Corrado Böhm

INFORMATICA: ORA O MAI PIÙ

Per potere valutare appieno i meriti della terna dei miei ex allievi Dezani, Ronchi e Coppo, oggi festeggiati, spero mi scuserete se ho dovuto drammatizzare il titolo di questa relazione, e farla precedere da tre sintesi.

Sintesi storica, dall'inizio a tutto oggi

Superata la crisi del software, a fine secolo, l'informatica è diventata un supporto indispensabile per la *comunicazione* a livello planetario, *fautrice di progresso* in tutte le attività umane, rivolte alla sussistenza, al miglioramento dello stile di vita ed alla ricerca scientifica, filosofica e culturale.

Sintesi storica, fino al 1960

Gli inizi dell'informatica teorica sono stati ammirevoli: nel 1910 Axel Thue inventò un sistema di regole di riscrittura (semi–Thue systems) che lo portava a dimostrare teoremi algebrici in modo quasi meccanico.

Nel 1920-4 M. Schönfinkel introdusse degli oggetti di base, chiamati *combinatori,* per rappresentare termini i quali, soddisfacendo certe regole di riscrittura, dimostravano l'esistenza di entità che godevano di proprietà logiche particolarmente ambite. I combinatori furono argomento di una tesi di dottorato nel 1930 da parte di H. Curry che ne fece oggetto di studio per tutta la sua vita, terminata 50 anni dopo. Nel 1937 A. Turing, inventore delle macchine portanti il suo nome, ospite come dottorando del logico Church, creatore del lambda-calcolo, mostrò che le *sue* macchine potevano calcolare tutte e solo le funzioni rappresentabili con lambda-termini in forma normale (funzioni che furono poi chiamate da Church, e da altri, *ricorsive*).

Dal 1948 in poi ebbe luogo negli Stati Uniti, e subito dopo in Gran Bretagna, la prima produzione dei "computer programmabili", la cui memoria ad accesso rapido era costruita con valvole termoioniche (sostituite ben presto dai transistor) mentre quelle ben più capienti, ad accesso più lento, constavano di tamburi magnetici (a loro volta sostituiti, più tardi, da dischi). Ciò costituiva parte del *hardware*. Invece i programmi costituivano il *software* (ben presto incorporato nei cosiddetti *linguaggi di programmazione* e nei *sistemi operativi*).

Sintesi storica (anche personale), fino al 1970

L'avvento dei computer non fu indolore. Vi furono alcune limitazioni di libertà nei concetti logici. Per esempio la macchina di Turing, la logica combinatoria e il lambda-calcolo permettono la rappresentazione di funzioni d'ordine superiore o *funzionali*, cosa non concessa dai primi linguaggi di programmazione, la cui *sintassi* impediva p.es. che lo stesso nome rappresentasse sia una funzione che un suo argomento. Un altro esempio era la sparizione d'una proprietà tipica delle operazioni del 2° ordine (inventata da Turing), cioè quella di poter mutare non solo gli operandi (qualora si dovesse tornare indietro a rifare un'operazione appartenente ad un ciclo, p. es. nelle operazioni su vettori) ma anche l'operazione stessa. In altre parole occorsero circa 20 anni (fino al 1970) per far sì che la *semantica formale* dei linguaggi di programmazione (introdotta da G.Plotkin, D. Scott, e C. Strachey) predominasse sulle regole sintattiche.

Per poter valutare correttamente il contributo all'informatica delle persone festeggiate devo ora brevemente riferire sul mio curriculum nello stesso periodo.
Dopo essermi laureato in ingegneria elettronica a Losanna (fine del 1946) nel '48 ero assistente al Politecnico di Zurigo all'istituto di Matematica applicata ed avevo il compito di valutare il computer programmabile Z2 con memoria meccanica a relais di Zuse (ingegnere tedesco) e deciderne l'eventuale acquisizione da parte del Politecnico di Zurigo. Constatato che Zuse si avvaleva del computer stesso per stampare il nastro perforato codificante un generico programma da sottoporre più tardi allo stesso computer, mi venne in mente di proporre, come soggetto del mio Dottorato di scienze matematiche, la costruzione d'un singolo universale programma (chiamato, anni dopo, *compilatore*) che fungesse da traduttore d'un qualsiasi programma scritto per i matematici (basato cioè su un algoritmo descritto da regole di riscrittura) nel programma, leggibile soltanto dal computer come sequenza di istruzioni soltanto da esso eseguibili.
Tale programma, essendo scritto nel linguaggio dei matematici, giocava il ruolo d'una macchina universale di Turing e risultò essere il primo compilatore scritto nel linguaggio *sorgente* (consistente allora d'una sequenza di assegnazioni algebrico di valori a variabili, denotanti ognuna un registro di una macchina di Vonn Neumann). L'esame di dottorato fu sostenuto nel 1952 (la tesi fu poi pubblicata nel '54 in francese). Devo riconoscenza al famoso logico prof. Paul Bernays (di cui risulto essere stato il suo 9° dottorando) che mi rese allora edotto dell'esistenza della macchina di Turing.
Nel 1953 vinsi il concorso come ricercatore all'INAC a Roma e nel '68 vinsi quello di professore di prima fascia in Analisi Numerica (la Scienza dell'informazione non era ancora ufficiale in Italia) ottenendo la cattedra a Modena. Fin dal 1960 il mio collega ed amico Wolf Gross mi consigliò di studiare la logica combinatoria di H. B. Curry e il lambda-calcolo di Church, da considerarsi entrambi come basi teoriche della programmazione (da notare che Mc Carthy fino dal 1959 aveva usato nel LISP parte della sintassi del lambda-calcolo). Nel 1964 Gross ed io scrivemmo un lavoro sul CuCh (nome ottenuto ricordando i contributi di *Cu*rry e *Ch*urch), la cui pubblicazione uscì nel 1966.
A questo punto devo citare un teorema del lambda-calcolo, dimostrato nel 1968, che porta il mio nome e che si è rivelato un punto di partenza per le ricerche nel periodo '71-74, anni in cui fui chiamato a dirigere l'Istituto di Scienza dell'Informazione all'Università di Torino.
Quel teorema dimostra che se si forza l'uguaglianza di due lambda-termini irriducibili (cioè in forma normale rispetto alle regole di riduzione del beta-eta calcolo) e differenti fra loro, allora tutti i termini del lambda-calcolo confluiscono in un solo termine. La dimostrazione consiste nel trovare un "contesto" in cui i due termini si trasformano in due termini irriducibili ma "arbitrari".
Appena mi fu possibile, come direttore d'Istituto, cercai d'ottenere collaboratori neo-laureati, intendendo interessarli all'arricchimento semantico del linguaggio CuCh e, fra l'altro, alla gödelizzazione, tramite i *suoi termini irriducibili*, di tutte le strutture di dati, fossero essi aritmetici, algebrici, logici, geometrici, multidimensionali o funzionali di ogni ordine.

Mariangiola Dezani e Simona Ronchi della Rocca furono le prime del susseguente equipaggio di collaboratrici: seguirono, a breve intervallo, Maddalena Zacchi (ancora nel '71) e nel '72 Ines Margaria. Nel '74 venne anche Mario Coppo. La presenza di queste persone mi rende oggi oltremodo felice. Esse si lasciarono guidare da me e, caso abbastanza insolito, la loro fantasia ha arricchito le mie conoscenze.
A questo punto lasciatemi annotare che sono stato fortunato di avere avuto poi "altri" valorosi allievi dai quali ho imparato. Emergono in ordine cronologico:
Giuseppe Jacopini, Marisa Venturini Zilli, Giorgio Ausiello e, in tempi successivi: Giovanna Sontacchi, Silvio Micali, Alessandro Berarducci, Adolfo Piperno, Enrico Tronci, Benedetto Intrigila ed Ugo de Liquoro che tuttavia non hanno ancora raggiunto il traguardo di 60 anni.

I meriti di Mariangiola Dezani
Mariangiola fu in grado, in meno d'un anno, d'affrontare il tema descritto da questi due titoli:

"Can syntax be ignored during translation?" (1972) e
"A CUCH-machine: the automatic treatment of bound variables"(1972).

Nel primo lavoro il mio contributo fu solo di mettere la mia allieva al corrente di:
- una breve nota di Church "Combinatory logic as a semigroup" (1937) ,
- due lavori di Landin "A correspondence between Algol and Church's lambda"
- il fondamentale lavoro di Strachey:"Towards a formal semantics " (1966).

Anch'io avevo imparato qualcosa. Vent'anni dopo, cioè nel 1993, a seguito di una discussione con Berarducci, preparai una conferenza dal titolo: "Equations in combinatory logic." L'osservazione di Church e il teorema di completezza di Curry, diedero luogo a due diversi modelli privilegiati interni (*pim*) del lambda-calcolo.
A rifletterci bene nei pim si nasconde un affascinante concetto di creatività, analogo agli aspetti positivi dei teoremi di Gödel, concetto spesso usato dai matematici, dai fisici e dai cosmologi, quando è dimostrato che un sistema di equazioni è irrisolubile. La scappatoia, almeno negli esempi citati, consiste nel cercare la soluzione in un universo con *almeno* una dimensione in più! (I fisici sono arrivati a 11).
Il secondo lavoro citato consisteva in una prima versione, schematica, di un interprete del linguaggio CuCh.Anche qui fu necessario attendere venti anni, quando diedi come tesi, ad un mio allievo d'ingegneria Stefano Guerrini, la costruzione effettiva d'un interprete per la CUCH-machine Il risultato fu doppio. Immediatamente: un prototipo di programma (successivamente perfezionato da altri studenti) e, dieci anni dopo: un professore d'informatica teorica in più!

Scelgo due altri lavori in collaborazione con Mariangiola, in cui il secondo è un perfezionamento del primo, ma le cui conseguenze - dopo più di trenta anni - mi sembrano ancor più affascinanti di quelle dei due precedenti lavori…
"Combinatorial problems, combinator equations and normal forms" (1974)
" λ-terms as total or partial functions on normal forms" (1975)

Riporto qui 2 modifiche dell'esempio 1 del secondo lavoro dove ho eliminato, per semplicità, le variabili libere prendendo cioè in considerazione solo termini chiusi in forma normale:

(0) λx0 x1 x2.x1(x0 (λx3 x4.x2 x3) (λx3 x4.x4 x3)(x2 x1 x0))

Si noti che l'indice iniziale delle variabili legate è 0 anziché 1, come è stato usato spesso da Henk Barendregt, per semplificare fra l'altro la descrizione dei selettori di n-ple di Church:

Uni = **K**^i **K**^(n-i) = λx0…xn.xi

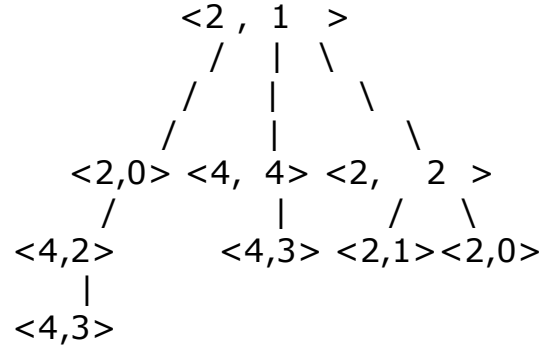
di cui avremo bisogno fra breve. La lettera x è superflua dal punto di vista semantico: tanto vale riscrivere il termine (0) nello stile di de Bruijn dove appare il solo indice della variabile x, così:

(1) λ012.1(0 (λ34.2 3)(λ34.43)(210)).

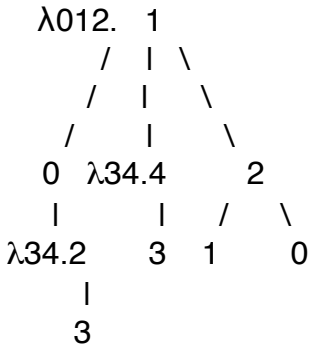
Mariangiola nel '74 (indipendentemente dalla nozione di BT, cioè albero di Böhm, introdotta da Barendregt) propone di rappresentare il termine in forma normale con un albero con i nodi etichettati da coppie.

La radice dell'albero è una coppia il cui primo elemento è il numero di astrazioni iniziali del termine, diminuito di 1; il secondo elemento è il numero che rappresenta la variabile di testa (ciò vale anche per ogni sotto termine). Il numero di figli (in ogni nodo) corrisponde al *grado* della variabile di testa (cioè il numero di argomenti della variabile di testa nel (sotto) termine considerato. Resta da spiegare qual è la prima componente di una coppia in un nodo diverso dalla radice. Ciò si ottiene per induzione ereditaria sui nodi dell'albero già costruito. Se il sottotermine considerato non ha astrazioni iniziali la prima componente è quella del padre, altrimenti vi si aggiunge il loro numero. Così:

(2)



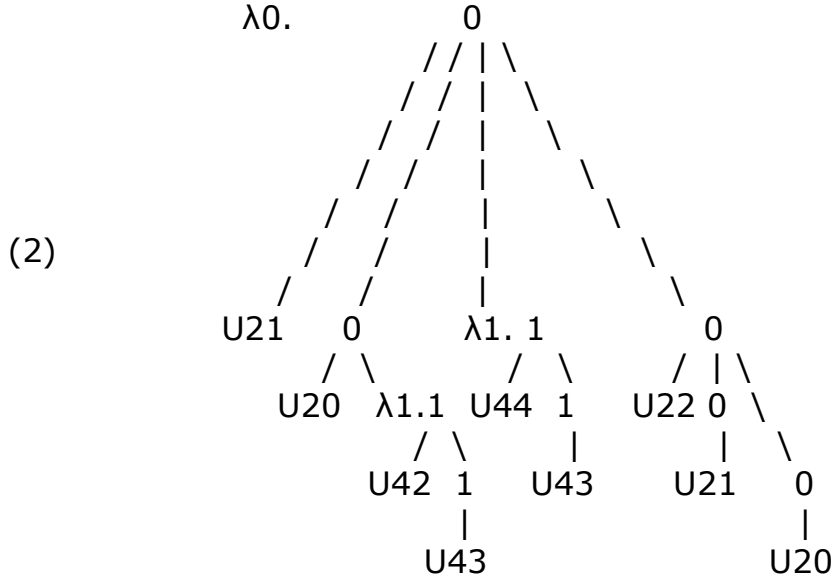
Il grande pregio della presente rappresentazione, che va aldilà del BT della (0) che risulta essere



è da chiarire in questo momento (33 anni dopo la pubblicazione del 1° lavoro di Mariangiola fra i due citati). Si tratta d'un profondo, diverso significato dell'albero (2) da cui si può derivare direttamente il BT del *deed ereditariamente finito (def)* in corrispondenza biunivoca con (1) o (0) (di cui tralascio la dimostrazione).
Qui si esige l'apertura d'una parentesi.

Uno alla volta, per carità!...[da "Largo al factotum" di G.Rossini]
Definizione di def
Un deed D, secondo [R.Statman '89], è un λ-termine chiuso in forma normale con una sola astrazione esterna. Si vede subito che un deed è una semplice estensione d'una t-pla di Church in quanto qui, nelle X0, …,Xn di D = λt.t X0 … Xn,
la variabile t può occorrere libera un qualsiasi numero di volte.
In un sottotermine proprio d'un deed possono però occorrere più nuove astrazioni per volta. Se invece non occorre mai più di una nuova astrazione per volta abbiamo un deed ereditariamente finito (*def*).Lascio alla vostra intuizione l'interesse che può avere la trasformazione che sto per descrivere. Esso è sintetizzato dal motto che ho premesso a questa parentesi: *Uno alla volta, per carità!* In sostanza incontrare meno astrazioni in un termine può significare una diminuzione del numero di sostituzioni, quindi una maggiore efficienza di calcolo...
L'abolizione delle astrazioni multiple all'esterno e all'interno d' un termine chiuso in forma normale espresso come albero delle coppie avviene eseguendo nell' ordine le seguenti istruzioni:
1) sostituire dappertutto <n, i> con il selettore Uni (definito da:
   Uni =λt0....tn. ti)
2) premettere λv0.v0 alla nuova radice U.. ..
3) ad ogni nodo Uni, figlio di λvj.vj Un'i' oppure di vj Un'i', premettere
   λvj+1.vj+1 se n>n', altrimenti vj.

Ecco il risultato sotto forma di BT del *def* corrispondente al termine (0):



Il termine corrispondente è dunque:
λ.0 U21(0 U20(λ1.1 U42(λ1.1 U43)))
(λ1.1U44(λ1.1U43))(0 U22(0 U21)(0 U20)).

Nota I deed ereditariamente finiti hanno un'ulteriore notevole proprietà che si scorge anche dal BT o dal termine appena scritto:
Proprietà dei *def*:
L'*autoapplicazione* di una qualsiasi variabile ove presente in un termine (o sottotermine) iniziale

…. λv.v …. (v…..v ….) …

sparisce nel *def* corrispondente e diventa al più un *autocomposizione*.
Esempio banale: λv.vv diventa λv.v **I**(v **I**).
Ciò facilita l'eventuale tipaggio del termine.
Nota personale
Non vorrei che questa mio suggerimento sui tipi venisse considerato una gaffe da parte mia, verso due persone, Mariangiola e Mario, il cui merito, nelle ricerche sui tipi, sia dal punto di vista teorico che applicativo, non va taciuto ma esaltato.
Tutt'altro: si tratta invece da parte mia, della timida richiesta di perdono per la mia annosa ostinazione, comprovata da numerosi articoli, che per molti anni hanno insistito nel far credere che l'*unico* mezzo per rappresentare le soluzioni di equazioni ricorsive mediante forme normali del λ-calcolo fosse l'uso di variabili *auto-applicate*.

Per terminare l'elogio di Mariangiola, abbandonando quello dei suoi primi risultati in informatica, credo di poter affermare che oggi Ella non è soltanto una grande teorica ma anche una grande esperta e risolutrice a livello planetario dei problemi di sicurezza nella trasmissione e nell'uso di informazioni riservate senza dover ricorrere necessariamente a metodi crittografici. Chi avrebbe mai pensato che partendo dai tipi intersezione si potesse arrivare a simili vette?

I meriti di Simona Ronchi Della Rocca
Simona divenne collaboratrice dell'Istituto qualche mese più tardi di Mariangiola e divenne schiava della mia (allora) passione per ogni specie di dati matematici o logici e funzionali cui davo il nome generico di "strutture informative ".
Perciò i suoi primi lavori furono:
(Con me) "Numbering methods of finite sets and multisets through trees "(1972).
(Con me e Mariangiola) "Questionari per strutture informative" (1974)
"Listing of information structures defined by fixed-point operators" (1974)
La caratteristica dei suoi lavori era una ricerca di cui fosse ben visibile la finalità e che dovesse avere un qualche sbocco applicativo.

I suoi lavori successivi furono quasi sempre in collaborazione, su vari argomenti.
Con Mariangiola e Mario ha scrittoun lavoro sulla " (Semi)-separability of finite sets of terms in Scott's D∞-models of the λ-calculus (1978)" .Specialmente con

Mariangiola ha scritto dei lavori sui tipi. Nel '79, insieme a Mariangiola, P. Peretti e me ha generalizzato il mio teorema applicato ad n>2 termini.
Vorrei segnalare l'ultimo lavoro di mia competenza:
"Discriminability of infinite sets in the D∞-models of the λ-calculus" (1981)
Da questo momento in poi le competenze di Simona si accrescono in svariate direzioni e diventano per me ingovernabili, ma sempre ammirevoli per lo slancio a carattere divulgativo che vi si può scoprire senza perdita di rigorosità.
Per finire voglio solo citare un libro in collaborazione con Luca Paolini dal titolo molto comprensivo: The parametric λ-calculus (Springer 1998).

Un ultimo accenno: Ti invidio oltremodo perchè Tu sei stata responsabile per almeno due anni nella direzione di un corso di master in bioinformatica.
E' proprio la materia che più mi avvince in questo periodo!

I meriti di Mario Coppo
Come già accennato egli diventò collaboratore del nostro Istituto a Torino nel 1974, l'ultimo anno del mio soggiorno a Torino. All'inizio egli frequentò Mariangiola e me .
Il suo primo lavoro, se non erro, fu proprio in collaborazione con Mariangiola e me stesso, dal titolo "Termination test inside λ-calculus" (1977).
Il suo prossimo lavoro fu quello con Mariangiola e Simona , già citato:
"(Semi)-separability of finite sets of terms in Scott's D∞-models of the λ-calculus" (1978).
Il lavoro che segnò il destino di Mario fu, molto probabilmente, il seguente:
Mario Coppo and Mariangiola Dezani–Ciancaglini. A new type–assignment for lambda terms. Archiv für Mathematische Logik, 19:139–156, 1978.
Poi Mario entrò nel *paradiso* dei tipi con particolare riguardo ai tipi ricorsivi.

Per avere un'idea dell'importanza che hanno avuto i nomi Dezani e Coppo e di buona parte degli informatici italiani (torinesi sopratutto) per quanto riguarda i tipi, con particolare riguardo a quelli "intersezione", consiglio di sfogliare le 187 referenze (in continuo aggiornamento) alla voce di Google seguente:
Bibliography for intersection types and related systems.

Termino questa rassegna consigliandovi di fare presto un brindisi ai nostri tre festeggiati scienziati torinesi.