Computer Science : Achievements and Challenges circa 2000

Robert L. Constable

Cornell University

March 2000

Intro Remarks:

As we get older, we become more reflective; the same with scientific fields as seen in book titles such as:

> “the end of science”
>
> “dreams of a final theory”

The organizers gave the speakers a topic that invites reflection.

Intro Remarks (Cont ...)

The theme of reflecting on CS achievements and challenges is timely world wide. In the US, the National Research Council is about to conduct a national study of the nature of CS.

Why? Because NSF must explain why CS is getting so much new research funding. Many traditional scientists scoff at the idea that CS is more than a “technology” and hence dependent mainly on physics, and is perhaps a passing fad, destined to have few more birthdays.

I think otherwise. I will state my views to you and try to defend them. But no “experiment” can decide. At one level, science is philosophy.

Outline

- The Nature of Computer Science
- Achievements : behind the Information Revolution
- Challenges : the unity of science, the scope of computer science, and universities in the Information Age

The Nature of Computer Science

- What is science?
- Confusion about Computer Science
- How to clarify the nature of CS?

What is science?

People use the word “science” in many ways

- content of what we know
- special method of discovery
- new ways of doing things - technology

Feynman from *The Meaning of it all* 1998.

“all other aspects and characteristics of science can be understood directly when we understand that observation is the ultimate and final judge of the truth of an idea.”

“So what we call scientific knowledge today is a body of statements of varying degrees of certainty.”

“The work is not done for the sake of applications. It is done for the excitement of what is found out.”

“Thus science is not a specialists business, it is completely universal.”

Confusion about computer science.

- are there experiments?
- are there theories to test?
- is there a body of knowledge of varying degrees of certainty?
- what are the exciting discoveries?
- how does CS relate to the other “universal” sciences?

How to clarify the nature of computer science?

Look at its achievements

- its content landmarks
- its methods
- its research agenda

Articulate its challenges

- new dimension to science
- possible milestones

Achievements : behind the Information Revolution

There is an Information Revolution

- obvious manifestations
- government data
- role of technology exponentials
- historians' viewpoint

Computer science is essential in the Revolution

- Brookes/Sutherland study
- our legendary figures
  - Turing
  - Church
  - Chomsky
- case studies

The Information Revolution - obvious manifestations

New world

- Explosion of information on the Web: 5M domain names, .5M servers, 30M computers connected, a million new pages per day!
- 1 billion web pages
- We are connected - live in cyberspace
- We are not alone - “digital companions”
- The Deep Blue machine is world chess champion

The Information Revolution - obvious manifestations

The new economy

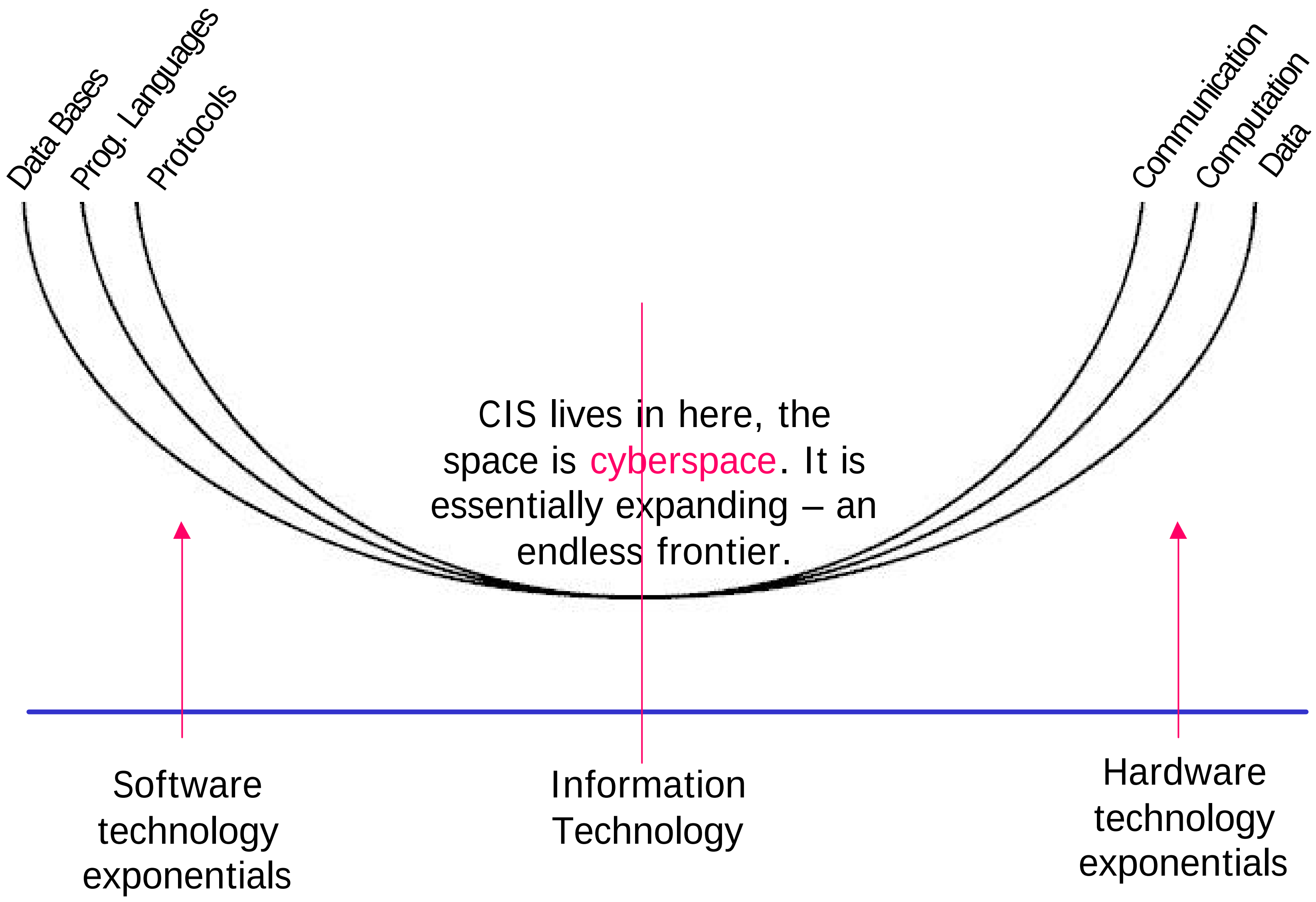
- Information corporations: Microsoft, Oracle, ...
- Information service industry: Amazon.com, theGlobe.com
- 1/3 of US economic growth since 1992
- 7.4 million Americans at wages 60% above average
- E-commerce is booming (expect $1.3 Trillion by 2003): CISCO, Compaq
- Israel as a second Silicon Valley

Scientific Basis - CIS is the intellectual core

- **The Internet**
  - Packet communications
  - Protocols (TCP/IP)
- **Web/Mobile code**
  - Java programming language
  - High-level PL's research
- **Cryptography**
  - Computational complexity theory
  - Intractable problems
- **E-business**
  - Security
  - Cryptography

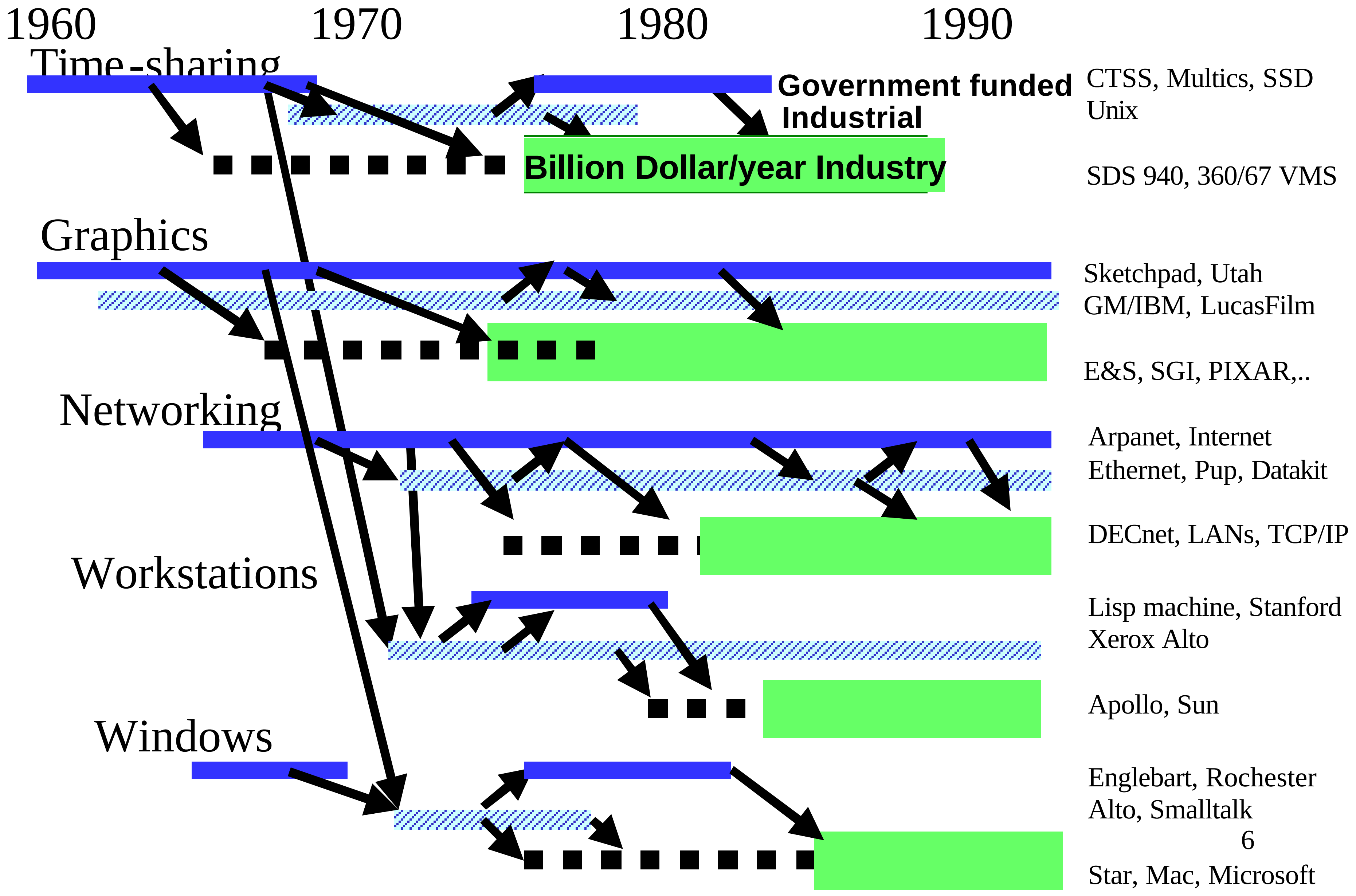
- **Multimedia**
  Data compression algorithms
  Computational geometry
- **Search engines**
  Vector space model
  Computational graph theory
  Natural language technology
- **Computational Science**
  Parallel compilation
  Cluster computing
  Theory Center role
- **Cognitive Science**
  Computational theory of mind
  Intelligent systems
  Robots

Relation to Technology

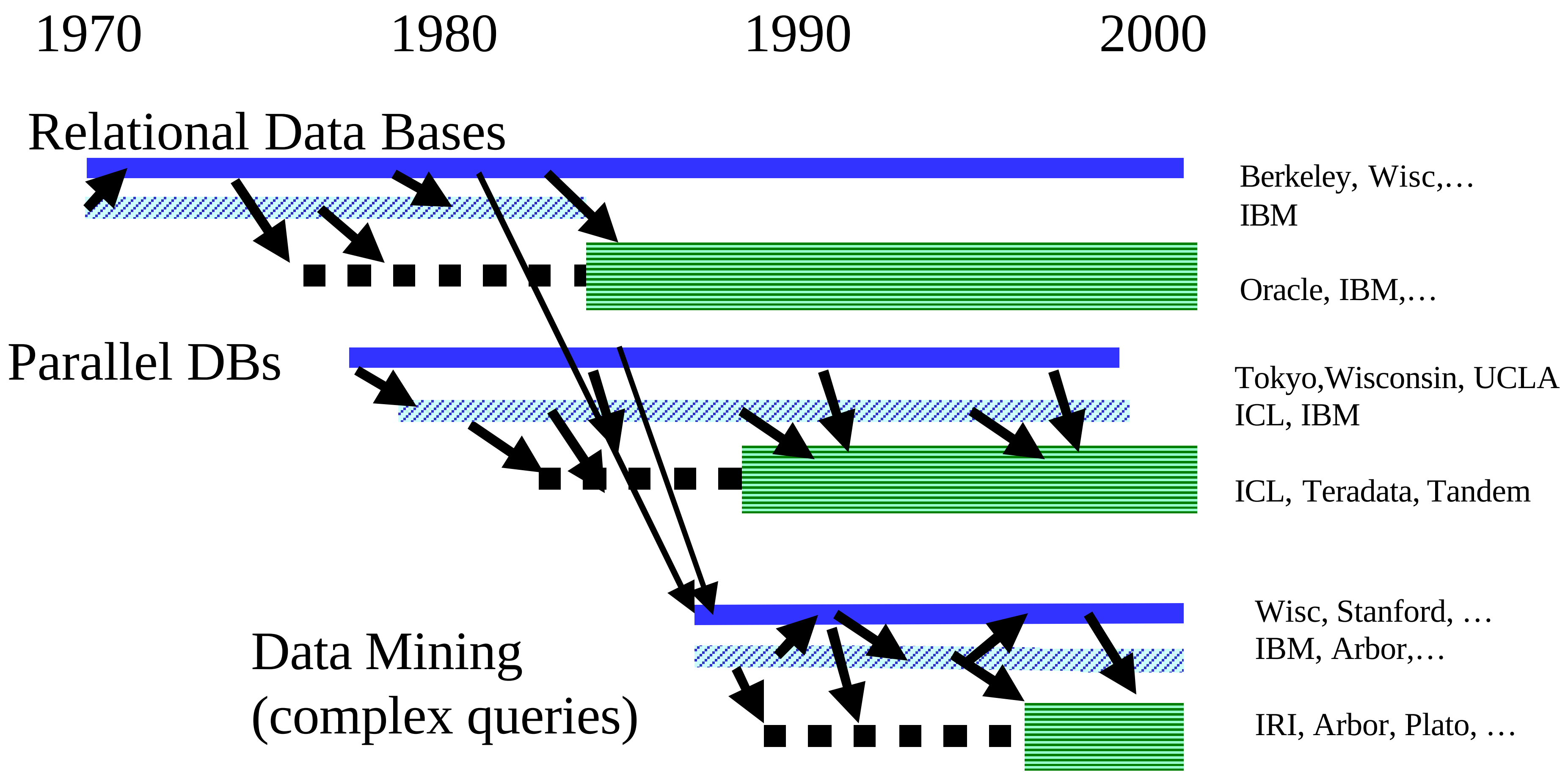

Data Bases
Prog. Languages
Protocols
Communication
Computation
Data
CIS lives in here, the space is cyberspace. It is essentially expanding – an endless frontier.
Software technology exponentials
Information Technology
Hardware technology exponentials

Research Investments Pay Off

CSTB –NRC Evolving the High-Performance Computing and Communications Imitative to Support the nations Information Infrastructure, NA Press, Washington DC, 1995.



Research Investments Pay Off



Alan Turing

Perhaps computer science started with him; he is one of the most influential figures. He wrote:

On computable numbers, with an application to the entscheidungsproblem *Proc. London Math Soc.* 1937

starts with a bad definition in first sentence, BUT …

Turing machines and compelling model of sequential string to string computation.

“a universal computing machine” (section 6)
diagonal method explored halting problem (circle-free problem)

Alan Turing – Computing machinery and intelligence *Mind* 1950

Can machines think?

Turing’s operational test – the imitation game

Turing’s belief – in 50 years his test will be passed at 70% level among average judges.

Turing’s role in COLOSSUS project 1943 at Bletchley Park

role of cryptography in WWII

contemporary significance of Turing’s work

Alonzo Church – The calculi of lambda conversion *Annals of Math studies* 1951

His lambda calculus inspired the programming language Lisp (1962) and provided the semantic model for Algol; for sequential computation over discrete data.

Alonzo Church – unsolvable problems and logic

Church also discovered one of the first unsolvable problems in logic (the entscheidungsproblem) and defined a simple theory of types, now used in program verification in HOL, Isabelle and PVS.

These provers all integrate model checking based on temporal logic from A. Pnueli.

Functional programming

fun(x.body) denotes a function, Church's λ x. body,

for example fun(x. x+5) is also written as:

$x \rightarrow x+5$ or λ x. x+5

the fix operator (or **Y** combinator) is a way to define

functions recursively, instead of writing

define f(x) = body[f] or

define f = fun(x. body[f]) we write

$$\boxed{\text{fix(fun(f. fun(x. body[f]))}}$$

Silly example of fix

define f = fun(x. f(x)) (this is equivalent to define f(x) = f(x))

f is a diverging recursive definition,

we call it bottom, $\perp$.

$$\boxed{\text{fix(f. fun(x. f(x))) is the fix form of this.}}$$

Unsolvability by fixed points *

Think of $\overline{\mathbf{N}} = \mathbf{N} \cup \{\perp\}$ where $\perp$ diverges, say $\perp$ = fix(λx. x)(0). Call these partial numbers.

If f is a function from $\mathbf{N}$ into $\overline{\mathbf{N}}$ then fix(f) $\in \overline{\mathbf{N}}$.

Is there a function $h : \overline{\mathbf{N}} \to \mathbf{B}$ such that h(t) = true iff t halts?

* Computational foundations of basic recursive function theory, with Scott Smith, TCS, 1993.

There is no computable halting function like h.

If h existed, we could define d in $\overline{\mathbf{N}}$ as

$$d = \mathrm{fix}(\lambda x.\ \text{if } h(x) \text{ then } \perp \text{ else } 1)$$

By this fix rule

$$\boxed{d = \text{if } h(d) \text{ then } \perp \text{ else } 1}$$

If h(d) then d halts, but d = ⊥, a contradiction to def of h.

If not h(d) then d diverges, but d = 1, a contradiction to def of h.

Noam Chomsky – ***Syntactic structures*** 1957

He introduced the notion of phrase structure grammars to study the structure of natural language. Grammars became the basis for a systematic study of the syntax of programming languages as well.

Compiler development

Compiler development is based on Chomsky’s work. Context free languages, context sensitive language, etc. In this area Jeff Ullman has been one of the leaders in both theory and practice.

The notion of nondeterministic computation matches the generation process in grammars. CS made sense of this and came up with the complexity class NP. This has led to the deep question

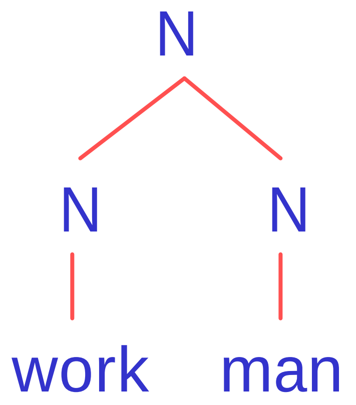
$$\boxed{P \overset{?}{=} NP}$$

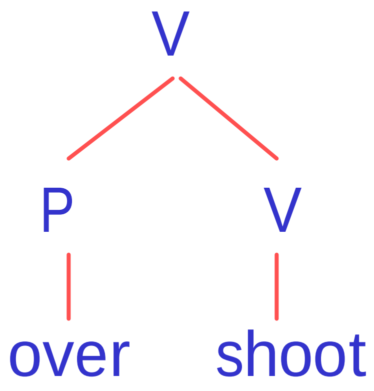
Fusion of Turing, Hartmanis, Chomsky.

Explaining natural language

Consider the rules governing compound words in English.

Compound words, like “workman” or “overshoot”; are two or more words put together to form one. There is a pattern, word structure pattern.





The right most noun is the head.

Explaining natural language (Cont…)

There is peculiar behavior here that many native speakers comment on. Let’s see how plurals and past tense are formed.

workman	workmen
horseshoe	horseshoes
overshoot	overshot

Plural is derived from the head.

Anomolous plurals

But why Toronto Maple Leafs not Maple Leaves ?

Why past tense of flys out is flied out not flew out ?

One plurals, Maple Leaf is a name, not a noun. It is headless, so its plural defaults to normal inflection.

For flied out, the word fly out is a verb based on a word that is not a verb.

Natural language processing

Linguistic and cognitive concepts can explain the strange behavior of compounds formed from irregular plurals as in

mice-infested vs rat-infested instead of rats-infested

men-bashing vs gay-bashing not gays-bashing

Peter Gordon confirmed that 3 to 5yr old children know this rule.

Cognitive explanation based on linguistic theory

Irregular plurals are stored in our mental dictionary whereas regular plurals are assembled as needed from inflection rules. Those rules apply too late in assembly process to help in forming compound words.

How people parse English

Evidence from garden path sentences

The horse raced past the barn fell.

Fat people eat accumulates.

The prime number few.

The tycoon sold the offshore oil tracts for a lot of money wanted to kill JR.

We parse depth-first using most likely alternative as our gamble. When it fails we must backup and try again.

Backtracking

The horse that raced past the barn fell.

The prime are few in number.

Linguistics and computer science

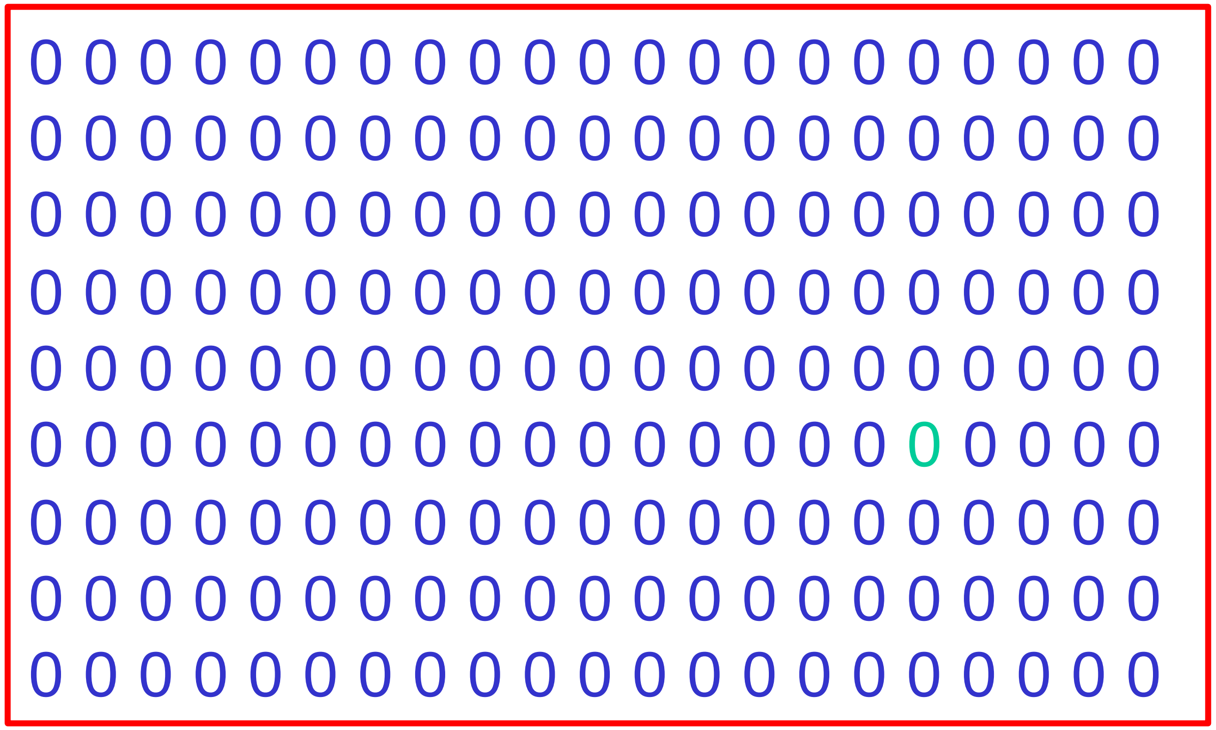
Linguistics and computer science are tightly linked,
As can be seen in my project with two BGU graduates:

Eli and Regina Barzilay.

Eli is studying reflection mechanisms in languages and logics and Regina is working on translating formal mathematical proofs into natural language.

Ann Treisman visual experiments

Green 0 in sea of blue.



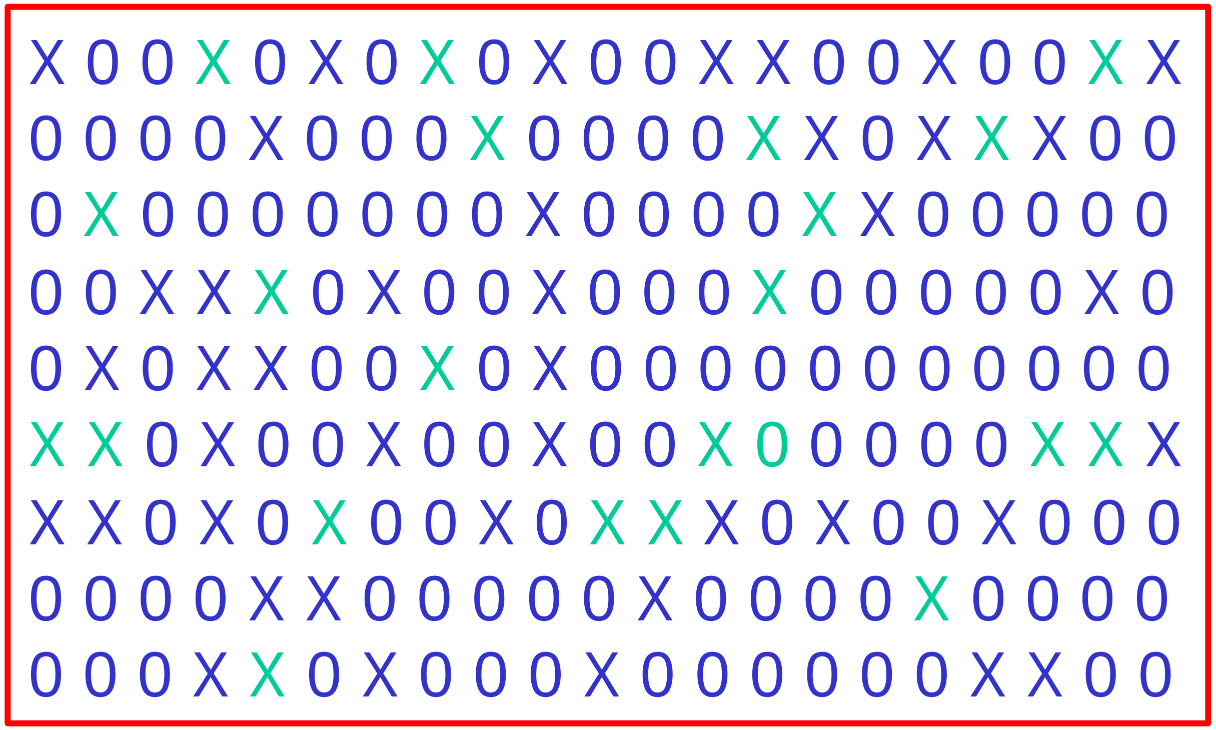
Ann Treisman visual experiments

One 0 in sea of X’s.

```
X X X X X X X X X X X X X X X X X X X X
X X X X X X X X X X X X X X X X X X X X
X X X X X 0 X X X X X X X X X X X X X X
X X X X X X X X X X X X X X X X X X X X
X X X X X X X X X X X X X X X X X X X X
X X X X X X X X X X X X X X X X X X X X
X X X X X X X X X X X X X X X X X X X X
X X X X X X X X X X X X X X X X X X X X
X X X X X X X X X X X X X X X X X X X X
```

Ann Treisman visual experiments

Green and 0.


X 0 0 X 0 X 0 X 0 X 0 0 X X 0 0 X 0 0 X X
0 0 0 0 X 0 0 0 X 0 0 0 0 X X 0 X X X 0 0
0 X 0 0 0 0 0 0 0 X 0 0 0 0 X X 0 0 0 0 0
0 0 X X X 0 X 0 0 X 0 0 0 X 0 0 0 0 0 X 0
0 X 0 X X 0 0 X 0 X 0 0 0 0 0 0 0 0 0 0 0
X X 0 X 0 0 X 0 0 X 0 0 X 0 0 0 0 0 X X X
X X 0 X 0 X 0 0 X 0 X X X 0 X 0 0 X 0 0 0
0 0 0 0 X X 0 0 0 0 0 X 0 0 0 0 X 0 0 0 0
0 0 0 X X 0 X 0 0 0 X 0 0 0 0 0 0 X X 0 0

Talking about computer systems

Why isn’t my computer printing? Because the OS does not know that you replaced your dot-matrix printer with a laser printer; it still thinks it’s talking to a dot-matrix printer and is waiting for an acknowledgement from the printer.

But the printer is ignoring the message; it does not understand it, because it expects it to begin with a %.

You need to get the attention of the OS.

Nuprl as an intelligent agent

It's successes embolden us to talk about "how Nuprl thinks."

Example 1.

My gosh, it realized that $\exists$x:A.B was decidable and performed a case analysis, how did it know that?

Mentalese and Nuprlese

Example 2.

Did Nuprl prove all the other subgoals? No, it doesn’t believe that E is an equivalence relation, so it asked a lot of questions I didn’t expect.

What is it asking about? It wants to know why certain values are in range? What values? Only those that change the security level.

It should know that, the value is in range. That is a require-ment of the decryption function. That’s part of the contract for that process.

Mentalese and Nuprlese (cont ...)

Example 2. (cont ...)

Yes, but Nuprl claims that the argument to the decryption function is not of the right type. But that can’t be, ask it to explain why it is not of the right type.

It reports that if process Q sent its message M with the decrypt argument just before monitor M failed, then Q could have used an undetected modular equivalence to generate the message value.

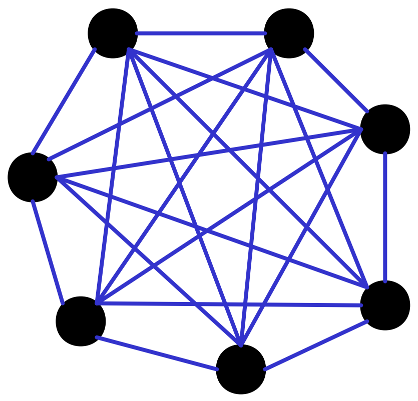
Well I’ll be darned. We all missed that.

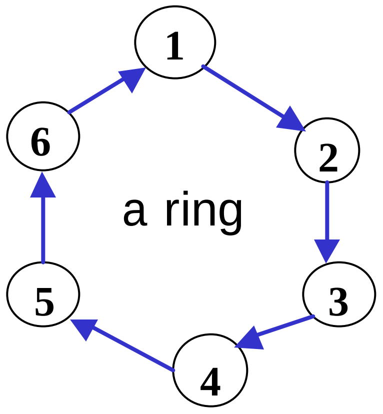
Distributed computing

K. Birman, J. Gray, C. A. R. Hoare, L. Lamport, B. Lampson, B. Liskov, N. Lynch, R. Milner, F, Schneider.

A new model of computing – event-driven, reactive communicating sequential processes.

We see it on the Internet, in the mind.





Example – asynchronous leader election (in a ring)

LCR algorithm (Le Lann, Chang, Roberts)

each process sends its identifier around ring

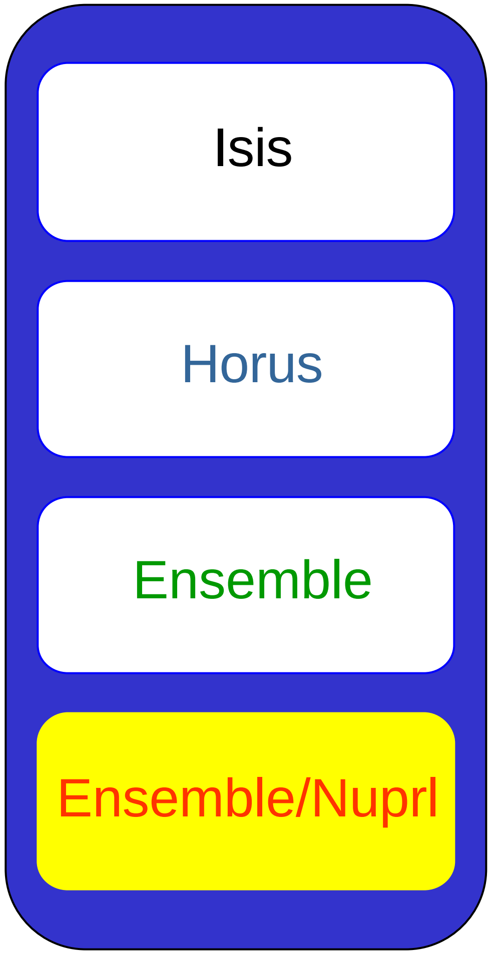
```
when    $p_i$ gets a $j<i$, discard
        $p_i$ gets a $j>i$, forward
        $p_i$ gets   $j=i$, declare itself the leader
```

How to reason about such processes?

Pnuelli gave us methods.

Group Communication Systems

Reliable and secure networking in safety-critical applications



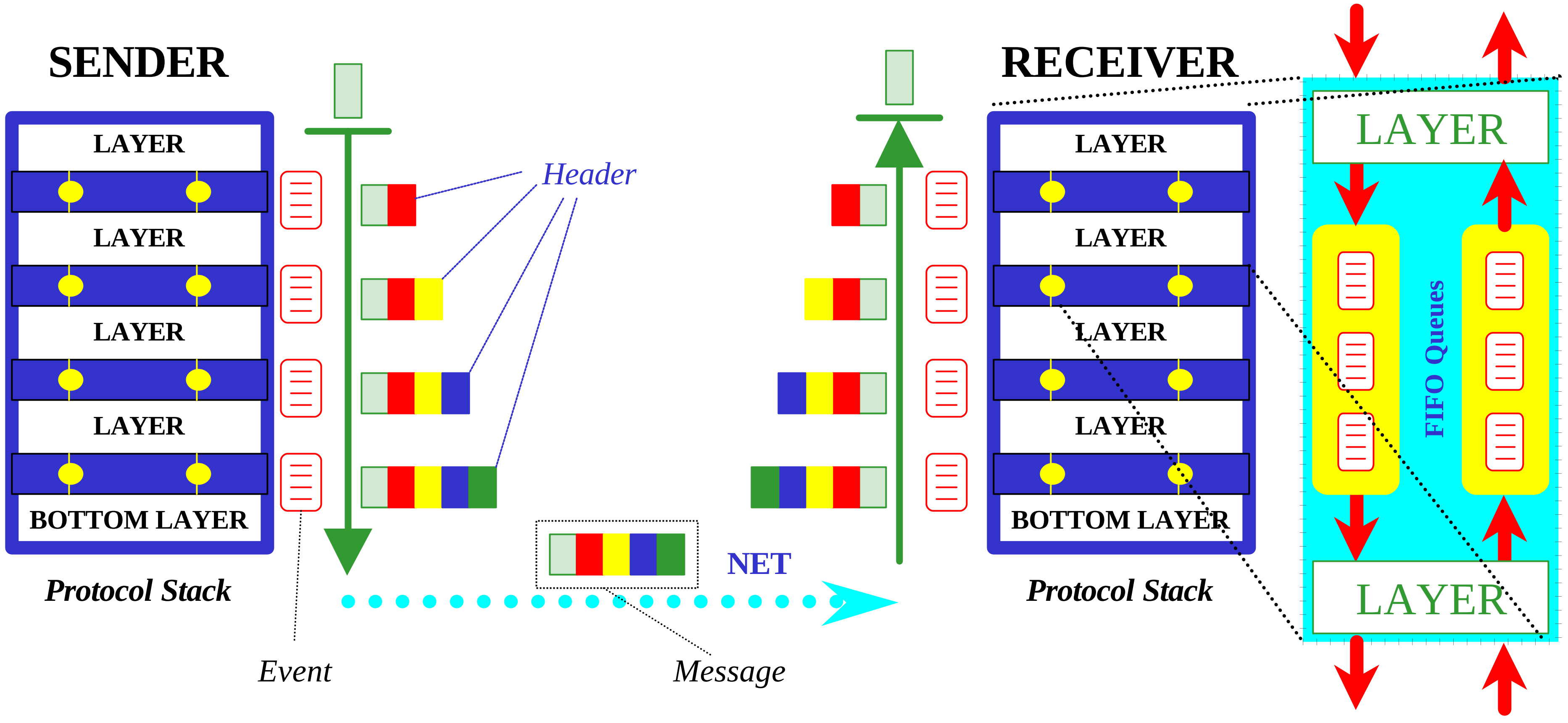
Technology for securing networked applications
- widely used: NY Stock Exchange, French Air Traffic Control …

Added flexibility through protocol stacking
- reconfigurable to specific needs of applications

Reference implementation in Ocaml
- small protocol layers, easy to check and modify
- portable to a variety of platforms
- highest performance due to fast-track reconfiguration

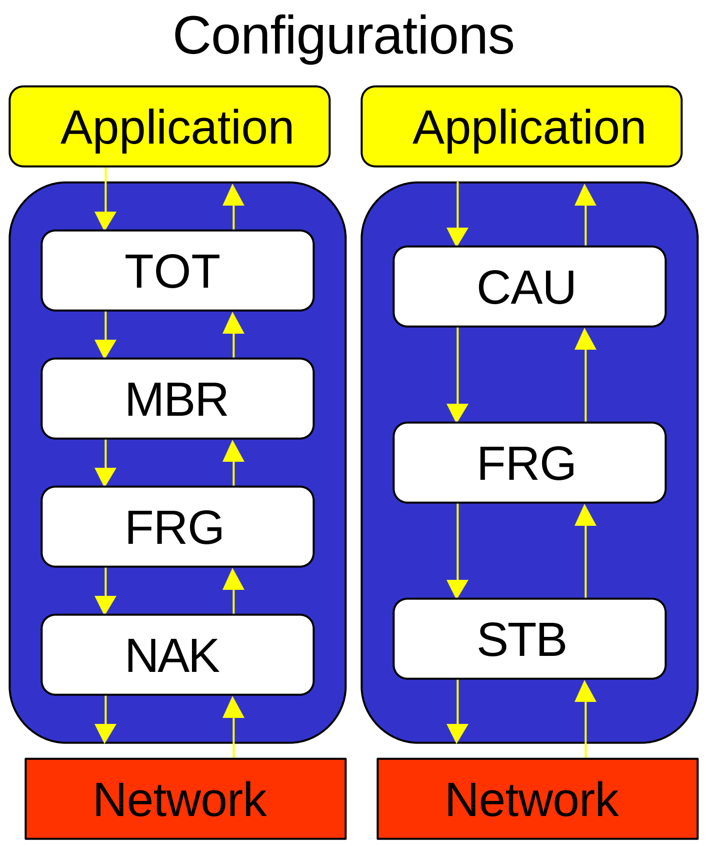
Logical development tools for network security
- verification of critical properties (beyond type checking)
- formal documentation / logical debugging
- automated and verified fast-track reconfiguration

Architecture of Ensemble

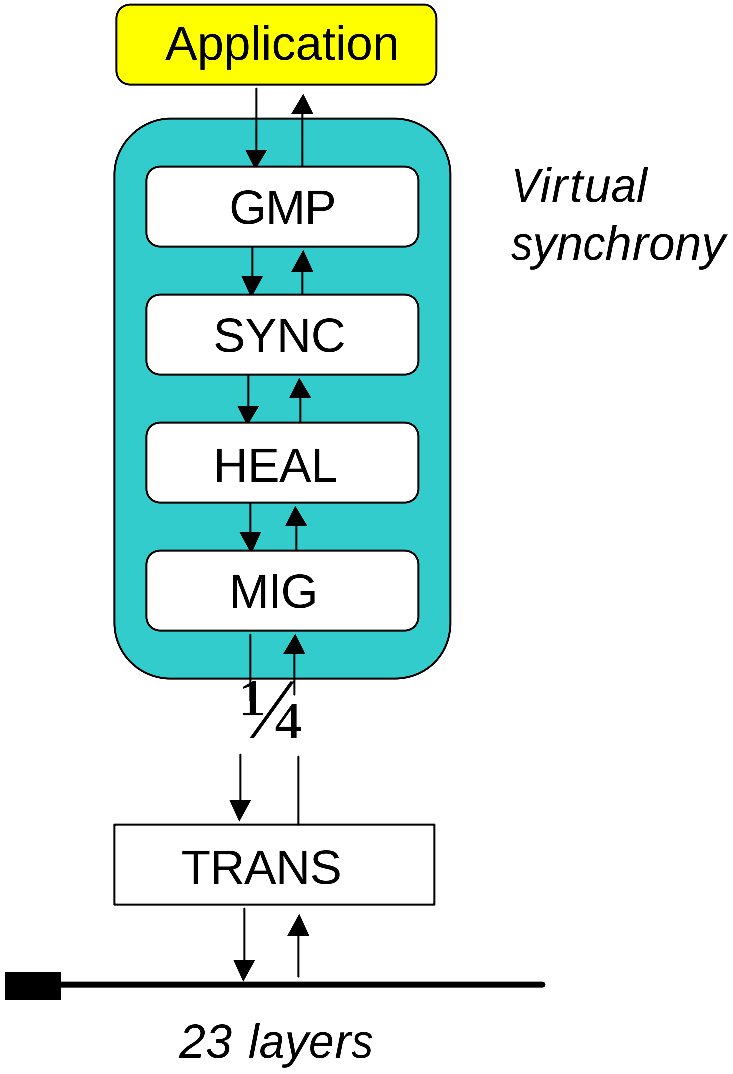


Cost of modularity

- Poor performance
  - redundant code
  - abstraction enforcement
- Difficult to verify complete systems
  - combinatorial number of configurations



Ensemble Implementation

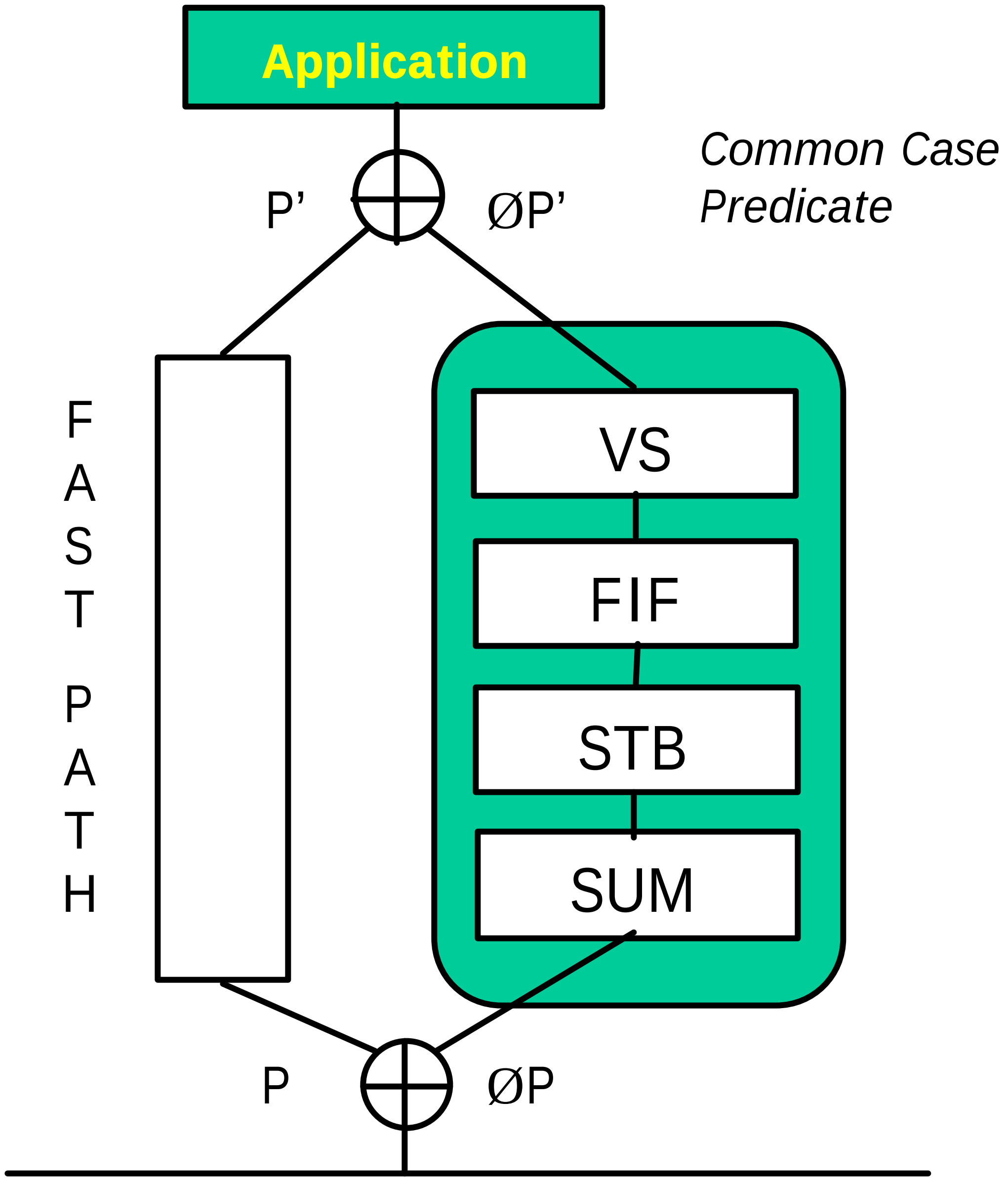


- Layered protocol stacks
- Each layer implements a property
- Protocols are
  - small (-300 lines ML)
  - roughly orthogonal
- Configuration is application-specific
- About 50 layers; thousands of protocols

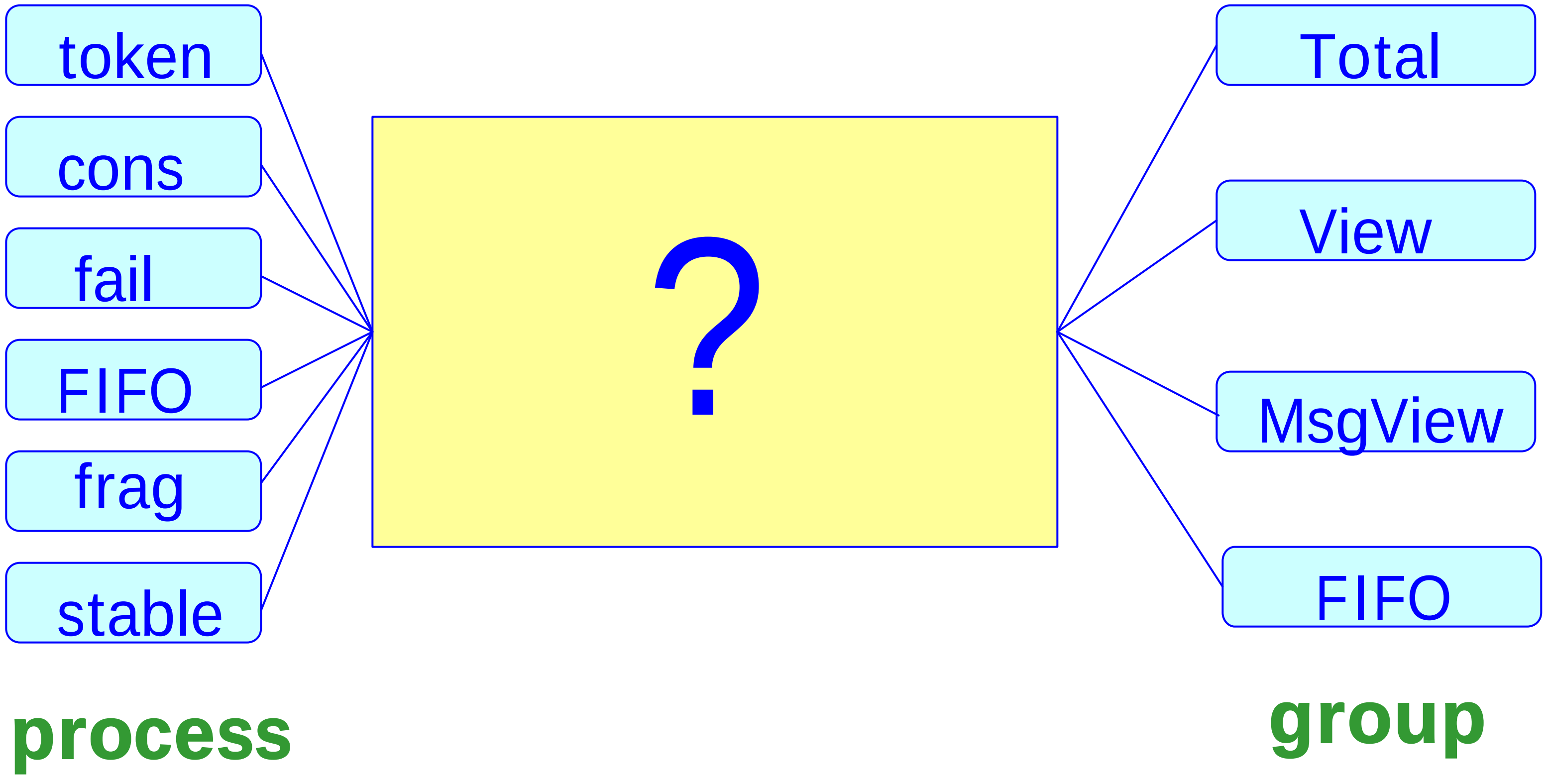
Fast path

Extract common path from the protocol

Speedups of x2-x50



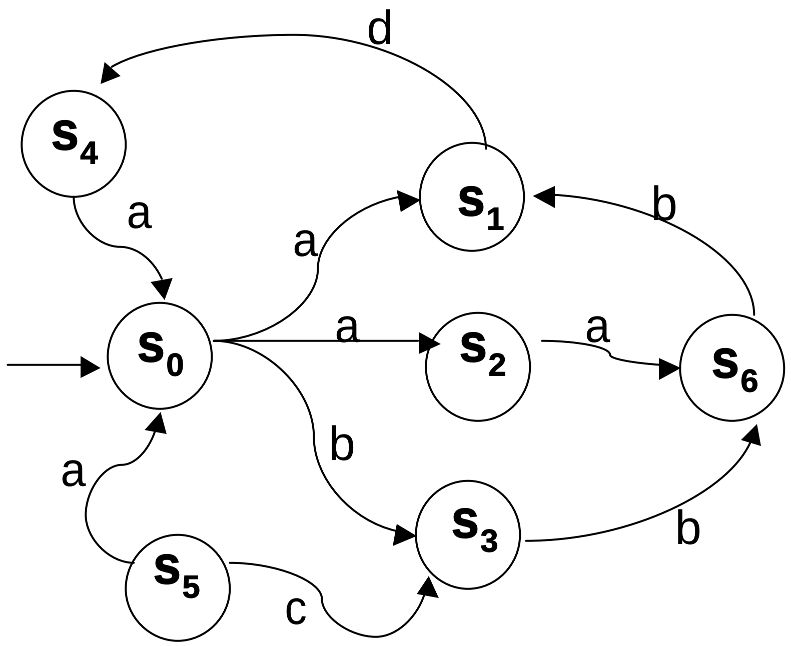
Verification - How do we bridge the gap?



Labeled Transition Systems (Nondeterministic Automata)

Signature   A : Type x S : Type x I : (S $\rightarrow$ Prop) x T : (S $\rightarrow$ A $\rightarrow$ S $\rightarrow$ Prop)

<A, S, I, T> is a system



Can consider these as unlabeled if we take the state space to be S x A.

Sample Formal IOA

state $x:T_1$
state $y:T_2$

initially $R_0(x,y)$
action $a:T_3$

precondition $a(v) \Rightarrow R_1(x,y,v)$

effect $a(v) \Rightarrow x := f(x,y,v)$
$a(v) \Rightarrow y := g(x,y,v)$

only [a] affects x
only [a] affects y

Note: inputs never have preconditions.

Establishing Invariants

Let A be a formal IOA, our goal is to prove theorems of the form (safety properties):

$$[[A]]_{\rho\ e} \models \text{always } s.\ [[I]]_{\rho\ e} \bullet$$

We formalize a method for doing this.

Method of proving invariants by induction ("VCG method")

Let A be the sample IOA and let I be

$$\{ R_3(x,y) \}.$$

ioa_inv_vc(A, I) ==

$\{R_0(x,y) \Rightarrow R_3(x,y) \} \cup$

a: $\{ R_3(x,y) \wedge R_1(x,y,v) \Rightarrow R_3(f(x,y,v),g(x,y,v)) \}$

If we know ioa_inv_vc(A, I) at any state, then we can prove that I is an invariant by induction. We prove this formally at the metalevel.

VCG Theorem

```
S vc_correctness
        ∀ A:Ioa. ∀ I:Fmla. ∀ rho:Decl. ∀ de:Sig. ∀ e:{[[de]] rho}.
           (tc_ioa(A;de)
           ⇒ tc_pred(I;A.ds;<>;de)
           ⇒ covers_pred(A;I)
           ⇒ closed_pred(I)
           ⇒ single_valued_decls(A.ds)
           ⇒ (∀ s:{[[A.ds]] rho}
                (reachable([[A]] rho de e;s) ) [[VCs(A;I)]](rho,A.ds,A.da,de) e s))
           ⇒ [[A]] rho de e |= always s.(pred_mng(I;rho;A.ds;<>;de;e;s;  )))
```

The proof of the vc-correctness theorem is by induction over the reachable states. The base case of the induction is proved by using the meaning of the initial vc.

In the induction step, we must show that for any (reachable) state s, and any action a and state s', if I holds in state s and the triple (s, a, s') is in the transition relation of A, then I holds in state s'.

We must first argue that it is enough to assume that action a is declared in A. Using classical reasoning, we can split into two cases. In case action a is not declared in A, the covers_pred(A,I) hypothesis lets us conclude that the meaning of I is the same in states s and s' because no variable mentioned in I is affected by action a.

Thus we are left with only the case where action a is a declared action of automaton A.

Lessons from achievements

are there experiments?

- we create theories that can be tested in other fields, theories of language, mind, protein folding, gene expression, …
- we explore computational phenomena – phase transition in SAT, algorithms in practice
- we fail at some automation, succeed at others (speech recognition vs compilers)

Lessons from achievements (Cont ...)

are there theories to test?

- theories and models are implemented, tested and demonstrated e.g., compilers, theorem provers, pattern recognizers
- semantic theories – guiding consistent design (Von Neuman machine, distributed computing)
- complexity theory – predicting algorithmic behavior

there are dynamic artifacts that survive and those that do not (type checkers, model checkers, theorem provers, spell checkers vs hand writing analyzers, speech understanding system, go-players, natural language database queries)

Lessons from achievements (Cont ...)

there is a large body of knowledge !
a widely taught body of knowledge

some statements are mathematical facts

are beliefs (P $\neq$ NP)

are conjectures

a unifying framework for diverse discoveries

What are the exciting discoveries in CS?

- we can define computability
- there are unsolvable and intractable problems
- constructive abstraction is key aspect of intelligent systems
- common sense is hard
- some abstract mathematics and games are easy
- we can formalize arguments about code, hence software is both an artifact and mathematical object
- we can build mathematical assistants

Is CS part of the universal fabric of science?

CS has extended science into a new dimension – cyberspace

results in that dimension unify those projected into ordinary space/time

we see this in Linguistics, Psychology and Biology – perhaps in Physics as well.

“Quantum computation is the nexus at which "software" concepts like information, randomness and order tangle with "hardware" concepts such as matter and force. I am convinced that in the search for a unified theory of nature, in which the deepest level of reality would be exposed, the study of quantum computation will mark the beginning of a new era.” *Paul Davies*

Challenges

Can we unify our major theories?

semantics and complexity
(reason about complexity as well as correctness)
processes and hueristics

Can our formalisms capture our design practice and help us automate it further?

(how far will constructive abstraction carry us,
e.g., modules, theories and aspects)

Can we build a self reproducing system?

Can we discover deeper unity in our basic concepts?

- generation and nondeterministic acceptance are the same
- types and propositions are the same
- constructive proofs define data and programs

A vision for designing, building and supporting systems

Reactive agents using formal mathematical database help us design, build and maintain systems. They explain design invariants and monitor them; they help find errors and restore invariants. They increase system reliability dramatically and increase programmer productivity and code performance. But they also embody a theory of intelligence and treat systems as mathematical objects – well defined objects in cyberspace.

Challenges for Universities in Information Age

- CIS is emerging as one of the major intellectual disciplines of the next century. Its scope in university terms will be at the scale of colleges not departments or centers.
- CIS as a body of knowledge occurs early in the "tree of knowledge" with consequences for nearly all other subjects and disciplines.

Challenges for Universities in Information Age (Cont…)

- CIS has the character of science, quantitative mathematical, predictive; the character of technology, practical, constructive, with an engineering component and the character of art, shaped by elegance and beauty and loosely constrained by laws limited largely by imagination.
- CIS is a bridge between the humanities and arts and the sciences as we see most clearly in the computational theory of mind.