Perche' Frances E. Allen ha vinto la Turing Award 2006?

Eugenio Moggi

moggi@disi.unige.it

DISI, Univ. of Genova

Sommario

- Cosa e' la Turing Award?
- Breve CV di Frances E. Allen
- Interpreti e Compilatori/Traduttori
- Perle di Saggezza (e Hype)

Cosa e’ la Turing Award?

Turing Award

- Premio istituito nel 1966
- dalla Association for Computing Machinery (ACM)
- descrizione presa da www.acm.org

  The Turing Award is ACM's most prestigious technical award. It is given to an individual selected for contributions of a technical nature made to the computing community. The contributions should be of lasting and major technical importance to the computer field.
- e' accompagnato da un compenso di 250.000 USD (100.000 USD fino al 2006) Attualmente il supporto finanziario e' fornito Intel Corporation e Google Inc.
- Prende il nome da Alan Turing (1912-54), matematico e logico inglese che
  - contribui' alla Teoria della Calcolabilita': Macchina di Turing
  - fu' un precursore dell'Intelligenza Artificiale: Test di Turing.

Association for Computing Machinery

- Associazione con sede centrale a New York fondata nel 1947, anno in cui fu' creato il primo computer digitale a programma memorizzato (ENIAC modificato)
- Descrizioni prese da www.acm.org
  - ACM is the world's oldest and largest educational and scientific computing society. Since 1947 ACM has provided a vital forum for the exchange of information, ideas, and discoveries.
  - Today, ACM serves a membership of computing professionals and students in more than 100 countries in all areas of industry, academia, and government.
  - The purpose of ACM is to advance the science, development, construction, and application of the new machinery for computing, reasoning, and other handling of information. Advancing Computing as a Science and a Profession.

Informatica = mix di Scienza(Matematica) e Tecnica(Ingegneria)

Computer Science&Engineering

Alcuni vincitori della Turing Award

1966 Alan J. Perlis: for his influence in the area of advanced programming techniques and compiler construction.

...

2005 Peter Naur: for fundamental contributions to programming language design and the definition of Algol 60, to compiler design, and to the art and practice of computer programming.

2006 Frances E. Allen: for pioneering contributions to the theory and practice of optimizing compiler techniques that laid the foundation for modern optimizing compilers and automatic parallel execution.
For contributions that fundamentally improved the performance of computer programs in solving problems, and accelerated the use of high performance computing.

Alcuni vincitori della Turing Award

1966 Alan J. Perlis: for his influence in the area of advanced programming techniques and compiler construction.

...

2005 Peter Naur: for fundamental contributions to programming language design and the definition of Algol 60, to compiler design, and to the art and practice of computer programming.

2006 Frances E. Allen: for pioneering contributions to the theory and practice of optimizing compiler techniques that laid the foundation for modern optimizing compilers and automatic parallel execution.
For contributions that fundamentally improved the performance of computer programs in solving problems, and accelerated the use of high performance computing.

2007 Edmund M. Clarke, E. Allen Emerson, and Joseph Sifakis: for their original and continuing research in a quality assurance process known as Model Checking. Their innovations transformed this approach from a theoretical technique to a highly effective verification technology that enables computer hardware and software engineers to find errors efficiently in complex system designs.

CV di Frances Elizabeth Allen
preso da
www.acm.org e en.wikipedia.org

Breve CV

1932 nasce nello stato di New York

Breve CV

1932 nasce nello stato di New York

1954 BSc in Mathematics, Albany State Teachers College

Breve CV

1932 nasce nello stato di New York

1954 BSc in Mathematics, Albany State Teachers College

1957 MSc in Mathematics, University of Michigan
Inizia l'attivita' di insegnante, ma per ripianare i debiti contratti per studiare, va' a lavorare all'IBM (T.J.Watson Research Center). . . . Rimase all'IBM per 45 anni.
IBM (International Business Machines) Corporation:

- tra le maggiori nel settore informatico (dall'HW alla consulenza)
- la piu' longeva nel settore e quella detentrice di piu' brevetti
- tuttora leader nel settore mainframe (e piu' di recente nei supercomputers)

Breve CV

1932 nasce nello stato di New York

1954 BSc in Mathematics, Albany State Teachers College

1957 MSc in Mathematics, University of Michigan

1957 insegna FORTRAN a scienziati ed ingegneri dell'IBM
FORTRAN (FORmula TRANslation) primo linguaggio *ad alto livello* di successo:

- 1954 all'IBM John Backus (Turing Award 1977) inizia a lavorare all'idea
- 1957 viene distribuito il primo compilatore (ma e' accolto con scetticismo)
- successive versione sono ancora in uso per applicazioni scientifiche

Allen riconosce la sfida posta dall'High Performance Computing: fornire alte prestazioni, senza dover esporre l'architettura sottostante.

Breve CV

1932 nasce nello stato di New York

1954 BSc in Mathematics, Albany State Teachers College

1957 MSc in Mathematics, University of Michigan

1957 insegna FORTRAN a scienziati ed ingegneri dell'IBM

1966 articolo Program Optimization getta le basi concettuali per l'analisi e la trasformazione sistematica dei programmi.

Breve CV

1932 nasce nello stato di New York

1954 BSc in Mathematics, Albany State Teachers College

1957 MSc in Mathematics, University of Michigan

1957 insegna FORTRAN a scienziati ed ingegneri dell'IBM

1966 articolo Program Optimization

1970 articoli Control Flow Analysis e A Basis for Program Optimization forniscono il contesto per avere analisi del flusso ed ottimizzazioni efficienti ed efficaci.

Breve CV

1932 nasce nello stato di New York

1954 BSc in Mathematics, Albany State Teachers College

1957 MSc in Mathematics, University of Michigan

1957 insegna FORTRAN a scienziati ed ingegneri dell'IBM

1966 articolo Program Optimization

1970 articoli Control Flow Analysis e A Basis for Program Optimization

1971 articolo A Catalog of Optimizing Transformations con J.Cocke (Turing award 1987) fornisce la prima descrizione sistematica delle trasformazioni ottimizzanti.
Allen lavora a compilatori sperimentali dell'IBM, che dimostrano la fattibilita' dei moderni compilatori ottimizzanti.

Breve CV

1932 nasce nello stato di New York

1954 BSc in Mathematics, Albany State Teachers College

1957 MSc in Mathematics, University of Michigan

1957 insegna FORTRAN a scienziati ed ingegneri dell'IBM

1966 articolo Program Optimization

1970 articoli Control Flow Analysis e A Basis for Program Optimization

1971 articolo A Catalog of Optimizing Transformations con J.Cocke (Turing award 1987)

1984 dirige il progetto PTRAN, che affronta le sfide poste dai calcolatori paralleli e sviluppa vari concetti (p.e. grafo delle dipendenze) usati in molti compilatori parallelizzanti.

Breve CV

1932 nasce nello stato di New York

1954 BSc in Mathematics, Albany State Teachers College

1957 MSc in Mathematics, University of Michigan

1957 insegna FORTRAN a scienziati ed ingegneri dell'IBM

1966 articolo Program Optimization

1970 articoli Control Flow Analysis e A Basis for Program Optimization

1971 articolo A Catalog of Optimizing Transformations con J.Cocke (Turing award 1987)

1984 dirige il progetto PTRAN, che affronta le sfide poste dai calcolatori paralleli

1995 presidente della IBM Academy of Technology , una struttura interna che fornisce consulenza tecnica all'IBM.

Breve CV

1932 nasce nello stato di New York

1954 BSc in Mathematics, Albany State Teachers College

1957 MSc in Mathematics, University of Michigan

1957 insegna FORTRAN a scienziati ed ingegneri dell'IBM

1966 articolo Program Optimization

1970 articoli Control Flow Analysis e A Basis for Program Optimization

1971 articolo A Catalog of Optimizing Transformations con J.Cocke (Turing award 1987)

1984 dirige il progetto PTRAN, che affronta le sfide poste dai calcolatori paralleli

1995 presidente della IBM Academy of Technology

2002 va' in pensione . . . dopo aver completamente ripianato i suoi debiti.
L'ultimo ruolo ricoperto all'IBM e' Senior Technical Advisor del Research Vice-President per Soluzioni, Applicazioni e Servizi.

Breve CV

1932 nasce nello stato di New York

1954 BSc in Mathematics, Albany State Teachers College

1957 MSc in Mathematics, University of Michigan

1957 insegna FORTRAN a scienziati ed ingegneri dell'IBM

1966 articolo Program Optimization

1970 articoli Control Flow Analysis e A Basis for Program Optimization

1971 articolo A Catalog of Optimizing Transformations con J.Cocke (Turing award 1987)

1984 dirige il progetto PTRAN, che affronta le sfide poste dai calcolatori paralleli

1995 presidente della IBM Academy of Technology

2002 va' in pensione

Alcuni importanti riconoscimenti:

- 1989 prima donna ad essere nominata IBM Fellow
- 2006 prima donna a vincere la Turing Award

Nel 2000 IBM crea la Frances E. Allen Women in Technology Mentoring Award.

Linguaggi di Programmazione
Interpreti/Macchine Virtuali
Traduttori/Compilatori

Hardware e Software

- HW Hardware: si puo' prendere a calci
  SW Software: si puo' solo imprecarci contro

Hardware e Software

- HW Hardware: si puo' prendere a calci
  SW Software: si puo' solo imprecarci contro
- rivoluzione industriale (materia+energia): telaio meccanico

$$Input \longrightarrow \boxed{\text{HW}} \longrightarrow Output$$

il processo di trasformazione non richiedere competenze artigianali specifiche

Hardware e Software

- HW Hardware: si puo' prendere a calci
  SW Software: si puo' solo imprecarci contro
- rivoluzione industriale (materia+energia): telaio meccanico

$$Input \longrightarrow \boxed{\text{HW}} \longrightarrow Output$$

  il processo di trasformazione non richiedere competenze artigianali specifiche
- rivoluzione informatica (materia+energia+informazione):
  calcolatore con programma memorizzato

$$Input \longrightarrow \boxed{\text{HW+SW}} \longrightarrow Output$$

  il processo di trasformazione e' modificabile cambiando il solo SW

Hardware e Software

- HW Hardware: si puo' prendere a calci
  SW Software: si puo' solo imprecarci contro
- rivoluzione industriale (materia+energia): telaio meccanico

$$Input \longrightarrow \boxed{\text{HW}} \longrightarrow Output$$

  il processo di trasformazione non richiedere competenze artigianali specifiche
- rivoluzione informatica (materia+energia+informazione):
  calcolatore con programma memorizzato

$$Input \longrightarrow \boxed{\text{HW+SW}} \longrightarrow Output$$

  il processo di trasformazione e' modificabile cambiando il solo SW
- l'economia del software (e dell'informazione digitale):
  - non si usura/consuma con l'uso
  - e' duplicabile (copyleft) e trasferibile a costo quasi nullo

Interpreti e Compilatori

- L linguaggio (di programmazione)
  P: L programma scritto nel linguaggio L
  $[\![P]\!]_L(in) = out$ significato (*semantica input-output*) di P , cioe' effetto

$$in \longrightarrow \boxed{\text{HW}+P} \longrightarrow out$$

dell'esecuzione di P su un macchina in grado di *capire* L

Interpreti e Compilatori

- L linguaggio (di programmazione)
  P: L programma scritto nel linguaggio L
  $[\![P]\!]_L(in) = out$ significato (*semantica input-output*) di P
- *programmi applicativi* risolvono *problemi reali* , p.e.:
  - stesura di un documento
  - previsioni atmosferiche per domani (meteo)
  - calcolo degli interessi sui c/c per il mese corrente (banca)
  - visualizzazione di una immagine TAC (medico)
  - analisi strutturale del modello matematico di un edificio (ingegnere)

Interpreti e Compilatori (programmi come dati)

- L linguaggio (di programmazione)
  P: L programma scritto nel linguaggio L
  $[\![P]\!]_L(in) = out$ significato (*semantica input-output*) di P
- *programmi applicativi* risolvono *problemi reali*
- programmi che migliorano la *produttivita'* di chi sviluppa SW applicativo
  - interprete(=traduttore in simultanea) P_{int} per L_1 scritto in L_0

    $$[\![P_{int}]\!]_{L_0}(P_1, in) = [\![P_1]\!]_{L_1}(in)$$

    cioe' $[\![P_{int}]\!]_{L_0}$ si comporta come un una macchina che capisce L_1
  - compilatore(=traduttore) P_{comp} da L_1 a L_2 scritto in L_0

    $$[\![[\![P_{comp}]\!]_{L_0}(P_1)]\!]_{L_2}(in) = [\![P_1]\!]_{L_1}(in)$$

    cioe' $[\![P_{comp}]\!]_{L_0}$ traduce P_1: L_1 in un P_2: L_2 con lo *stesso significato* di P_1

per P_{int} e P_{comp} il programma P_1 e' un dato di input.

Interpreti e Compilatori (programmi come dati)

- L linguaggio (di programmazione)
  P: L programma scritto nel linguaggio L
  $[\![P]\!]_L(in) = out$ significato (*semantica input-output*) di P
- *programmi applicativi* risolvono *problemi reali*
- programmi che migliorano la *produttivita'* di chi sviluppa SW applicativo
  - interprete(=traduttore in simultanea) P_{int} per L_1 scritto in L_0
  - compilatore(=traduttore) P_{comp} da L_1 a L_2 scritto in L_0
- conviene fare una *buona* traduzione $P_1 \longmapsto P_2$, se $\boxed{\text{si usa } P_2 \textit{ molte volte}}$
  - traduzione *corretta* $\overset{\Delta}{\Longleftrightarrow} [\![P_2]\!]_{L_2} = [\![P_1]\!]_{L_1}$
  - traduzione *ottimizzante* $\overset{\Delta}{\Longleftrightarrow} P_2$ e' il *migliore* tra i P: L_2 t.c. $[\![P]\!]_{L_2} = [\![P_1]\!]_{L_1}$

Interpreti e Compilatori (programmi come dati)

- L linguaggio (di programmazione)
  P: L programma scritto nel linguaggio L
  $[\![P]\!]_L(in) = out$ significato (*semantica input-output*) di P
- *programmi applicativi* risolvono *problemi reali*
- programmi che migliorano la *produttivita'* di chi sviluppa SW applicativo
  - interprete(=traduttore in simultanea) P_{int} per L_1 scritto in L_0
  - compilatore(=traduttore) P_{comp} da L_1 a L_2 scritto in L_0
- conviene fare una *buona* traduzione $P_1 \longmapsto P_2$, se $\boxed{\text{si usa } P_2 \textit{ molte volte}}$
  - traduzione *corretta* $\overset{\Delta}{\Longleftrightarrow} [\![P_2]\!]_{L_2} = [\![P_1]\!]_{L_1}$
  - traduzione *ottimizzante* $\overset{\Delta}{\Longleftrightarrow} P_2$ e' il *migliore* tra i P: L_2 t.c. $[\![P]\!]_{L_2} = [\![P_1]\!]_{L_1}$

Mission Impossible!

P_{comp} *ottimizzante* solo in senso euristico (problema ingegneristico), p.e.

produce risultati *confrontabili* a quelli di un programmatore esperto

Linguaggi

user (alto livello) $\longrightarrow$ intermediate $\longrightarrow$ machine (basso livello)

- Linguaggi ad alto livello - programmi comprensibili (*specifiche eseguibili*)

* linguaggi funzionali (LISP fine anni 50')

$$\begin{array}{ll} \text{let} & f_1(\overline{x}_1) = e_1 \\ & \ldots \\ & f_n(\overline{x}_n) = e_n \\ \text{in} & e \end{array}$$

valuta l'espressione e nel contesto di una dichiarazione locale di funzioni f_i *mutuamente ricorsive*. Sintassi delle espressioni

$e ::=$	$x \mid f$	nome di variabile o funzione
$\mid$	$e(\overline{e})$	applicazione funzionale
$\mid$	let $\ldots$ in e	espressione con dichiarazione locale

Linguaggi

user (alto livello) $\longrightarrow$ intermediate $\longrightarrow$ machine (basso livello)

- Linguaggi ad alto livello - programmi comprensibili (*specifiche eseguibili*)

* linguaggi logici (PROLOG inizio anni 70')

axioms $p_1(\overline{x}_1) \Leftarrow \phi_1$

$\ldots$

$p_n(\overline{x}_n) \Leftarrow \phi_n$

query $\{\overline{x} \mid g(\overline{x})\}$

genera tutte le tuple $\overline{e}$ t.c. l'*istanza* $g(\overline{e})$ segue dagli *assiomi condizionali*.

* linguaggi d'interrogazione per basi di dati (SQL 1974)

SELECT	nome, cognome	colonne
FROM	Impiegati	tabella
WHERE	salario> 1000	criterio di selezione delle righe

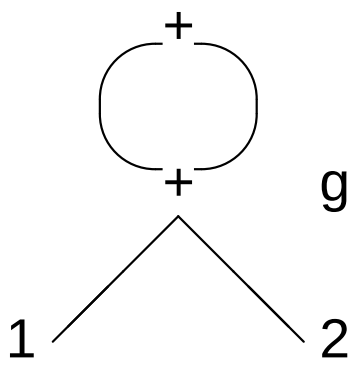
equivalente notazione insiemistica $\{(i.n, i.c) \mid i \in$ Impiegati, $i.s > 1000\}$

Linguaggi

user (alto livello) $\longrightarrow$ intermediate $\longrightarrow$ machine (basso livello)

- Linguaggi ad alto livello - programmi comprensibili (*specifiche eseguibili*)
- Linguaggi intermedi - programmi manipolabili

  * *supercombinatori* e grafi, macchine virtuali per *graph reduction*



grafo che rappresenta let $x = (1 + 2)$ in $(x + x)$

Linguaggi

user (alto livello) $\longrightarrow$ intermediate $\longrightarrow$ machine (basso livello)

- Linguaggi ad alto livello - programmi comprensibili (*specifiche eseguibili*)
- Linguaggi intermedi - programmi manipolabili
- Linguaggi a basso livello - programmi eseguibili

  * linguaggio macchina (o assembler) legato ad una particolare macchina
    - istruzioni eseguibili direttamente dalla CPU
    - manipolazione dei registri della CPU
    - modalita' di indirizzamento della RAM

Linguaggi

user (alto livello) $\longrightarrow$ intermediate $\longrightarrow$ machine (basso livello)

- Linguaggi ad alto livello - programmi comprensibili (*specifiche eseguibili*)
- Linguaggi intermedi - programmi manipolabili
- Linguaggi a basso livello - programmi eseguibili

Struttura e *Fasi* di un Compilatore Ottimizzante

L_{high}

$\downarrow$

front — analisi sintattica, traduzione o error-reporting $\dashrightarrow$ user

L_{int} — analisi/ottimizzazione — machine-independent

$\downarrow$

back — analisi/ottimizzazione, code-generation — machine-dependent $\dashleftarrow$ L_{HDL}

$\downarrow$

L_{low} — analisi/ottimizzazione — machine-dependent

Perle di Saggezza

Perle di Saggezza da Intervista del 16/04/2003 a Frances Allen

- HW and SW Trade-offs (pag 20)

  I ended up understanding so much about the HW and SW trade-offs, because, as a result of our experience with STRETCH, in the ACS project [1961-62], we built the compiler before the machine, in order to be able to design the machine.

- Compilers (pag 21)

  [Compilers] have to be considered at the same time, or ahead of time, because it really is ultimately how the performance gets delivered. . . We've lost a lot of that.

Perle di Saggezza da Intervista del 16/04/2003 a Frances Allen

- Meta-compilers (pag 26)

  I got involved with something called experimental compiling system. . . . use the compiler technology to make compiler writing easier.

- The Ultimate Goal (pag 26-27)

  My goal - a goal I've not achieved - is to SUPPORT LANGUAGES that are useful by the APPLICATION WRITER. Useful by the physicist, useful by the person who i s solving a problem, and have a language or . . . that is NATURAL for the way that person thinks about the problem and the way the person want to EXPRESS THE PROBLEM. . .

  That's what I've always felt was the ultimate role of compilers: to hide all the details of the hardware in the system, still exploit it, but hide that from users, so that they can get on with solving their problem and being comfortable with the results that they were getting, in terms of performance and cost time, and everything else. We've taken a bad direction in languages and compilers.

  What I wanted to do with the experiment on the compiler system, was to build a system that would allow compilers to be built for automated, for multiple kind of source languages, or for multiple target machines.

Perle di Saggezza da Intervista del 16/04/2003 a Frances Allen

- Parallelism (pag 29)

  We got into parallelism as a compiler problem. . . I had a PTRAN group, just a fantastic group of young people. . . The work was a huge stack of papers, which had vast influence on the direction of the field.

  Workshop on Architectures and Compilers for Multithreading
  13-15/12/2007, IIT, Kanpur, India
  Frances Allen: Languages and Compilers for Multicore Computing Systems

Hype

Multicore Computing

- Dana S. Scott (Prof. Emeritus of Comp. Sci., Phil. and Math. Logic at CMU) nel suo discorso per la EATCS Award 2007 identifica il Multicore Computing come la prossima grande sfida per l'Informatica Teorica.

  per il multicore computing c'e' bisogno di co-progettare HW e SW (e linguaggi), o i multicore computers rischiano di rimanere sottoutilizzati.

- multicore computing = il ritorno dei supercomputers
  web computing = il ritorno dei mainframes (per le masse)

Multicore Computing (nella rassegna stampa di ACM TechNews)

- Dana S. Scott (Prof. Emeritus of Comp. Sci., Phil. and Math. Logic at CMU) nel suo discorso per la EATCS Award 2007 identifica il Multicore Computing come la prossima grande sfida per l'Informatica Teorica.

  per il multicore computing c'e' bisogno di co-progettare HW e SW (e linguaggi), o i multicore computers rischiano di rimanere sottoutilizzati.

- multicore computing = il ritorno dei supercomputers
  web computing = il ritorno dei mainframes (per le masse)

- Researchers Ready System to Explore Parallel Computing EE Times (13/03/08)
  Researchers at the University of California, Berkeley are nearly finished building the Berkeley Emulation Engine version 3 (BEE3), an FPGA-based computer that could help researchers find a parallel programming model for advanced multicore processors. BEE3 is intended to help researchers quickly prototype processors with hundreds or thousands of cores and find new ways to program them.. . .
  The system is the centerpiece of the Research Accelerator for Multiple Processors (RAMP) program, a collaborative effort involving Berkeley, Microsoft, Intel, and five other U.S. universities, including MIT and Stanford.. . .

Multicore Computing (nella rassegna stampa di ACM TechNews)

- Dana S. Scott (Prof. Emeritus of Comp. Sci., Phil. and Math. Logic at CMU) nel suo discorso per la EATCS Award 2007 identifica il Multicore Computing come la prossima grande sfida per l'Informatica Teorica.

  per il multicore computing c'e' bisogno di co-progettare HW e SW (e linguaggi), o i multicore computers rischiano di rimanere sottoutilizzati.

- multicore computing = il ritorno dei supercomputers
  web computing = il ritorno dei mainframes (per le masse)

- Industry Giants Try to Break Computing's Dead End New York Times (19/03/08)

  Intel and Microsoft yesterday announced that they will provide 20 million USD over five years to two groups of university researchers that will work to design a new generation of computing systems. The goal is to prevent the industry from coming to a dead end that would halt decades of performance increases in computers.. . . Each lab will work to reinvent computing by developing hardware, software, and a new generation of applications powered by computer chips containing multiple processors. The research effort is partially motivated by an increasing sense that the industry is in a crisis because advanced parallel software has failed to emerge quickly. The problem is that software needed to keep dozens of processors busy simultaneously does not exist.. . .

Multicore Computing (nella rassegna stampa di ACM TechNews)

- Dana S. Scott (Prof. Emeritus of Comp. Sci., Phil. and Math. Logic at CMU) nel suo discorso per la EATCS Award 2007 identifica il Multicore Computing come la prossima grande sfida per l'Informatica Teorica.

  per il multicore computing c'e' bisogno di co-progettare HW e SW (e linguaggi), o i multicore computers rischiano di rimanere sottoutilizzati.

- multicore computing = il ritorno dei supercomputers
  web computing = il ritorno dei mainframes (per le masse)

- Making 'Parallel Programming' Synonymous with 'Programming' HPC Wire (21/03/08)
  Academic experts are engaged in efforts to transform mainstream programming by forcing multiple processing units to cooperate on the performance of a single task, which is being funded by Intel and Microsoft. . . .
  Leading academic teams will focus on developing an effective methodology for multicore processor programming,. . . The UC center's software work concentrates on two different layers,. . .
  The productivity layer will employ abstractions to conceal much of the complexity of parallel programming, while the efficiency layer will allow experts to retrieve the details for maximum performance.

Multicore Computing (nella rassegna stampa di ACM TechNews)

- Dana S. Scott (Prof. Emeritus of Comp. Sci., Phil. and Math. Logic at CMU) nel suo discorso per la EATCS Award 2007 identifica il Multicore Computing come la prossima grande sfida per l'Informatica Teorica.

  per il multicore computing c'e' bisogno di co-progettare HW e SW (e linguaggi), o i multicore computers rischiano di rimanere sottoutilizzati.

- multicore computing = il ritorno dei supercomputers
  web computing = il ritorno dei mainframes (per le masse)

- MicroSoft (SW) e Intel (HW) detengono *2 meta' di una banconota*, ma non sanno come incollarle! Probabilmente, IBM ha un vantaggio strategico, per la sua posizione e le sue competenze nel settore dei supercomputers.

Conclusioni

La Turing Award a Frances Allen ha una doppia lettura:

1. premio alla carriera in un settore (HPC) *di nicchia*, ma *strategico*
2. riconoscimento della rilevanza di sue *idee datate* per affrontare una *grande sfida*!

Dal Vernacoliere del 01/04/1958

Il FORTRAN sfonda!

- Maestrina neo-assunta convince i burberi scienziati ed ingegneri dell'IBM ad adottare in massa il FORTRAN

Il FORTRAN sfonda!

- Maestrina neo-assunta convince i burberi scienziati ed ingegneri dell'IBM ad adottare in massa il FORTRAN
- Le tecniche pedagogiche *non convenzionali* di Frances E. Allen riescono

dove i *diktat* di Thomas J. Watson, Jr. (presidente IBM) hanno fallito!