Geometric Computation (EGC):
  - an approach requiring error-free zero & sign computation

# Again, What is Geometry?

- Geometric vein permeates all of Mathematics
  - \* Number theory, algebraic & differential geometry, topology, probability, ...
- What is Geometry?
  - \* Euclid: Axiomatic Approach
  - \* Descartes: Algebraization of Geometry
  - \* Klein: Transformation Groups
  - \* Hilbert: Logical Foundations
  - \* Tarski: Elementary Geometry and Algebra
  - \* Erdos: Combinatorial Vein
- Computational Perspective:
  - \* (1) Geometry is comprised of discrete relations among geometric objects
  - \* (2) Computational Geometry computes these relations, by deciding zero & sign of expressions
- Exact Geometric Computation (EGC):
  - an approach requiring error-free zero & sign computation

# Again, What is Geometry?

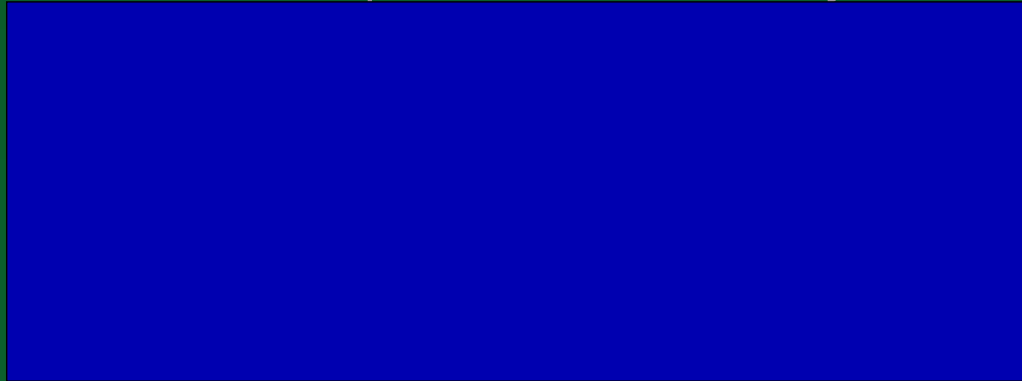
- Geometric vein permeates all of Mathematics
  - \* Number theory, algebraic & differential geometry, topology, probability, ...
- What is Geometry?
  - \* Euclid: Axiomatic Approach
  - \* Descartes: Algebraization of Geometry
  - \* Klein: Transformation Groups
  - \* Hilbert: Logical Foundations
  - \* Tarski: Elementary Geometry and Algebra
  - \* Erdos: Combinatorial Vein
- Computational Perspective:
  - \* (1) Geometry is comprised of discrete relations among geometric objects
  - \* (2) Computational Geometry computes these relations, by deciding zero & sign of expressions
- Exact Geometric Computation (EGC):
  - an approach requiring error-free **zero & sign** computation

# Again, What is Geometry?

- Geometric vein permeates all of Mathematics
  - \* Number theory, algebraic & differential geometry, topology, probability, ...
- What is Geometry?
  - \* Euclid: Axiomatic Approach
  - \* Descartes: Algebraization of Geometry
  - \* Klein: Transformation Groups
  - \* Hilbert: Logical Foundations
  - \* Tarski: Elementary Geometry and Algebra
  - \* Erdos: Combinatorial Vein
- Computational Perspective:
  - \* (1) Geometry is comprised of discrete relations among geometric objects
  - \* (2) Computational Geometry computes these relations, by deciding zero & sign of expressions
- Exact Geometric Computation (EGC):
  - an approach requiring error-free zero & sign computation

# Robust Geometric Computation

- Widespread numerical nonrobustness issues in computational science and engineering
  - \* Computer Science's dirty secret...
- Geometric Computation is inherently discontinuous



- Upshot: if we can efficiently solve the zero problem
  - \* ... then we can produce robust software

# Robust Geometric Computation

- Widespread numerical nonrobustness issues in computational science and engineering
  - \* Computer Science's dirty secret...

- Geometric Computation is inherently discontinuous



- Upshot: if we can efficiently solve the zero problem
  - \* ... then we can produce robust software

# Robust Geometric Computation

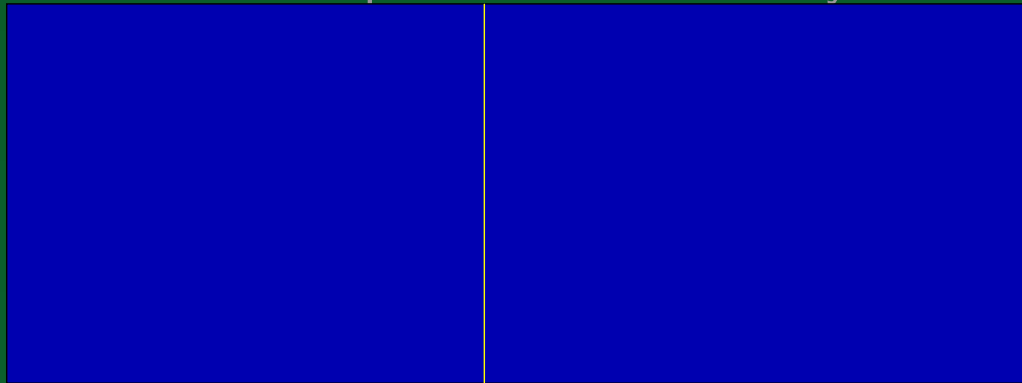
- Widespread numerical nonrobustness issues in computational science and engineering
  - \* Computer Science's dirty secret...
- Geometric Computation is inherently discontinuous



- Upshot: if we can efficiently solve the zero problem
  - \* ... then we can produce robust software

# Robust Geometric Computation

- Widespread numerical nonrobustness issues in computational science and engineering
  - \* Computer Science's dirty secret...
- Geometric Computation is inherently discontinuous



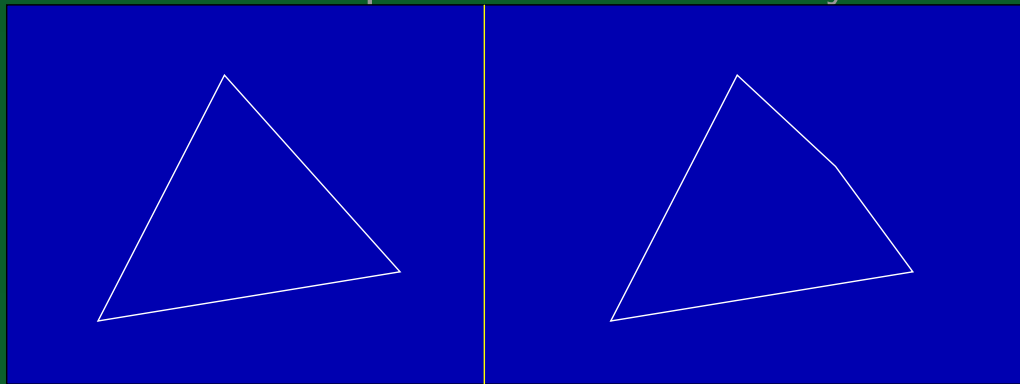
- Upshot: if we can efficiently solve the zero problem
  - \* ... then we can produce robust software



# Robust Geometric Computation

- Widespread numerical nonrobustness issues in computational science and engineering
  - \* Computer Science's dirty secret...

- Geometric Computation is inherently discontinuous



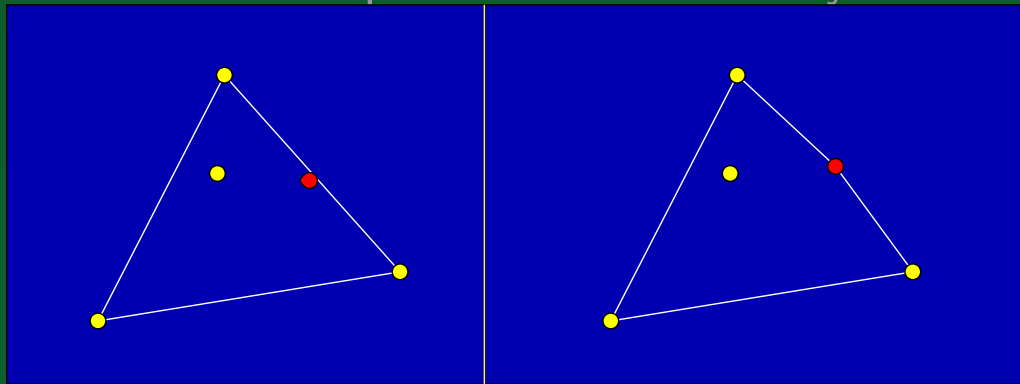
- Upshot: if we can efficiently solve the zero problem
  - \* ... then we can produce robust software

# Robust Geometric Computation

22

- Widespread numerical nonrobustness issues in computational science and engineering
  - \* Computer Science's dirty secret...

- Geometric Computation is inherently discontinuous



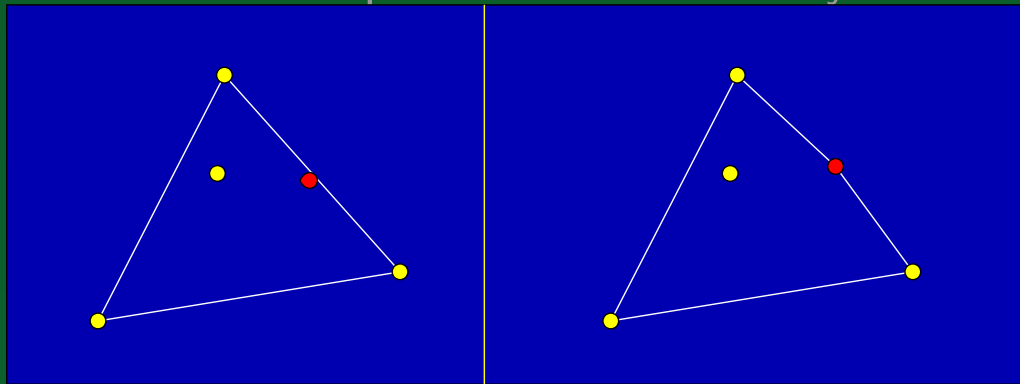
- Why EGC?
  - \* (1) Inconsistencies cause nonrobustness
  - \* (2) EGC ensures exact, hence consistent, geometry
  - \* (3) Most successful of known approaches
- Upshot: if we can efficiently solve the zero problem
  - \* ... then we can produce robust software

# Robust Geometric Computation

22

- Widespread numerical nonrobustness issues in computational science and engineering
  - \* Computer Science's dirty secret...

- Geometric Computation is inherently discontinuous



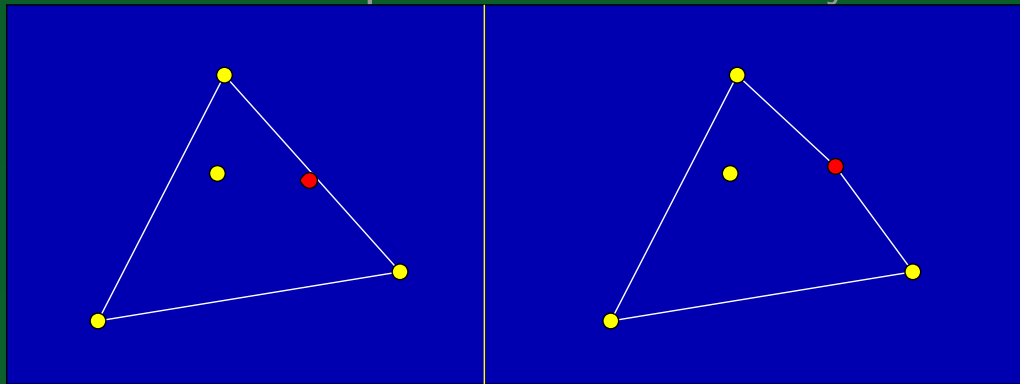
- Why EGC?
  - \* (1) Inconsistencies cause nonrobustness
  - \* (2) EGC ensures exact, hence consistent, geometry
  - \* (3) Most successful of known approaches
- Upshot: if we can efficiently solve the zero problem
  - \* ... then we can produce robust software

# Robust Geometric Computation

22

- Widespread numerical nonrobustness issues in computational science and engineering
  - \* Computer Science's dirty secret...

- Geometric Computation is inherently discontinuous



- Why EGC?
  - \* (1) Inconsistencies cause nonrobustness
  - \* (2) EGC ensures exact, hence consistent, geometry
  - \* (3) Most successful of known approaches
- Upshot: if we can efficiently solve the zero problem
  - \* ... then we can produce robust software

# Other Applications of Zero Problems

- Surface-Surface Intersection (SSI) (Farouki)
  - \* “The single greatest cause of poor reliability of CAD systems is lack of topologically consistent surface intersection algorithms” .
  - \* – consensus, Workshop on Math. Foundations of CAD, MSRI, Berkeley (1999)
- Computer-Aided Theorem Proving
  - \* E.g., Kepler’s conjecture (T. Hale)
- Automated Theorem Proving (D.M.Wang)
  - \* Randomized Testing, Proving by Example, etc
- Table Maker’s Dilemma (J-M.Muller)
- Test Suites for Statistical Software (B.D.McCullough)
- Guaranteed Precision Arithmetic software
  - \* E.g., Core Library, LEDA

# Other Applications of Zero Problems

- Surface-Surface Intersection (SSI) (Farouki)
  - \* “The single greatest cause of poor reliability of CAD systems is lack of topologically consistent surface intersection algorithms”.
  - \* – consensus, Workshop on Math. Foundations of CAD, MSRI, Berkeley (1999)
- Computer-Aided Theorem Proving
  - \* E.g., Kepler’s conjecture (T. Hale)
- Automated Theorem Proving (D.M.Wang)
  - \* Randomized Testing, Proving by Example, etc
- Table Maker’s Dilemma (J-M.Muller)
- Test Suites for Statistical Software (B.D.McCullough)
- Guaranteed Precision Arithmetic software
  - \* E.g., Core Library, LEDA

# Other Applications of Zero Problems

- Surface-Surface Intersection (SSI) (Farouki)
  - \* “The single greatest cause of poor reliability of CAD systems is lack of topologically consistent surface intersection algorithms” .
  - \* – consensus, Workshop on Math. Foundations of CAD, MSRI, Berkeley (1999)
- Computer-Aided Theorem Proving
  - \* E.g., Kepler’s conjecture (T. Hale)
- Automated Theorem Proving (D.M.Wang)
  - \* **Randomized Testing, Proving by Example, etc**
- Table Maker’s Dilemma (J-M.Muller)
- Test Suites for Statistical Software (B.D.McCullough)
- Guaranteed Precision Arithmetic software
  - \* E.g., Core Library, LEDA

# Other Applications of Zero Problems

- Surface-Surface Intersection (SSI) (Farouki)
  - \* “The single greatest cause of poor reliability of CAD systems is lack of topologically consistent surface intersection algorithms”.
  - \* – consensus, Workshop on Math. Foundations of CAD, MSRI, Berkeley (1999)
- Computer-Aided Theorem Proving
  - \* E.g., Kepler’s conjecture (T. Hale)
- Automated Theorem Proving (D.M.Wang)
  - \* Randomized Testing, Proving by Example, etc
- Table Maker’s Dilemma (J-M.Muller)
- Test Suites for Statistical Software (B.D.McCullough)
- Guaranteed Precision Arithmetic software
  - \* E.g., Core Library, LEDA



# Other Applications of Zero Problems

- Surface-Surface Intersection (SSI) (Farouki)
  - \* “The single greatest cause of poor reliability of CAD systems is lack of topologically consistent surface intersection algorithms”.
  - \* – consensus, Workshop on Math. Foundations of CAD, MSRI, Berkeley (1999)
- Computer-Aided Theorem Proving
  - \* E.g., Kepler’s conjecture (T. Hale)
- Automated Theorem Proving (D.M.Wang)
  - \* Randomized Testing, Proving by Example, etc
- Table Maker’s Dilemma (J-M.Muller)
- Test Suites for Statistical Software (B.D.McCullough)
- Guaranteed Precision Arithmetic software
  - \* E.g., Core Library, LEDA

# Other Applications of Zero Problems

- Surface-Surface Intersection (SSI) (Farouki)
  - \* “The single greatest cause of poor reliability of CAD systems is lack of topologically consistent surface intersection algorithms” .
  - \* – consensus, Workshop on Math. Foundations of CAD, MSRI, Berkeley (1999)
- Computer-Aided Theorem Proving
  - \* E.g., Kepler’s conjecture (T. Hale)
- Automated Theorem Proving (D.M.Wang)
  - \* Randomized Testing, Proving by Example, etc
- Table Maker’s Dilemma (J-M.Muller)
- Test Suites for Statistical Software (B.D.McCullough)
- Guaranteed Precision Arithmetic software
  - \* E.g., Core Library, LEDA

# Other Applications of Zero Problems

- Surface-Surface Intersection (SSI) (Farouki)
  - \* “The single greatest cause of poor reliability of CAD systems is lack of topologically consistent surface intersection algorithms”.
  - \* – consensus, Workshop on Math. Foundations of CAD, MSRI, Berkeley (1999)
- Computer-Aided Theorem Proving
  - \* E.g., Kepler’s conjecture (T. Hale)
- Automated Theorem Proving (D.M.Wang)
  - \* Randomized Testing, Proving by Example, etc
- Table Maker’s Dilemma (J-M.Muller)
- Test Suites for Statistical Software (B.D.McCullough)
- Guaranteed Precision Arithmetic software
  - \* E.g., Core Library, LEDA

# PART III.

## On Zero Bounds and Adaptivity

*“It can be of no practical use to know that  $\pi$  is irrational, but if we can know, it surely would be intolerable not to know.”*

— E.C. Titchmarsh

# The Numerical Halting Problem

- To decide if  $e = 0$ , we compute approximations
  - \*  $e_1, e_2, e_3, \dots$
  - \* where  $e = e_n \pm 2^{-n}$
- OUTPUT “ $e \neq 0$ ” when  $|e_n| > 2^{-n}$
- What if  $e = 0$ ?
- The ZERO PROBLEM is the continuous analogue of the HALTING problem
  - \* — like the halting problem, ZERO is “semi-decidable”
- The HALTING problem is complete for *Prec*  
We will show the continuous analogue.

# The Numerical Halting Problem

- To decide if  $e = 0$ , we compute approximations
  - \*  $e_1, e_2, e_3, \dots$
  - \* where  $e = e_n \pm 2^{-n}$
- OUTPUT “ $e \neq 0$ ” when  $|e_n| > 2^{-n}$
- What if  $e = 0$ ?
- The ZERO PROBLEM is the continuous analogue of the HALTING problem
  - \* — like the halting problem, ZERO is “semi-decidable”
- The HALTING problem is complete for *Prec*  
We will show the continuous analogue.

# The Numerical Halting Problem

- To decide if  $e = 0$ , we compute approximations
  - \*  $e_1, e_2, e_3, \dots$
  - \* where  $e = e_n \pm 2^{-n}$
- OUTPUT “ $e \neq 0$ ” when  $|e_n| > 2^{-n}$
- What if  $e = 0$ ?
- The ZERO PROBLEM is the continuous analogue of the HALTING problem
  - \* — like the halting problem, ZERO is “semi-decidable”
- The HALTING problem is complete for *Prec*  
We will show the continuous analogue.

# The Numerical Halting Problem

- To decide if  $e = 0$ , we compute approximations
  - \*  $e_1, e_2, e_3, \dots$
  - \* where  $e = e_n \pm 2^{-n}$
- OUTPUT “ $e \neq 0$ ” when  $|e_n| > 2^{-n}$
- What if  $e = 0$ ?
- The ZERO PROBLEM is the continuous analogue of the HALTING problem
  - \* — like the halting problem, ZERO is “semi-decidable”
- The HALTING problem is complete for *Prec*  
We will show the continuous analogue.



# The Numerical Halting Problem

- To decide if  $e = 0$ , we compute approximations
  - \*  $e_1, e_2, e_3, \dots$
  - \* where  $e = e_n \pm 2^{-n}$
- OUTPUT “ $e \neq 0$ ” when  $|e_n| > 2^{-n}$
- What if  $e = 0$ ?
- The ZERO PROBLEM is the continuous analogue of the HALTING problem
  - \* — like the halting problem, ZERO is “semi-decidable”
- The HALTING problem is complete for *Prec*  
We will show the continuous analogue.

# The Numerical Halting Problem

- To decide if  $e = 0$ , we compute approximations
  - \*  $e_1, e_2, e_3, \dots$
  - \* where  $e = e_n \pm 2^{-n}$
- OUTPUT “ $e \neq 0$ ” when  $|e_n| > 2^{-n}$
- What if  $e = 0$ ?
- The ZERO PROBLEM is the continuous analogue of the HALTING problem
  - \* — like the halting problem, ZERO is “semi-decidable”
- The HALTING problem is complete for *Prec*  
We will show the continuous analogue.

## Two Ways to to Use Zero Bounds

- Suppose we have  $B(e) > 0$  such that:
  - \* if  $\text{Val}(e) \neq 0$ , then  $|\text{Val}(e)| > B(e)$
  - \*  $B$  is a (conditional) zero bound function
  
- METHOD 1: Compute approximation  $\tilde{e}$  for  $e$  so that  $|\tilde{e} - e| < B(e)/2$ 
  - \* If  $|\tilde{e}| \geq B(e)$ , then OUTPUT “ $e \neq 0$ ”
  - \* Otherwise, OUTPUT “ $e = 0$ ”
  
- METHOD 2: Compute sequence  $(e_1, e_2, e_3, \dots)$  as before
  - \* If  $|e_n| > 2^{-n}$ , OUTPUT “ $e \neq 0$ ”
  - \* If  $2^{-n} < \frac{B(e)}{2}$ , OUTPUT “ $e = 0$ ”
  
- Semi-adaptivity of METHOD 2
  - \* if  $x \neq y$ , the complexity depends on  $-\log_2 |x - y|$

## Two Ways to to Use Zero Bounds

- Suppose we have  $B(e) > 0$  such that:
  - \* if  $\text{Val}(e) \neq 0$ , then  $|\text{Val}(e)| > B(e)$
  - \*  $B$  is a (conditional) zero bound function
  
- METHOD 1: Compute approximation  $\tilde{e}$  for  $e$  so that  $|\tilde{e} - e| < B(e)/2$ 
  - \* If  $|\tilde{e}| \geq B(e)$ , then OUTPUT “ $e \neq 0$ ”
  - \* Otherwise, OUTPUT “ $e = 0$ ”
  
- METHOD 2: Compute sequence  $(e_1, e_2, e_3, \dots)$  as before
  - \* If  $|e_n| > 2^{-n}$ , OUTPUT “ $e \neq 0$ ”
  - \* If  $2^{-n} < \frac{B(e)}{2}$ , OUTPUT “ $e = 0$ ”
  
- Semi-adaptivity of METHOD 2
  - \* if  $x \neq y$ , the complexity depends on  $-\log_2 |x - y|$

## Two Ways to to Use Zero Bounds

- Suppose we have  $B(e) > 0$  such that:
  - \* if  $\text{Val}(e) \neq 0$ , then  $|\text{Val}(e)| > B(e)$
  - \*  $B$  is a (conditional) zero bound function
  
- METHOD 1: Compute approximation  $\tilde{e}$  for  $e$  so that  $|\tilde{e} - e| < B(e)/2$ 
  - \* If  $|\tilde{e}| \geq B(e)$ , then OUTPUT “ $e \neq 0$ ”
  - \* Otherwise, OUTPUT “ $e = 0$ ”
  
- METHOD 2: Compute sequence  $(e_1, e_2, e_3, \dots)$  as before
  - \* If  $|e_n| > 2^{-n}$ , OUTPUT “ $e \neq 0$ ”
  - \* If  $2^{-n} < \frac{B(e)}{2}$ , OUTPUT “ $e = 0$ ”
  
- Semi-adaptivity of METHOD 2
  - \* if  $x \neq y$ , the complexity depends on  $-\log_2 |x - y|$

## Two Ways to to Use Zero Bounds

- Suppose we have  $B(e) > 0$  such that:
  - \* if  $\text{Val}(e) \neq 0$ , then  $|\text{Val}(e)| > B(e)$
  - \*  $B$  is a (conditional) zero bound function
  
- METHOD 1: Compute approximation  $\tilde{e}$  for  $e$  so that  $|\tilde{e} - e| < B(e)/2$ 
  - \* If  $|\tilde{e}| \geq B(e)$ , then OUTPUT “ $e \neq 0$ ”
  - \* Otherwise, OUTPUT “ $e = 0$ ”
  
- METHOD 2: Compute sequence  $(e_1, e_2, e_3, \dots)$  as before
  - \* If  $|e_n| > 2^{-n}$ , OUTPUT “ $e \neq 0$ ”
  - \* If  $2^{-n} < \frac{B(e)}{2}$ , OUTPUT “ $e = 0$ ”
  
- Semi-adaptivity of METHOD 2
  - \* if  $x \neq y$ , the complexity depends on  $-\log_2 |x - y|$

## Two Ways to to Use Zero Bounds

- Suppose we have  $B(e) > 0$  such that:
  - \* if  $\text{Val}(e) \neq 0$ , then  $|\text{Val}(e)| > B(e)$
  - \*  $B$  is a (conditional) zero bound function
  
- METHOD 1: Compute approximation  $\tilde{e}$  for  $e$  so that  $|\tilde{e} - e| < B(e)/2$ 
  - \* If  $|\tilde{e}| \geq B(e)$ , then OUTPUT “ $e \neq 0$ ”
  - \* Otherwise, OUTPUT “ $e = 0$ ”
  
- METHOD 2: Compute sequence  $(e_1, e_2, e_3, \dots)$  as before
  - \* If  $|e_n| > 2^{-n}$ , OUTPUT “ $e \neq 0$ ”
  - \* If  $2^{-n} < \frac{B(e)}{2}$ , OUTPUT “ $e = 0$ ”
  
- Semi-adaptivity of METHOD 2
  - \* if  $x \neq y$ , the complexity depends on  $-\log_2 |x - y|$

# Why Adaptivity Algorithms

- Adaptive Complexity
  - \* (1) The numerical method (B) is adaptive...
  - \* (2) Algebraic methods are inherently non-adaptive, too inefficient
  - \* (3) Only algebraic information: Zero Bounds
- Other advantages:
  - \* simpler algorithms
  - \* exploits geometry
  - \* algorithms are independent of bounds
- General trend in computer algebra
  - \* (1) PROBLEM: most adaptive algorithms are incomplete
  - \* (2) E.g., no known adaptive complete algorithm for topological analysis of curve
  - \* (3) ONE SOLUTION: Hybrid algorithms
- OPEN PROBLEM:  
Construct Complete and Fully Adaptive Algorithms for basic problems
  - \* E.g., topological analysis of curves



# Why Adaptivity Algorithms

- Adaptive Complexity
  - \* (1) The numerical method (B) is adaptive...
  - \* (2) Algebraic methods are inherently non-adaptive, too inefficient
  - \* (3) Only algebraic information: Zero Bounds
- Other advantages:
  - \* simpler algorithms
  - \* exploits geometry
  - \* algorithms are independent of bounds
- General trend in computer algebra
  - \* (1) PROBLEM: most adaptive algorithms are incomplete
  - \* (2) E.g., no known adaptive complete algorithm for topological analysis of curve
  - \* (3) ONE SOLUTION: Hybrid algorithms
- OPEN PROBLEM:  
Construct Complete and Fully Adaptive Algorithms for basic problems
  - \* E.g., topological analysis of curves

# Why Adaptivity Algorithms

- Adaptive Complexity
  - \* (1) The numerical method (B) is adaptive...
  - \* (2) Algebraic methods are inherently non-adaptive, too inefficient
  - \* (3) Only algebraic information: Zero Bounds
- Other advantages:
  - \* simpler algorithms
  - \* exploits geometry
  - \* algorithms are independent of bounds
- General trend in computer algebra
  - \* (1) **PROBLEM**: most adaptive algorithms are incomplete
  - \* (2) E.g., no known adaptive complete algorithm for topological analysis of curve
  - \* (3) **ONE SOLUTION**: Hybrid algorithms
- **OPEN PROBLEM**:  
Construct Complete and Fully Adaptive Algorithms for basic problems
  - \* E.g., topological analysis of curves

# Why Adaptivity Algorithms

- Adaptive Complexity
  - \* (1) The numerical method (B) is adaptive...
  - \* (2) Algebraic methods are inherently non-adaptive, too inefficient
  - \* (3) Only algebraic information: Zero Bounds
- Other advantages:
  - \* simpler algorithms
  - \* exploits geometry
  - \* algorithms are independent of bounds
- General trend in computer algebra
  - \* (1) PROBLEM: most adaptive algorithms are incomplete
  - \* (2) E.g., no known adaptive complete algorithm for topological analysis of curve
  - \* (3) ONE SOLUTION: Hybrid algorithms
- OPEN PROBLEM:  
Construct Complete and Fully Adaptive Algorithms for basic problems
  - \* E.g., topological analysis of curves

# Why Adaptivity Algorithms

- Adaptive Complexity
  - \* (1) The numerical method (B) is adaptive...
  - \* (2) Algebraic methods are inherently non-adaptive, too inefficient
  - \* (3) Only algebraic information: Zero Bounds
- Other advantages:
  - \* simpler algorithms
  - \* exploits geometry
  - \* algorithms are independent of bounds
- General trend in computer algebra
  - \* (1) PROBLEM: most adaptive algorithms are incomplete
  - \* (2) E.g., no known adaptive complete algorithm for topological analysis of curve
  - \* (3) ONE SOLUTION: Hybrid algorithms
- OPEN PROBLEM:  
Construct Complete and Fully Adaptive Algorithms for basic problems
  - \* E.g., topological analysis of curves

# Where do we get Zero Bounds?

- Classical Root Bounds
  - \* Constructive Root Bounds

- BFMS Bound

$$* B(e) = \frac{1}{L(e)U(e)^{D(e)^2-1}}$$

	$e$	$U(e)$	$L(e)$
1.	rational $a/b$	$a$	$b$
2.	$e_1 \pm e_2$	$U(e_1)L(e_2) + L(e_1)U(e_2)$	$L(e_1)L(e_2)$
3.	$e_1 \times e_2$	$U(e_1)U(e_2)$	$L(e_1)L(e_2)$
4.	$e_1 \div e_2$	$U(e_1)L(e_2)$	$L(e_1)U(e_2)$
5.	$\sqrt[k]{e_1}$	$\sqrt[k]{U(e_1)}$	$\sqrt[k]{L(e_1)}$

- Other Constructive Bounds:  
BFMSS, Mahler Measure, Li-Yap, k-ary, etc

# Where do we get Zero Bounds?

- Classical Root Bounds
  - \* Constructive Root Bounds

- BFMS Bound

$$* B(e) = \frac{1}{L(e)U(e)D(e)^{2-1}}$$

	$e$	$U(e)$	$L(e)$
1.	rational $a/b$	$a$	$b$
2.	$e_1 \pm e_2$	$U(e_1)L(e_2) + L(e_1)U(e_2)$	$L(e_1)L(e_2)$
3.	$e_1 \times e_2$	$U(e_1)U(e_2)$	$L(e_1)L(e_2)$
4.	$e_1 \div e_2$	$U(e_1)L(e_2)$	$L(e_1)U(e_2)$
5.	$\sqrt[k]{e_1}$	$\sqrt[k]{U(e_1)}$	$\sqrt[k]{L(e_1)}$

- Other Constructive Bounds:  
BFMSS, Mahler Measure, Li-Yap, k-ary, etc

# Where do we get Zero Bounds?

- Classical Root Bounds
  - \* Constructive Root Bounds

- BFMS Bound

$$* B(e) = \frac{1}{L(e)U(e)^{D(e)^2-1}}$$

	$e$	$U(e)$	$L(e)$
1.	rational $a/b$	$a$	$b$
2.	$e_1 \pm e_2$	$U(e_1)L(e_2) + L(e_1)U(e_2)$	$L(e_1)L(e_2)$
3.	$e_1 \times e_2$	$U(e_1)U(e_2)$	$L(e_1)L(e_2)$
4.	$e_1 \div e_2$	$U(e_1)L(e_2)$	$L(e_1)U(e_2)$
5.	$\sqrt[k]{e_1}$	$\sqrt[k]{U(e_1)}$	$\sqrt[k]{L(e_1)}$

- Other Constructive Bounds:  
BFMSS, Mahler Measure, Li-Yap, k-ary, etc

# Where do we get Zero Bounds?

- Classical Root Bounds
  - \* Constructive Root Bounds

- BFMS Bound

$$* B(e) = \frac{1}{L(e)U(e)^{D(e)^2-1}}$$

	$e$	$U(e)$	$L(e)$
1.	rational $a/b$	$a$	$b$
2.	$e_1 \pm e_2$	$U(e_1)L(e_2) + L(e_1)U(e_2)$	$L(e_1)L(e_2)$
3.	$e_1 \times e_2$	$U(e_1)U(e_2)$	$L(e_1)L(e_2)$
4.	$e_1 \div e_2$	$U(e_1)L(e_2)$	$L(e_1)U(e_2)$
5.	$\sqrt[k]{e_1}$	$\sqrt[k]{U(e_1)}$	$\sqrt[k]{L(e_1)}$

- Other Constructive Bounds:  
BFMSS, Mahler Measure, Li-Yap, k-ary, etc



# The Central Algorithm

- Guaranteed precision computation
  - \* Viewed as a generalization of EGC
- Basic idea:
 

Each Operator  $f \in \Omega$  is computed by  $\mathcal{R}_f$

  - \* Precision (a priori error) is driven down
  - \* Approximation (+ a posteriori error) is propagated up
  - \*  $\mathcal{R}_f$  is applied to approximations, and propagated up (unless)

	Precision in $x$	Precision in $y$	Operation Precision
$z = x \pm y$	$p + 2$	$p + 2$	$\infty$
$z = x \times y$	$p + 2 + \mu^+(y)$	$p + 1 + \mu^+(x)$	$\infty$
$z = x/y$	$p + 2 - \mu^-(y)$	$\max\{1 - \mu^-(y),$ $p + 2 - 2\mu^-(y) + 2\mu^+(x)\}$	$p + 1$ $p + 1$
$z = \sqrt{x}$	$\max\{p + 1, 1 - \mu^-(x)/2$		$p + 1$
$z = \exp x$	$\max\{1, p + 2 + 2\mu^+(x) + 1\}$		$p + 1$
$z = \log x$	$\max\{1 - \mu^-(x), p + 2 - \mu^-(x)\}$		$p + 1$

# The Central Algorithm

- Guaranteed precision computation
  - \* Viewed as a generalization of EGC
- Basic idea:
 

Each Operator  $f \in \Omega$  is computed by  $\mathcal{R}_f$

  - \* Precision (a priori error) is driven down
  - \* Approximation (+ a posteriori error) is propagated up
  - \*  $\mathcal{R}_f$  is applied to approximations, and propagated up (unless)

	Precision in $x$	Precision in $y$	Operation Precision
$z = x \pm y$	$p + 2$	$p + 2$	$\infty$
$z = x \times y$	$p + 2 + \mu^+(y)$	$p + 1 + \mu^+(x)$	$\infty$
$z = x/y$	$p + 2 - \mu^-(y)$	$\max\{1 - \mu^-(y),$ $p + 2 - 2\mu^-(y) + 2\mu^+(x)\}$	$p + 1$ $p + 1$
$z = \sqrt{x}$	$\max\{p + 1, 1 - \mu^-(x)/2$		$p + 1$
$z = \exp x$	$\max\{1, p + 2 + 2\mu^+(x) + 1\}$		$p + 1$
$z = \log x$	$\max\{1 - \mu^-(x), p + 2 - \mu^-(x)\}$		$p + 1$

# The Central Algorithm

- Guaranteed precision computation
  - \* Viewed as a generalization of EGC
- Basic idea:
 

Each Operator  $f \in \Omega$  is computed by  $\mathcal{R}_f$

  - \* Precision (a priori error) is driven down
  - \* Approximation (+ a posteriori error) is propagated up
  - \*  $\mathcal{R}_f$  is applied to approximations, and propagated up (unless)

	Precision in $x$	Precision in $y$	Operation Precision
$z = x \pm y$	$p + 2$	$p + 2$	$\infty$
$z = x \times y$	$p + 2 + \mu^+(y)$	$p + 1 + \mu^+(x)$	$\infty$
$z = x/y$	$p + 2 - \mu^-(y)$	$\max\{1 - \mu^-(y),$ $p + 2 - 2\mu^-(y) + 2\mu^+(x)\}$	$p + 1$ $p + 1$
$z = \sqrt{x}$	$\max\{p + 1, 1 - \mu^-(x)/2\}$		$p + 1$
$z = \exp x$	$\max\{1, p + 2 + 2\mu^+(x) + 1\}$		$p + 1$
$z = \log x$	$\max\{1 - \mu^-(x), p + 2 - \mu^-(x)\}$		$p + 1$

# The Central Algorithm

- Guaranteed precision computation
  - \* Viewed as a generalization of EGC
- Basic idea:
 

Each Operator  $f \in \Omega$  is computed by  $\mathcal{R}_f$

  - \* Precision (a priori error) is driven down
  - \* Approximation (+ a posteriori error) is propagated up
  - \*  $\mathcal{R}_f$  is applied to approximations, and propagated up (unless)

	Precision in $x$	Precision in $y$	Operation Precision
$z = x \pm y$	$p + 2$	$p + 2$	$\infty$
$z = x \times y$	$p + 2 + \mu^+(y)$	$p + 1 + \mu^+(x)$	$\infty$
$z = x/y$	$p + 2 - \mu^-(y)$	$\max\{1 - \mu^-(y),$ $p + 2 - 2\mu^-(y) + 2\mu^+(x)\}$	$p + 1$ $p + 1$
$z = \sqrt{x}$	$\max\{p + 1, 1 - \mu^-(x)/2\}$		$p + 1$
$z = \exp x$	$\max\{1, p + 2 + 2\mu^+(x) + 1\}$		$p + 1$
$z = \log x$	$\max\{1 - \mu^-(x), p + 2 - \mu^-(x)\}$		$p + 1$

- Cast of Characters

Three Musketeers and a friend:

- \* (A)  $\alpha(e; p)$  — absolute approximation
- \* (M)  $\mu^+(e)$  — upper bound on  $\mu(e) = \lg |\text{Val}(e)|$
- \* (S)  $\text{sign}(e)$  — sign of  $\text{Val}(e)$
- \* (B)  $\beta(e)$  — root bound function

- OPEN PROBLEM: What is the optimal algorithm for evaluation?

- Cast of Characters

Three Musketeers and a friend:

- \* (A)  $\alpha(e; p)$  — absolute approximation
- \* (M)  $\mu^+(e)$  — upper bound on  $\mu(e) = \lg |\text{Val}(e)|$
- \* (S)  $\text{sign}(e)$  — sign of  $\text{Val}(e)$
- \* (B)  $\beta(e)$  — root bound function

- OPEN PROBLEM: What is the optimal algorithm for evaluation?



# Transcendental Number Theory

- Another conditional zero bound:

$$B(e) = 1$$

- \* (Fundamental Theorem of TNT)

- How do you prove that  $\beta = \sum_{n=0}^{\infty} (2/3)^{2^n}$  is irrational?

- Outline of TNT (D.Masser)

- \* (AP) Construction of Auxilliary Polynomial
  - \* (UB) Obtaining an upper bound
  - \* (LB) Obtaining a lower bound
  - \* (NV) Proving non-vanishing

- Let  $\beta_n = \sum_{n=0}^n (2/3)^{2^n}$ ,  $R_n = \beta - \beta_n$

- \* (AP) Let  $P(X, Y) = 2XY^2 + 4XY - 3Y^2 + X - Y$

- \* (UB)  $|\alpha_n| < (1/10)^{2^{n+1}}$  where  $|P((2/3)^{2^{n+1}}, R_n)| < (1/10)^{2^{n+1}}$

- \* (LB) If  $\beta = r/s$  and  $\alpha_n \neq 0$ , then  $|P((2/3)^{2^{n+1}}, R_n)| \geq s^{-2}(1/9)^{2^{n+1}}$

- \* (NV)  $\alpha_n \neq 0$

- From irrationality/transcendence to transcendence measures



# Transcendental Number Theory

- Another conditional zero bound:

$$B(e) = 1$$

- \* (Fundamental Theorem of TNT)

- How do you prove that  $\beta = \sum_{n=0}^{\infty} (2/3)^{2^n}$  is irrational?

- Outline of TNT (D.Masser)

- \* (AP) Construction of Auxilliary Polynomial
- \* (UB) Obtaining an upper bound
- \* (LB) Obtaining a lower bound
- \* (NV) Proving non-vanishing

- Let  $\beta_n = \sum_{n=0}^n (2/3)^{2^n}$ ,  $R_n = \beta - \beta_n$ 
  - \* (AP) Let  $P(X, Y) = 2XY^2 + 4XY - 3Y^2 + X - Y$
  - \* (UB)  $|\alpha_n| < (1/10)^{2^{n+1}}$  where  $|P((2/3)^{2^{n+1}}, R_n)| < (1/10)^{2^{n+1}}$
  - \* (LB) If  $\beta = r/s$  and  $\alpha_n \neq 0$ , then  $|P((2/3)^{2^{n+1}}, R_n)| \geq s^{-2}(1/9)^{2^{n+1}}$
  - \* (NV)  $\alpha_n \neq 0$

- From irrationality/transcendence to transcendence measures

# Transcendental Number Theory

- Another conditional zero bound:

$$B(e) = 1$$

- \* (Fundamental Theorem of TNT)

- How do you prove that  $\beta = \sum_{n=0}^{\infty} (2/3)^{2^n}$  is irrational?

- Outline of TNT (D.Masser)

- \* (AP) Construction of Auxilliary Polynomial
- \* (UB) Obtaining an upper bound
- \* (LB) Obtaining a lower bound
- \* (NV) Proving non-vanishing

- Let  $\beta_n = \sum_{n=0}^n (2/3)^{2^n}$ ,  $R_n = \beta - \beta_n$ 
  - \* (AP) Let  $P(X, Y) = 2XY^2 + 4XY - 3Y^2 + X - Y$
  - \* (UB)  $|\alpha_n| < (1/10)^{2^{n+1}}$  where  $|P((2/3)^{2^{n+1}}, R_n)| < (1/10)^{2^{n+1}}$
  - \* (LB) If  $\beta = r/s$  and  $\alpha_n \neq 0$ , then  $|P((2/3)^{2^{n+1}}, R_n)| \geq s^{-2}(1/9)^{2^{n+1}}$
  - \* (NV)  $\alpha_n \neq 0$

- From irrationality/transcendence to transcendence measures

# Transcendental Number Theory

- Another conditional zero bound:  
 $B(e) = 1$ 
  - \* (Fundamental Theorem of TNT)
- How do you prove that  $\beta = \sum_{n=0}^{\infty} (2/3)^{2^n}$  is irrational?
- Outline of TNT (D.Masser)
  - \* (AP) Construction of Auxilliary Polynomial
  - \* (UB) Obtaining an upper bound
  - \* (LB) Obtaining a lower bound
  - \* (NV) Proving non-vanishing
- Let  $\beta_n = \sum_{n=0}^n (2/3)^{2^n}$ ,  $R_n = \beta - \beta_n$ 
  - \* (AP) Let  $P(X, Y) = 2XY^2 + 4XY - 3Y^2 + X - Y$
  - \* (UB)  $|\alpha_n| < (1/10)^{2^{n+1}}$  where  $|P((2/3)^{2^{n+1}}, R_n)| < (1/10)^{2^{n+1}}$
  - \* (LB) If  $\beta = r/s$  and  $\alpha_n \neq 0$ , then  $|P((2/3)^{2^{n+1}}, R_n)| \geq s^{-2}(1/9)^{2^{n+1}}$
  - \* (NV)  $\alpha_n \neq 0$
- From irrationality/transcendence to transcendence measures

# Transcendental Number Theory

- Another conditional zero bound:

$$B(e) = 1$$

- \* (Fundamental Theorem of TNT)

- How do you prove that  $\beta = \sum_{n=0}^{\infty} (2/3)^{2^n}$  is irrational?

- Outline of TNT (D.Masser)

- \* (AP) Construction of Auxilliary Polynomial
- \* (UB) Obtaining an upper bound
- \* (LB) Obtaining a lower bound
- \* (NV) Proving non-vanishing

- Let  $\beta_n = \sum_{n=0}^n (2/3)^{2^n}$ ,  $R_n = \beta - \beta_n$ 
  - \* (AP) Let  $P(X, Y) = 2XY^2 + 4XY - 3Y^2 + X - Y$
  - \* (UB)  $|\alpha_n| < (1/10)^{2^{n+1}}$  where  $|P((2/3)^{2^{n+1}}, R_n)| < (1/10)^{2^{n+1}}$
  - \* (LB) If  $\beta = r/s$  and  $\alpha_n \neq 0$ , then  $|P((2/3)^{2^{n+1}}, R_n)| \geq s^{-2}(1/9)^{2^{n+1}}$
  - \* (NV)  $\alpha_n \neq 0$

- From irrationality/transcendence to transcendence measures

# Transcendental Number Theory

- Another conditional zero bound:

$$B(e) = 1$$

- \* (Fundamental Theorem of TNT)

- How do you prove that  $\beta = \sum_{n=0}^{\infty} (2/3)^{2^n}$  is irrational?

- Outline of TNT (D.Masser)

- \* (AP) Construction of Auxilliary Polynomial
- \* (UB) Obtaining an upper bound
- \* (LB) Obtaining a lower bound
- \* (NV) Proving non-vanishing

- Let  $\beta_n = \sum_{n=0}^n (2/3)^{2^n}$ ,  $R_n = \beta - \beta_n$

- \* (AP) Let  $P(X, Y) = 2XY^2 + 4XY - 3Y^2 + X - Y$

- \* (UB)  $|\alpha_n| < (1/10)^{2^{n+1}}$  where  $|P((2/3)^{2^{n+1}}, R_n)| < (1/10)^{2^{n+1}}$

- \* (LB) If  $\beta = r/s$  and  $\alpha_n \neq 0$ , then  $|P((2/3)^{2^{n+1}}, R_n)| \geq s^{-2}(1/9)^{2^{n+1}}$

- \* (NV)  $\alpha_n \neq 0$

- From irrationality/transcendence to transcendence measures



# PART IV

## On Foundations of Real Computation

*“There is a substantial conflict between theoretical computer science and numerical analysis. ... The conflict has at its roots another age-old conflict, that between the continuous and the discrete. ”*

— Blum, Cucker, Shub, Smale (1996, Manifesto)

# Two Approaches to Real Computation

- Analytic Approach
  - \* Turing(1936), Grzegorzczuk(1955), Weihrauch, Ko, etc
  - \* Real numbers represented by rapidly converging Cauchy sequences
  - \* Extend Turing machines to compute with infinite input/output sequences
- Algebraic Approach
  - \* Blum-Shub-Smale (BSS) model, Real RAMs
  - \* Real numbers directly represented as atomic objects
  - \* Real numbers are compared without error
  - \* Algebraic operators are carried out without error



# Two Approaches to Real Computation

- Analytic Approach
  - \* Turing(1936), Grzegorzczuk(1955), Weihrauch, Ko, etc
  - \* Real numbers represented by rapidly converging Cauchy sequences
  - \* Extend Turing machines to compute with infinite input/output sequences
- Algebraic Approach
  - \* Blum-Shub-Smale (BSS) model, Real RAMs
  - \* Real numbers directly represented as atomic objects
  - \* Real numbers are compared without error
  - \* Algebraic operators are carried out without error

# Two Approaches to Real Computation

- Analytic Approach
  - \* Turing(1936), Grzegorzczuk(1955), Weihrauch, Ko, etc
  - \* Real numbers represented by rapidly converging Cauchy sequences
  - \* Extend Turing machines to compute with infinite input/output sequences
  
- Algebraic Approach
  - \* Blum-Shub-Smale (BSS) model, Real RAMs
  - \* Real numbers directly represented as atomic objects
  - \* Real numbers are compared without error
  - \* Algebraic operators are carried out without error

# Where is the Zero Problem?

- Zero Problem is
  - \* undecidable in Analytic Approach
  - \* trivial in Algebraic Approach
  - \* more nuanced in our Approximation Approach
- Other Issues
  - \* (Analytic) Only continuous functions are computable (no geometry!)
  - \* (Algebraic) Exponential function is not computable
  - \* (Approximation) Composition is not automatic
- THEOREM: There exists  $\mathcal{R}$ -approximable  $f, g$  such that their composition  $f \circ g$  is not  $\mathcal{A}$ -approximable

# Where is the Zero Problem?

- Zero Problem is
  - \* undecidable in Analytic Approach
  - \* trivial in Algebraic Approach
  - \* more nuanced in our Approximation Approach
- Other Issues
  - \* (Analytic) Only continuous functions are computable (no geometry!)
  - \* (Algebraic) Exponential function is not computable
  - \* (Approximation) Composition is not automatic
- THEOREM: There exists  $\mathcal{R}$ -approximable  $f, g$  such that their composition  $f \circ g$  is not  $\mathcal{A}$ -approximable

# Where is the Zero Problem?

- Zero Problem is
  - \* undecidable in Analytic Approach
  - \* trivial in Algebraic Approach
  - \* more nuanced in our Approximation Approach
- Other Issues
  - \* (Analytic) Only continuous functions are computable (no geometry!)
  - \* (Algebraic) Exponential function is not computable
  - \* (Approximation) Composition is not automatic
- THEOREM: There exists  $\mathcal{R}$ -approximable  $f, g$  such that their composition  $f \circ g$  is not  $\mathcal{A}$ -approximable

# Where is the Zero Problem?

- Zero Problem is
  - \* undecidable in Analytic Approach
  - \* trivial in Algebraic Approach
  - \* more nuanced in our Approximation Approach
- Other Issues
  - \* (Analytic) Only continuous functions are computable (no geometry!)
  - \* (Algebraic) Exponential function is not computable
  - \* (Approximation) Composition is not automatic
- THEOREM: There exists  $\mathcal{R}$ -approximable  $f, g$  such that their composition  $f \circ g$  is not  $\mathcal{A}$ -approximable

# How We Solve Numerical Problems

- E.g., Solving a PDE model, A numerical optimization, etc
- STEP A:
  - \* Design an ideal Algorithm A
  - \* Assume certain operations such as  $\pm$ ,  $\times$ ,  $\exp()$
  - \* Show that problem is solvable by Algorithm A
- STEP B:
  - \* Implement Algorithm A as a Numerical Program B
  - \* Account for numerical representation, errors, convergence, etc
  - \* Specify the “correctness” criteria

# How We Solve Numerical Problems

- E.g., Solving a PDE model, A numerical optimization, etc
- STEP A:
  - \* Design an ideal Algorithm A
  - \* Assume certain operations such as  $\pm$ ,  $\times$ ,  $\exp()$
  - \* Show that problem is solvable by Algorithm A
- STEP B:
  - \* Implement Algorithm A as a Numerical Program B
  - \* Account for numerical representation, errors, convergence, etc
  - \* Specify the “correctness” criteria



# How We Solve Numerical Problems

- E.g., Solving a PDE model, A numerical optimization, etc
- STEP A:
  - \* Design an ideal Algorithm A
  - \* Assume certain operations such as  $\pm$ ,  $\times$ ,  $\exp()$
  - \* Show that problem is solvable by Algorithm A
- STEP B:
  - \* Implement Algorithm A as a Numerical Program B
  - \* Account for numerical representation, errors, convergence, etc
  - \* Specify the “correctness” criteria

# How We Solve Numerical Problems

- E.g., Solving a PDE model, A numerical optimization, etc
  
- STEP A:
  - \* Design an ideal Algorithm A
  - \* Assume certain operations such as  $\pm$ ,  $\times$ ,  $\exp()$
  - \* Show that problem is solvable by Algorithm A
  
- STEP B:
  - \* Implement Algorithm A as a Numerical Program B
  - \* Account for numerical representation, errors, convergence, etc
  - \* Specify the “correctness” criteria

# A New Synthesis

- Step A:
  - \* Algorithm A belongs to an Algebraic Model (e.g., BSS Model)
  - \* Basis  $\Omega = \{\pm, \times, \exp(), \dots\}$
- Step B:
  - \* Program B belongs to some numerical model (Turing or Pointer Model)
- Critical Question:
  - \* Can Algorithm A be implemented by some Program B?
- Validity:
  - Even numerical analysis books proceed to Step B via Step A
- The algebraic and numerical models play complementary roles!

# A New Synthesis

- Step A:
  - \* Algorithm A belongs to an Algebraic Model (e.g., BSS Model)
  - \* Basis  $\Omega = \{\pm, \times, \exp(), \dots\}$
- Step B:
  - \* Program B belongs to some numerical model (Turing or Pointer Model)
- Critical Question:
  - \* Can Algorithm A be implemented by some Program B?
- Validity:

Even numerical analysis books proceed to Step B via Step A
- The algebraic and numerical models play complementary roles!

# A New Synthesis

- Step A:
  - \* Algorithm A belongs to an Algebraic Model (e.g., BSS Model)
  - \* Basis  $\Omega = \{\pm, \times, \exp(), \dots\}$
- Step B:
  - \* Program B belongs to some numerical model (Turing or Pointer Model)
- Critical Question:
  - \* Can Algorithm A be implemented by some Program B?
- Validity:

Even numerical analysis books proceed to Step B via Step A
- The algebraic and numerical models play complementary roles!

# A New Synthesis

- Step A:
  - \* Algorithm A belongs to an Algebraic Model (e.g., BSS Model)
  - \* Basis  $\Omega = \{\pm, \times, \exp(), \dots\}$
- Step B:
  - \* Program B belongs to some numerical model (Turing or Pointer Model)
- Critical Question:
  - \* Can Algorithm A be implemented by some Program B?
- Validity:

Even numerical analysis books proceed to Step B via Step A
- The algebraic and numerical models play complementary roles!

# A New Synthesis

- Step A:
  - \* Algorithm A belongs to an Algebraic Model (e.g., BSS Model)
  - \* Basis  $\Omega = \{\pm, \times, \exp(), \dots\}$
- Step B:
  - \* Program B belongs to some numerical model (Turing or Pointer Model)
- Critical Question:
  - \* Can Algorithm A be implemented by some Program B?
- Validity:

Even numerical analysis books proceed to Step B via Step A
- The algebraic and numerical models play complementary roles!

# A New Synthesis

- Step A:
  - \* Algorithm A belongs to an Algebraic Model (e.g., BSS Model)
  - \* Basis  $\Omega = \{\pm, \times, \exp(), \dots\}$
- Step B:
  - \* Program B belongs to some numerical model (Turing or Pointer Model)
- Critical Question:
  - \* Can Algorithm A be implemented by some Program B?
- Validity:

Even numerical analysis books proceed to Step B via Step A
- The algebraic and numerical models play complementary roles!



# Transfer Theorem

- Algebraic Pointer Model:
  - \* Schönhage's Pointer Machines, augmented with algebraic operations from  $\Omega$
- Numerical Pointer Model:
  - \* like the Algebraic Pointer Model, but replace  $f \in \Omega$  by  $\mathcal{R}_f$
- Approximation in the EGC sense
  - \* Combinatorially exact, but numerically approximate
- THEOREM:  
The following are equivalent:
  - \* (I)  $\text{Val}_\Omega$  is  $\mathcal{R}$ -approximable over  $\Omega$
  - \* (II) For all problems  $F$ , if  $F$  is  $\Omega$ -computable (in algebraic model) then  $F$  is  $\Omega$ -approximable (in numerical model).
- OPEN PROBLEM: Explore other transfer theorems

# Transfer Theorem

- Algebraic Pointer Model:
  - \* Schönhage's Pointer Machines, augmented with algebraic operations from  $\Omega$
- Numerical Pointer Model:
  - \* like the Algebraic Pointer Model, but replace  $f \in \Omega$  by  $\mathcal{R}_f$
- Approximation in the EGC sense
  - \* Combinatorially exact, but numerically approximate
- THEOREM:  
The following are equivalent:
  - \* (I)  $\text{Val}_\Omega$  is  $\mathcal{R}$ -approximable over  $\Omega$
  - \* (II) For all problems  $F$ , if  $F$  is  $\Omega$ -computable (in algebraic model) then  $F$  is  $\Omega$ -approximable (in numerical model).
- OPEN PROBLEM: Explore other transfer theorems

# Transfer Theorem

- Algebraic Pointer Model:
  - \* Schönhage's Pointer Machines, augmented with algebraic operations from  $\Omega$
- Numerical Pointer Model:
  - \* like the Algebraic Pointer Model, but replace  $f \in \Omega$  by  $\mathcal{R}_f$
- Approximation in the EGC sense
  - \* Combinatorially exact, but numerically approximate
- THEOREM:  
The following are equivalent:
  - \* (I)  $\text{Val}_\Omega$  is  $\mathcal{R}$ -approximable over  $\Omega$
  - \* (II) For all problems  $F$ , if  $F$  is  $\Omega$ -computable (in algebraic model) then  $F$  is  $\Omega$ -approximable (in numerical model).
- OPEN PROBLEM: Explore other transfer theorems

# Transfer Theorem

- Algebraic Pointer Model:
  - \* Schönhage's Pointer Machines, augmented with algebraic operations from  $\Omega$
- Numerical Pointer Model:
  - \* like the Algebraic Pointer Model, but replace  $f \in \Omega$  by  $\mathcal{R}_f$
- Approximation in the EGC sense
  - \* Combinatorially exact, but numerically approximate
- THEOREM:  
The following are equivalent:
  - \* (I)  $\text{Val}_\Omega$  is  $\mathcal{R}$ -approximable over  $\Omega$
  - \* (II) For all problems  $F$ , if  $F$  is  $\Omega$ -computable (in algebraic model) then  $F$  is  $\Omega$ -approximable (in numerical model).
- OPEN PROBLEM: Explore other transfer theorems

# Transfer Theorem

- Algebraic Pointer Model:
  - \* Schönhage's Pointer Machines, augmented with algebraic operations from  $\Omega$
- Numerical Pointer Model:
  - \* like the Algebraic Pointer Model, but replace  $f \in \Omega$  by  $\mathcal{R}_f$
- Approximation in the EGC sense
  - \* Combinatorially exact, but numerically approximate
- THEOREM:  
The following are equivalent:
  - \* (I)  $\text{Val}_\Omega$  is  $\mathcal{R}$ -approximable over  $\Omega$
  - \* (II) For all problems  $F$ , if  $F$  is  $\Omega$ -computable (in algebraic model) then  $F$  is  $\Omega$ -approximable (in numerical model).
- OPEN PROBLEM: Explore other transfer theorems

# Transfer Theorem

- Algebraic Pointer Model:
  - \* Schönhage's Pointer Machines, augmented with algebraic operations from  $\Omega$
- Numerical Pointer Model:
  - \* like the Algebraic Pointer Model, but replace  $f \in \Omega$  by  $\mathcal{R}_f$
- Approximation in the EGC sense
  - \* Combinatorially exact, but numerically approximate
- THEOREM:  
The following are equivalent:
  - \* (I)  $\text{Val}_\Omega$  is  $\mathcal{R}$ -approximable over  $\Omega$
  - \* (II) For all problems  $F$ , if  $F$  is  $\Omega$ -computable (in algebraic model) then  $F$  is  $\Omega$ -approximable (in numerical model).
- OPEN PROBLEM: Explore other transfer theorems

# Connection to Analytic School

- THEOREM: Let  $f : S \subseteq \mathbb{R} \dashrightarrow \mathbb{R}$  and  $S$  regular.  
The following are equivalent:
  - \*  $f$  is computable (in analytic model)
  - \*  $f$  is partially  $\mathcal{A}$ -approximable and has a recursively enumerable modulus cover
- Thus, approximability of  $f$  gives up precisely one thing: continuity of  $f$

# Connection to Analytic School

- THEOREM: Let  $f : S \subseteq \mathbb{R} \rightarrow \mathbb{R}$  and  $S$  regular.  
The following are equivalent:
  - \*  $f$  is computable (in analytic model)
  - \*  $f$  is partially  $\mathcal{A}$ -approximable and has a recursively enumerable modulus cover
- Thus, approximability of  $f$  gives up precisely one thing: continuity of  $f$



# Connection to Analytic School

- THEOREM: Let  $f : S \subseteq \mathbb{R} \rightarrow \mathbb{R}$  and  $S$  regular.  
The following are equivalent:
  - \*  $f$  is computable (in analytic model)
  - \*  $f$  is partially  $\mathcal{A}$ -approximable and has a recursively enumerable modulus cover
- Thus, approximability of  $f$  gives up precisely one thing: continuity of  $f$

# PART V

## CONCLUSION

*“Why is  $\alpha_n \neq 0$ ? This innocent question will become more and more of a nuisance until it almost takes over the subject.”*

— David Masser (2000)

# Other Zero Problems (This Workshop)

- Functional Zero Problems (Van Der Hoeven)
  - \* — in our framework, we may admit variables in  $\Omega$
  - \* — cf. Constant Zero Problem, Singularity Theory, Canonicity
- Zero Finding (Myunghi, Sharma, Shub, Verschelde)
  - \* — viewed as the inverse of Constant Zero Problem
  - \* — condition numbers as counter part to zero bounds
- Applications (Choi, Farouki, Wang)
  - \* — positive results from TNT!
  - \* — approaches to non-robustness in CAD
  - \* — theorem proving
- Elementary Problems (Choi, Richardson)
  - \* — frontier of solvable Zero Problems
- Theory of Real Computation (Ziegler)
  - \* — rich interplay possible between algebraic and analytic views
  - \* — complexity of zero problems and real approximation

# Other Zero Problems (This Workshop)

- Functional Zero Problems (Van Der Hoeven)
  - \* — in our framework, we may admit variables in  $\Omega$
  - \* — cf. Constant Zero Problem, Singularity Theory, Canonicity
- Zero Finding (Myunghi, Sharma, Shub, Verschelde)
  - \* — viewed as the inverse of Constant Zero Problem
  - \* — condition numbers as counter part to zero bounds
- Applications (Choi, Farouki, Wang)
  - \* — positive results from TNT!
  - \* — approaches to non-robustness in CAD
  - \* — theorem proving
- Elementary Problems (Choi, Richardson)
  - \* — frontier of solvable Zero Problems
- Theory of Real Computation (Ziegler)
  - \* — rich interplay possible between algebraic and analytic views
  - \* — complexity of zero problems and real approximation

# Other Zero Problems (This Workshop)

- Functional Zero Problems (Van Der Hoeven)
  - \* — in our framework, we may admit variables in  $\Omega$
  - \* — cf. Constant Zero Problem, Singularity Theory, Canonicity
- Zero Finding (Myunghi, Sharma, Shub, Verschelde)
  - \* — viewed as the inverse of Constant Zero Problem
  - \* — condition numbers as counter part to zero bounds
- Applications (Choi, Farouki, Wang)
  - \* — positive results from TNT!
  - \* — approaches to non-robustness in CAD
  - \* — theorem proving
- Elementary Problems (Choi, Richardson)
  - \* — frontier of solvable Zero Problems
- Theory of Real Computation (Ziegler)
  - \* — rich interplay possible between algebraic and analytic views
  - \* — complexity of zero problems and real approximation

# Other Zero Problems (This Workshop)

- Functional Zero Problems (Van Der Hoeven)
  - \* — in our framework, we may admit variables in  $\Omega$
  - \* — cf. Constant Zero Problem, Singularity Theory, Canonicity
- Zero Finding (Myunghi, Sharma, Shub, Verschelde)
  - \* — viewed as the inverse of Constant Zero Problem
  - \* — condition numbers as counter part to zero bounds
- Applications (Choi, Farouki, Wang)
  - \* — positive results from TNT!
  - \* — approaches to non-robustness in CAD
  - \* — theorem proving
- Elementary Problems (Choi, Richardson)
  - \* — **frontier of solvable Zero Problems**
- Theory of Real Computation (Ziegler)
  - \* — rich interplay possible between algebraic and analytic views
  - \* — complexity of zero problems and real approximation

# Other Zero Problems (This Workshop)

- Functional Zero Problems (Van Der Hoeven)
  - \* — in our framework, we may admit variables in  $\Omega$
  - \* — cf. Constant Zero Problem, Singularity Theory, Canonicity
- Zero Finding (Myunghi, Sharma, Shub, Verschelde)
  - \* — viewed as the inverse of Constant Zero Problem
  - \* — condition numbers as counter part to zero bounds
- Applications (Choi, Farouki, Wang)
  - \* — positive results from TNT!
  - \* — approaches to non-robustness in CAD
  - \* — theorem proving
- Elementary Problems (Choi, Richardson)
  - \* — frontier of solvable Zero Problems
- Theory of Real Computation (Ziegler)
  - \* — rich interplay possible between algebraic and analytic views
  - \* — complexity of zero problems and real approximation

# Other Zero Problems (This Workshop)

- Functional Zero Problems (Van Der Hoeven)
  - \* — in our framework, we may admit variables in  $\Omega$
  - \* — cf. Constant Zero Problem, Singularity Theory, Canonicity
- Zero Finding (Myunghi, Sharma, Shub, Verschelde)
  - \* — viewed as the inverse of Constant Zero Problem
  - \* — condition numbers as counter part to zero bounds
- Applications (Choi, Farouki, Wang)
  - \* — positive results from TNT!
  - \* — approaches to non-robustness in CAD
  - \* — theorem proving
- Elementary Problems (Choi, Richardson)
  - \* — frontier of solvable Zero Problems
- Theory of Real Computation (Ziegler)
  - \* — rich interplay possible between algebraic and analytic views
  - \* — complexity of zero problems and real approximation





END OF TALK

# Thanks for Listening!

“A rapacious monster lurks within every computer,  
and it dines exclusively on accurate digits.”

– B.D. McCullough (2000)

- Software and Papers can be downloaded from <http://cs.nyu.edu/exact/>
- ★ Core Library Software
- ★ “Theory of Real Computation according to EGC”,  
Issue from Dagstuhl Workshop on Real Computation, 2006.
- ★ “Complete Subdivision Algorithms 1: Intersecting Bezier Curves”,  
SoCG’06.
- ★ “Complete Numerical Isolation of Real Zeros in General Triangular  
Systems”,

(with Jin-San Cheng and Xiao-shan Gao, ISSAC'07

45

END OF TALK

# Schönhage's Pointer Model

- 
- $\Delta$ -graph  $G$ : a labelled digraph
  - \* Pointer machine transforms  $G$  by re-directing edges

Type	Name	Instruction	Meaning
(i)	Node Assignment	$w \leftarrow w'$	$[w]_{G'} = [w']_G$
(ii)	Node Creation	$w \leftarrow \mathbf{new}$	$[w]_{G'}$ is new
(iii)	Node Comparison	<b>if</b> $w \equiv w'$ <b>goto</b> $L$	$G' = G$
(iv)	Halt and Output	<b>HALT</b> ( $w$ )	Output $G w$
(v)	Value Comparison	<b>if</b> $(w \circ w')$ <b>goto</b> $L$ where $\circ \in \{=, <, \leq\}$	Compare $Val_G(w) \circ Val_G(w')$
(vi)	Value Assignment	$w := f(w_1, \dots, w_m)$ where $f \in \Omega$ and $w, w_i \in \Delta^*$	$Val_{G'}(w) = f(Val_G(w_1), \dots, Val_G(w_m))$

# Schönhage's Pointer Model

- 
- $\Delta$ -graph  $G$ : a labelled digraph
  - \* Pointer machine transforms  $G$  by re-directing edges

Type	Name	Instruction	Meaning
(i)	Node Assignment	$w \leftarrow w'$	$[w]_{G'} = [w']_G$
(ii)	Node Creation	$w \leftarrow \mathbf{new}$	$[w]_{G'}$ is new
(iii)	Node Comparison	<b>if</b> $w \equiv w'$ <b>goto</b> $L$	$G' = G$
(iv)	Halt and Output	<b>HALT</b> ( $w$ )	Output $G w$
(v)	Value Comparison	<b>if</b> $(w \circ w')$ <b>goto</b> $L$ where $\circ \in \{=, <, \leq\}$	Compare $Val_G(w) \circ Val_G(w')$
(vi)	Value Assignment	$w := f(w_1, \dots, w_m)$ where $f \in \Omega$ and $w, w_i \in \Delta^*$	$Val_{G'}(w) = f(Val_G(w_1), \dots, Val_G(w_m))$