

New Systems, New Tools, and New Math

Richard Shoup

Boundary Institute
Los Altos, California, USA

rshoup@boundary.org

Overview

- Computer engineering hasn't changed much in 50 years...
 - What has and hasn't changed
 - Why the field is a mess and the tools are awful
- Parallelism, reconfigurable computing, multicore, etc.
- Formalization needed throughout
- New math foundations needed
- Examples of new directions, ideas



50 Years vs 5B Years

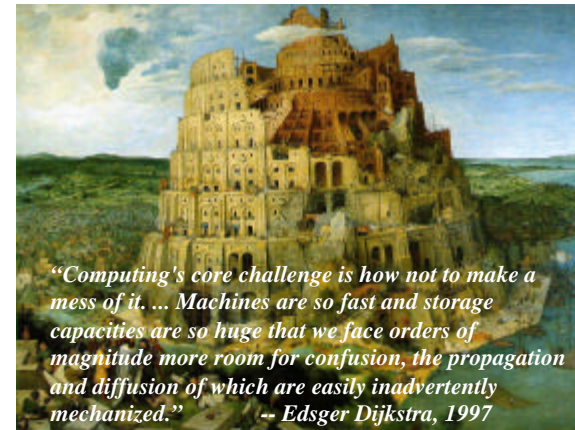
Computing Systems

serial, sequential
digital, discrete
unidirectional, functional
singular, brittle
centralized, concentrated
linear, controlled
homogeneous
mechanical, intermittent
difficult, arcane
designed, crafted

Natural Systems

highly parallel, concurrent
analog, continuous
omnidirectional, relational
redundant, resilient
distributed, pervasive
non-linear, chaotic
heterogeneous
alive, continuous
accessible, familiar
self-organizing, evolving

Which is a better match to human beings and their needs?



Computing - 50 years

- Digital technology = $F(1, \dots)$
- Not much change except size/speed/cost
 - Exceptions: user interface, networking, apps
- Architecture: von Neumann (Eckert)
- Theory: logic, set theory, Turing-Church
- Programming model: algorithms
 - “A step-by-step problem-solving procedure”
 - Sequential processors, instructions
 - Unfolding/mapping computation in Time



Muhammad al-Khwarizmi
~780-850 AD
al-Khwarizmi **P** “algorithm”
Hisab al-jabr **P** “algebra”

50 Years of Computing - What *Has* Changed

- Faster, cheaper, smaller !!
 - But nearing practical limitations; Power \propto Freq³
 - What are we going to do with 1B or 10B transistors?
 - Use Area to get more Performance
- Better user interface (2D+, interactive)
- Communications, networks
- Much broader range of applications
- ...

50 Years of Computing - What Has *Not* Changed

- Von Neumann model: stored program, serial execution
- Centralized memory & processing, buses
- Synchronous, global clocking
- Awful hwd & sw design tools, bad bugs
- Linear text-based programming
- Compilers, debuggers, crashes
- Chip & protocol chasing
- Little or no formal verification or security



*Kids running down a hill,
out of control*

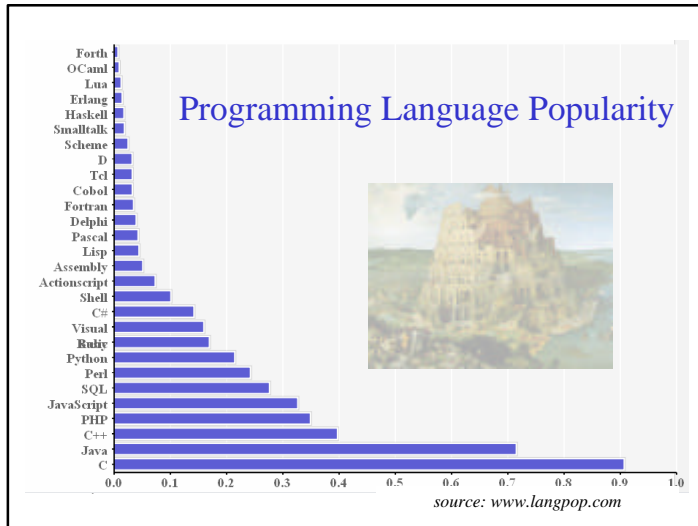
Programming – 1962 vs 2002

```
Read(a, b, c);
if a=0 then
  begin
    Roots := 1;
    x1real := -c/b;
    x2real := 0;
    goto zeros;
  end;
Roots := 2;
disc := b2 - 4*a*c;
if disc < 0 then
  begin
    x1real := x2real := -b / (2*a);
    x1imag := sqrt(-disc) / (2*a);
    x2imag := -x1imag;
    goto output;
  end;
sqrt := sqrt(disc);
x1real := (-b + sqrt) / (2*a);
x2real := (-b - sqrt) / (2*a);
zeros:
x1imag := x2imag := 0;
Print(a, b, c, x1real, x1imag, x2real,
      x2imag, Roots);
```

Source: *A Guide to Algol Programming*,
D. McCracken, 1962

```
printf("Enter value for a : ");
scanf("%lf", &a);
if ( a == 0.0 ) { break; }
printf("Enter value for b : ");
scanf("%lf", &b);
printf("Enter value for c : ");
scanf("%lf", &c);
aux = bb * bb - 4 * aa * cc;
if ( aux > 0.0 )
{
  aux = sqrt( aux );
  root1 = ( -bb + aux ) / ( 2 * aa );
  root2 = ( -bb - aux ) / ( 2 * aa );
  printf("\nRoots are %.4lf and %.4lf\n", root1, root2 );
}
else if ( aux == 0.0 )
{
  root1 = -bb / ( 2 * aa );
  printf("\nRoot is %.4lf\n", root1 );
}
else
{
  printf("\nNo real roots found\n");
}
```

Source: On-line C++ programming course,
xx University, 2002



Heterogeneous Design Hierarchy

↑	user interface, application	⇒	<i>menus, dialogs, commands</i>
	very-high-level language	⇒	<i>threads, app objects, users</i>
	high-level language	⇒	<i>processes, threads</i>
	system libraries, OS calls	⇒	<i>procedures, synchronization</i>
	low-level language	⇒	<i>data structures, subroutines</i>
	assembly/machine code	⇒	<i>storage allocation</i>
	microcode, FSMs	⇒	<i>register allocation</i>
	gates, flip-flops, registers	⇒	<i>timing, power</i>
	transistors, gates	⇒	<i>layout, timing</i>
	silicon structures	⇒	<i>electrical properties, yield</i>
	semiconductor physics	⇒	<i>processes, chemistry, optics</i>

Heterogeneous Data Hierarchy

- Network, domain
 - Computer
 - Disk, partition
 - Folder
 - File
 - Byte, word
 - Bit
- } different types

... but object-oriented languages & systems mitigate

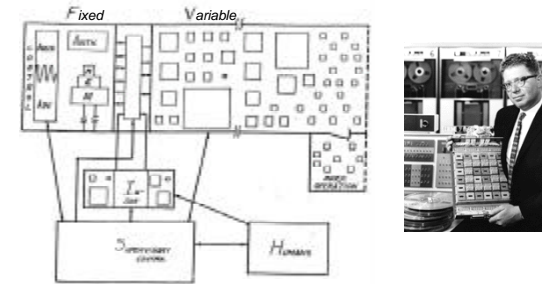
What's the Problem?

1. Free lunch (I, Moore's Law and implications)
 - Chasing the latest chip, board, protocol, device, application, etc
 - Improvement in performance, cost, etc every 2 years
 - Decline in innovation in design and programming
2. Lack of deep fundamental principles and mathematical foundations in computer engineering
 - Limits applications, reliability, and security of complex, distributed, highly-parallel information systems *today*.

Parallel, Reconfigurable, Multicore, etc

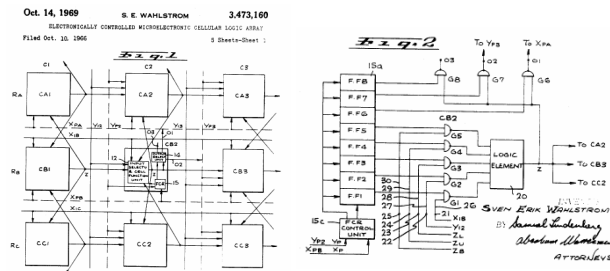
- Need for more computation (100× .. 1M× .. 1B× ...)
 - Simulation, vision, AI, data mining, genomics, etc, etc
 - ⇒ Parallelism - custom pipelines/structures
 - Unfold computation in *space* (data parallel), or *time* (pipeline parallel)
 - ⇒ Parallel hardware is more specific (lower generality)
 - ⇒ *Reconfigurable* parallel hardware (FPGAs to multicores)
- ... *But our programming paradigm and nearly all our tools are still conceptualized for fixed, sequential hardware.*
- ⇒ *A challenge, a motivation, ... and an opportunity*

Early Reconfigurable Computing



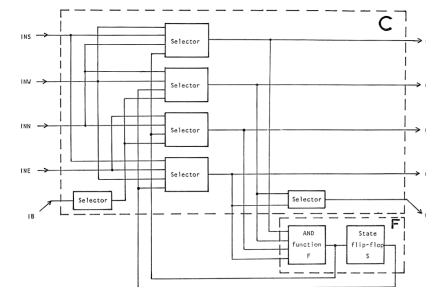
Gerald Estrin et al, UCLA – “Fixed+Variable Computer”, early 1960s

Early FPGA



Sven Wahlstrom, SRI – “Cellular Logic Array”, 1966-9

Early FPGA



Richard Shoup, CMU – “Programmable Cellular Logic Arrays”, 1970

Things I would not have believed back then ...

- Reconfigurable hardware (FPGAs) won't be commercialized for 15 years, won't become popular for 25, and isn't fully exploited 35 yrs later.
- Engineering will be done informally, without mathematical grounding, flying by the seat of the pants, in most hardware & software design.
- CPUs will be built on <45 nm elements, run at >5 GHz, yet will use essentially the same architecture and programming model.
- Computing will still be serial by default, parallel a special case.
- Industry will continue to deploy, and consumers will accept, operating systems and applications with significant known bugs.
- Industry and academia will deploy networking on a global scale with little or no security ... and leave it that way for many years.
- Everything will still run as "applications".
- Turn-on and boot will still take a long time.
- The word "crash" will still be in common use! ***

Progress in CS?

"Our basic ideas about what a computer is, what it does, and how it does it, have hardly changed for decades."

-- Paul Dourish, Where the Action Is, 2001

"Of all undergraduate CS subjects, theoretical computer science has changed the least over the decades."

-- SIGACT News, March 2004

Progress in Languages & Design?

"Modern programming languages are highly complex and are standardised by committees, ... the resulting languages will inevitably contain compromise, inconsistency, incompleteness and some bits that simply do not work as expected. ... failures occur with monotonous regularity and with a similar frequency to 20 years ago."

-- Les Hatton, IT Week, April 2006

"The introduction of the microprocessor in 1971 marked the beginning of a 30-year stall in design methods for electronic systems."

-- Nick Tredennick

"If a system has an extremely powerful debugger, it may be because the system needs an extremely powerful debugger." -- anonymous

Progress in AI?

"The question [Can machines think?]. ... I believe to be too meaningless to deserve discussion. Nevertheless, I believe that at the end of the century the use of words and general educated opinion will have altered so much that one will be able to speak of machines thinking without expecting to be contradicted."

-- Alan Turing, 1950

Progress in Reliability & Security?

- ... millions of credit card transactions stolen, compromised ...
- ... thousands of employee personnel files accessed ...
- ... virus shuts down thousands of PCs ...
- ... etc
- Two words: *Patch Tuesdays*
 - "Four bulletins for nine flaws ... considered light ..." - IW, July 2008
- Two more words: *DNS poisoning*
- ... etc

But, kudos to Microsoft for SLAM !

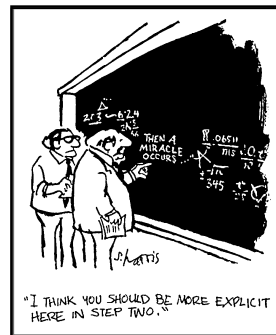
New Foundations

- Computer Sci & Eng need deeper math *foundations*.
- Emphasis on *structure* & relations, not algorithms.
- Single abstract *formal* representation/language.
- *Concurrent* by default, serial a special case.
- Design is *formal* from the beginning & throughout.
- Reconfigurable Computing remains an important challenge and opportunity for fundamental progress.

"We have to do it while it still seems crazy!" – Pierre St. Hilaire

-- New Math --

- Boundary Math
 - Logic with one symbol
- Complex Logics
 - Imaginary logic values
- Link Theory
 - Composite relations
- Constraint Graphs
 - Sequential inference



"Basic" ZFC Set Theory

- $\forall x \forall y [(x = y) \leftrightarrow \forall z (z \in x \leftrightarrow z \in y)]$ (*extensionality*)
- $\forall x \forall y \exists z (z = \{x, y\})$ (*pairing*)
- $\forall z \exists y (z \in y \leftrightarrow z \in x \wedge \phi(z))$ (*comprehension schema*)
- $\exists x (\{ \} \in x \wedge \forall y \in x (S(y) \in x))$ (*infinity*)
- $\forall x \exists y (z \in y \leftrightarrow z \subset x)$ (*power set*)
- $\exists x (x \in a \wedge x \cap a = \{ \})$ (*foundation*)
- $\forall w \exists s (s = \{y : \exists x \in w \wedge \phi(x, y)\})$ (*replacement schema*)
- $\forall X \exists f (\forall x \in X - \{ \} (f(x) \in x))$ (*choice*)

"Later generations will regard set theory as a disease from which one has recovered." -- Henri Poincaré

Logic with 1 Concept

~~F~~ ~~T~~ ~~~~~ ~~∧~~ ~~∨~~ ()

⇓

()

“Boundary Math”

(Laws of Form – George Spencer-Brown, 1969)
(Developed and applied by William Bricken, et al)

BMath - Notation

Boolean algebra

Boundary Math

F

T

a

$\sim a$

$a \vee b$

$a \vee b \vee c$

$a \wedge b$

$a \Rightarrow b$

()

a

(a)

a b

a b c

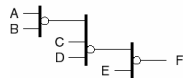
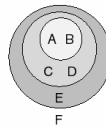
((a) (b))

(a) b

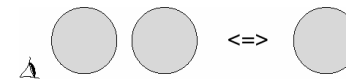
BMath - Operators vs Space

- “ $\dot{\cup}$ ” operator vs “in same space”
 $\sim(\sim(\sim(A \dot{\cup} B) \dot{\cup} C \dot{\cup} D) \dot{\cup} E) = F$
- Commutative? \otimes
 – no sequence, no linear order
- Associative? \otimes
 – no sequence, no linear order
- Arity? \otimes
 – arbitrary or none
- Relation vs Function
- Set Theory basis, etc

$(((A B) C D) E) F$



Distinctions - BMath Axioms



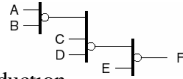
Outside 1st D - symmetrical, cardinality, bosons



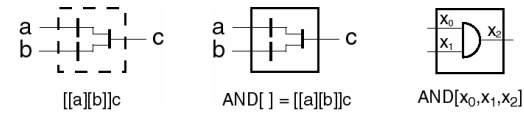
Inside 1st D - asymmetrical, ordinality, fermions

Boundary Math - Summary

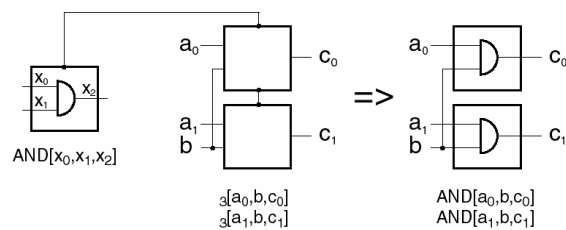
- Single concept: the *Distinction* or Difference
- “Simplicity Itself”, both object and operator
- Beneath/prior to standard logic, set theory
- Unified combinational arithmetic, algebra, calculus
- Interpretation for logic, circuits
 - Concise, efficient representation \Rightarrow ()
- Transformations, optimization \Rightarrow reduction
- A new way to approach the world ...
 - Distinctions, boundaries instead of objects



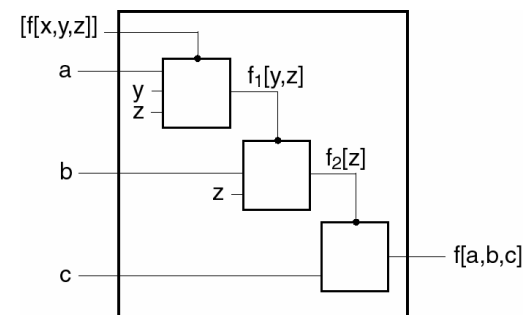
Encapsulation & Abstraction



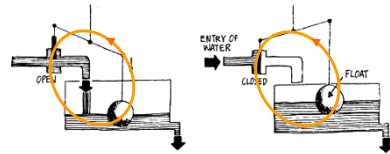
Instancing & Replication



Lambda Apply (Currying)



Self-Reference, Feedback, and Paradox

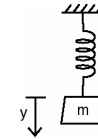


Negative feedback

"All Cretans are liars." -- Epimenides the Cretan
 "This sentence is False." -- anonymous
 "The set of all sets not containing themselves." -- Bertrand Russell
 "This statement is unprovable." -- Kurt Gödel
 "This program does not halt." -- Alan Turing

Classical Self-Reference (Feedback)

- Mechanical systems
- Control theory
- Systems theory
- Cybernetics
- etc ...



$$y = -\frac{m}{k} \frac{d^2 y}{dt^2}$$

$$= C_1 e^{+i\omega t} + C_2 e^{-i\omega t}$$

$$= C \sin(\omega t + \phi)$$

Negative feedback is commonly used in engineering,
 ... but paradox is forbidden in Logic and Set Theory

Imaginary Numbers & Logic Values

Complex Numbers

$$x = -1/x \quad (x^2 = -1)$$

$$x = \pm\sqrt{-1} = i \text{ or } -i$$

$$1, -1, i, -i \cdot$$

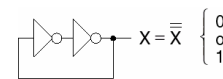
Complex Logic Values

$$x = \sim x$$

$$x = i \text{ or } -i$$

$$F, T, i, -i \otimes$$

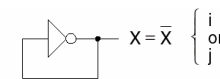
Self-Reference - the Origin of Time



X_{in} -----

X_{out} -----

Autonomy (Memory)

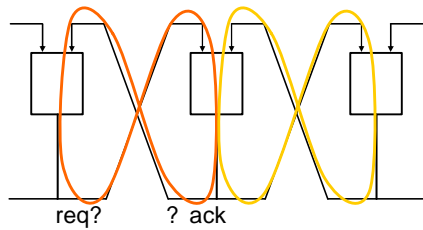


X_{in} [square wave]

X_{out} [square wave]

Antinomy/Paradox (Clock)

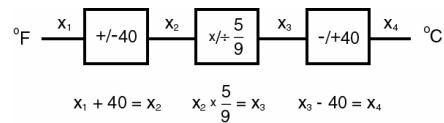
Imagineries in Async Pipelines?



Complex Logics - Summary

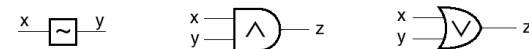
- 4-valued logic holds simple Paradox (F, T, i, -i)
 - Unify combinational and sequential domains
 - Time-Space trade-off: 4 timeless values \Leftrightarrow 2 values + time
 - Higher-order paradoxes, 2^{2^n} values \Leftrightarrow arbitrary waveforms
cf: Walsh, Hadamard, etc, Vuillemin: "On Circuits and Numbers"
- Paradoxes are common, unavoidable (e.g. Gödel), *useful*
 - Bridge from finite to infinite forms
 - Looping synchronous or async circuits in hardware or software
 - Inconsistent databases, expert systems, logical controls, etc.
 - Proof techniques, re-examine *reductio ad absurdum*, etc.

Relations vs Functions



Fahrenheit – Centigrade converter

NOT, AND, OR as Relations

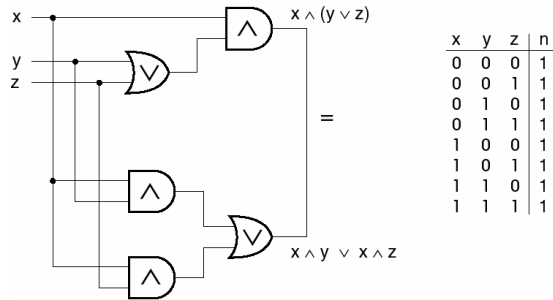


x	y	n
0	0	0
0	1	1
1	0	1
1	1	0

x	y	z	n _∧
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

x	y	z	n _∨
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Example: Distributive Law



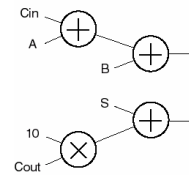
Example: Cryptarithm

SEND
+MORE

MONEY

$$\Rightarrow \begin{aligned} D + E &= Y + C_1 \cdot 10 \\ C_1 + N + R &= E + C_2 \cdot 10 \\ C_2 + E + O &= N + C_3 \cdot 10 \\ C_3 + S + M &= O + M \cdot 10 \end{aligned}$$

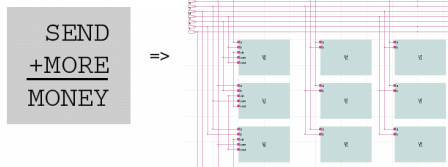
with $S, E, N, D, M, O, R, Y \in \{0-9\}$
uniquely, and $C_i \in \{0, 1\}$



$$C_{in} + A + B = S + C_{out} \cdot 10$$

for each column

Cryptarithm Solution



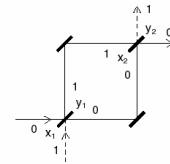
(4 tables, each 200 cases,
+ uniqueness tables)

S E N D M O R Y
9 5 6 7 1 0 8 2

9567
+1085

10652

Example: Quantum Interference

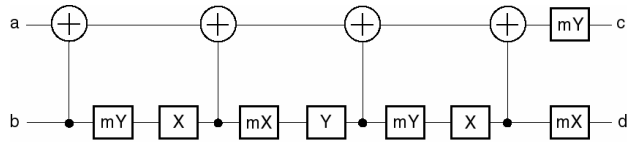


$$\begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \Rightarrow \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

x_1	y_1	π_1	x_2	y_2	π_2
0	0	1	0	0	1
0	1	1	0	1	-1
1	0	-1	1	0	1
1	1	1	1	1	1

$$\oplus \Rightarrow \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \Rightarrow \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

Example: Quantum Computation



x	y	n_y	n_{mY}	n_x	n_{mX}
0	0	1	1	1	1
0	1	1	-1	i	$-i$
1	0	-1	1	i	$-i$
1	1	1	1	1	1



x	y	z	n_{CNOT}
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Case Counts & Domains

all 1s : independent, uncorrelated

one 1 : selector, state, classical

more than one 1 : logic, relation, superposition

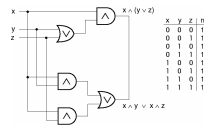
one or more >1 : statistics, multisets, don't cares

both + and - : quantum realm, interference

x	y	n_0	n_1	n_2	n_3	n_4
0	0	1	0	0	1	1
0	1	1	0	1	2	1
1	0	1	0	0	2	-1
1	1	1	1	1	1	1

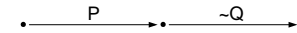
Link Theory - Summary

- Theory of *composite relations* (bidirectional constraints)
- *Case counts*, a set of *possibilities*
- *Symmetrical*, directionless, timeless
- Represent a wide range of problems in one paradigm
 - Logic, combinatorics, arithmetic, statistical, ... , Qcomputing
 - New basis for physics?
- General theory of *structure*
 - Quantum, molecular computing, etc
 - Evolutionary, self-organizing systems



(Tom Etter & Richard Shoup, Boundary Institute)

Sequential Constraints



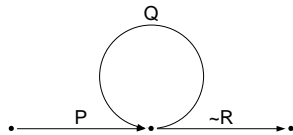
disallowed: [P] [~Q]

“Prohibited: P followed by NOT Q in the next state”

“If P, then Q in the next state”

“If NOT Q, then NOT P in the preceding state”

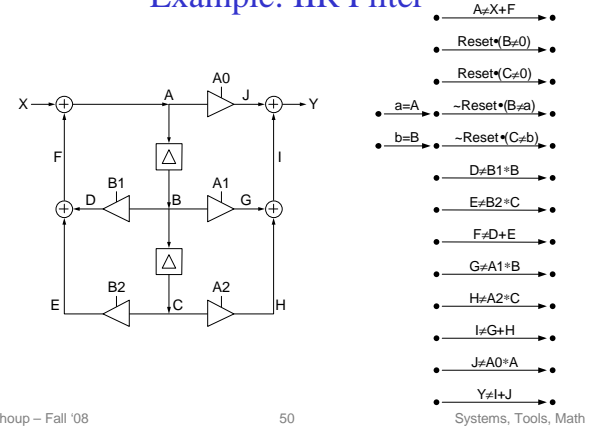
Sequential Constraints con't



disallowed: $[P] [Q]^* [-R]$

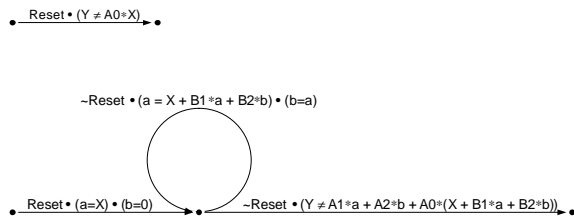
"If P, then while Q, R in the next state"

Example: IIR Filter



IIR Filter: Visible Behavior

Resolve & delete the hidden signals (A,B,C,D,E,F,G,H,I,J) to obtain the visible (input-output) behavior:



Sequential Constraint Graphs

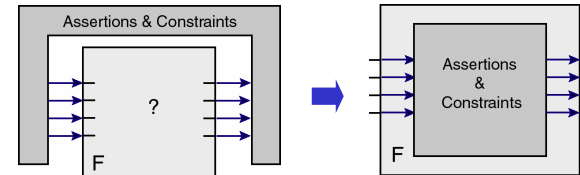
- Set of behavioral, sequential *constraints*
- Constraint specifies *joint state* of variables in time
 - Logical, arithmetic, and timing *dependencies*
 - *Disallowed* states, what cannot occur
 - Equivalent to / based upon *regular expressions*
- Inference engine combines constraints
 - Proofs, queries, simulation, analyses of conditions, etc
- Formal Verification
 - Abstract, RTL or gate-level, hardware or software, etc
 - No global state, avoids most combinatorial explosions

(Fred Furtek, *Applied Combinatorics*, Boundary Institute)

Remedies – Computing

- Single representation/language, unified hardware & software
- Deep primitives, new foundations (e.g. Boundary Math)
 - Beneath/prior to Logic, Set Theory, Turing machine
- Hardware/software structure model
 - Relational, hierarchical, self-referring, etc
- Formal techniques from the beginning
 - Initial specification, design, optimization, verification, debugging, field configuration, upgrading, etc.
 - Operation provably correct, by construction and anytime
 - Security, privacy provable and verifiable

Example: Design by Eversion



A proper formal specification is an implementation.

Summary & Conclusion

- Computer Science & Engineering need an overhaul
- Reconfigurable Computing is a challenge & an opportunity
 - Similarly any significantly parallel structure, e.g. multicore
- New maths and new tools are required beneath
- Single formal representation/language essential
- Deep primitives, mathematical foundations, bedrock
- Design is formal from the beginning, throughout
 - A proper formal specification is an implementation

The lack of adequate foundations is limiting applications, reliability, and security of complex, distributed, highly-parallel computing and information systems today.



Thanks for listening

rgshoup@rgshoup.com
www.rgshoup.com
rshoup@boundary.org
www.boundary.org