Estratto da

R. Ferro e A. Zanardo (a cura di), *Atti degli incontri di logica matematica* Volume 3, Siena 8-11 gennaio 1985, Padova 24-27 ottobre 1985, Siena 2-5 aprile 1986.

Disponibile in rete su http://www.ailalogica.it

PROGRAMMAZIONE E LOGICA: ESPERIENZE E ASPETTATIVE

INTERVENTO DI
EUGENIO OMODEO (ENIDATA)

C'e` stato un periodo, negli ultimi anni '50 e primi anni '60, in cui i logici hanno lavorato "gomito a gomito" con gli informatici. Erano gli anni nei quali Martin Davis procedeva con slancio ad implementare un algoritmo di decisione per l'aritmetica intera additiva, mentre Dag Prawitz, Hao Wang e lo stesso Davis creavano le basi per la deduzione automatica impostata sul calcolo predicativo clausale. Sempre in quegli anni Abraham Robinson caldeggiava l'impiego degli elaboratori elettronici nelle dimostrazioni matematiche, mentre altri logici proponevano di tradurre in software il metodo di Tarski per il trattamento dell'algebra reale.

In seguito gli scambi fra logici e informatici non sono stati piu` cosi` intensi. E` probabile che, da un lato, gli entusiasmi dei logici che si improvvisavano informatici siano stati frenati dalle costrizioni che il diffondersi di una maggiore disciplina della programmazione e` venuto via via imponendo. Inoltre l'"esplosione combinatoria" dei procedimenti deduttivi implementati si rivelava incontrollabile, quando ancora le modalita` di interazione uomo-macchina erano assai povere e quando non erano ancora disponibili gli strumenti teorici necessari ad un'analisi fine della complessita` degli algoritmi.

Dall'altro lato, quegli informatici che pure credevano nella logica tendevano a sopravvalutare i successi raggiunti (ad es. la scoperta del metodo di risoluzione) e si ritenevano in grado di proseguire nella ricerca con mezzi propri.

Oggi tuttavia sembra che vi siano le premesse per un rilancio della collaborazione fra logica ed informatica. Con una metafora priva di intendimenti polemici direi che il linguaggio Prolog, con tutta la sua miriade di dialetti, e' stato quasi un "cavallo di Troia" della logica nell'informatica. Il caso di Prolog illustra come basti l'aggiunta di pochissimi ingredienti extra-logici a trasformare un formalismo della logica in un linguaggio di programmazione. Poiche' Prolog si presenta ancora inadeguato a sostenere applicazioni di "portata industriale", la Comunita' Europea, attraverso il programma Esprit, sta investendo cifre significative per "immergere" questo linguaggio in un comodo ambiente di sviluppo di programmi e cosi' incoraggiare la diffusione dello stile di programmazione detto "dichiarativo". Esperienze affini a quelle promosse da Esprit riguardo al Prolog sono in corso nel programma britannico Alvey, nonche' in Giappone, in Israele e in Ungheria.

Da dove nasce tutto questo interesse? Accenno solo ad alcuni aspetti. Da un lato, comincia a profilarsi per gli anni '90 una generazione di calcolatori di concezione alquanto diversa dagli attuali ed e' facile prevedere che sara' problematico travasare in queste nuove macchine il software esistente: si rischia di sfruttare poco le risorse in piu' messe a disposizione dalla tecnologia, o di dover riscrivere software gia' lungamente collaudato, con la certezza di introdurvi delle cause di malfunzionamento. Una programmazione di tipo "dichiarativo" come quella esemplificata dal Prolog promette di essere piu' stabile rispetto all'evoluzione tecnologica di quanto non sia la programmazione "procedurale", oggi tanto piu' diffusa. Questa maggiore stabilita' deriva, in parte, dal fatto che la programmazione dichiarativa esenta il programmatore dal descrivere i dettagli minuti dell'esecuzione, il che e' gia' di per se' un gran pregio (sempre che non comporti un significativo rallentamento dei tempi di esecuzione).

Man mano che le applicazioni dei calcolatori diventano piu' complesse e delicate, aumenta negli informatici l'esigenza di strumenti che facilitino la costruzione di programmi affidabili. Due vie di soluzione, sensibili alle attrattive della logica, sono state da molti intraprese: (1) sintesi di algoritmi, attuata a partire dalla descrizione formale dei problemi che ciascun algoritmo deve risolvere; (2) validazione di procedimenti di calcolo alle cui istruzioni siano inframmezzate: un'ipotesi circa i dati in ingresso, una tesi circa il risultato, asserzioni esprimenti certi fatti che il programmatore si aspetta debbano valere quando determinati punti dell'esecuzione vengono raggiunti. Costituisce pero', generalmente, un ostacolo a questi due filoni di indagine la dicotomia esistente fra il formalismo logico (nel quale sono descritti problemi, ipotesi, tesi ed asserzioni) e quello procedurale (in cui sono descritti i "passi" degli algoritmi). La programmazione dichiarativa appare promettente anche per rimuovere questi ostacoli. Essa inoltre consente di mutuare piu' facilmente i modi di inferenza della logica per riutilizzarli come metodi di trasformazione di programmi.

Ora che ho riassunto, telegraficamente, alcune delle ragioni del corrente riavvicinamento dell'informatica alla logica, voglio notare che l'esigenza di approfondimenti della logica si fa particolarmente sentire in un gruppo di lavoro come quello di cui faccio parte presso l'Enidata. Le attivita' del mio gruppo

si imperniano attorno ad alcuni progetti Esprit: uno di questi ha come fine la messa a punto di un ambiente di sviluppo di programmi Prolog, un altro l'integrazione di Prolog con un sistema di gestione di basi di dati. Problemi di tipo "logico" che nascono in questa attivita' sono, per esempio: (1) come dare una semantica della "negazione come fallimento finito" che sia soddisfacentemente nitida e, al tempo stesso, non riduttiva e implementabile come costrutto efficiente? (2) come potenziare l'apparato inferenziale del Prolog in modo che riesca a trattare una ricca famiglia di costrutti modali, introdotti (definendo la semantica di ciascuno di essi) dallo stesso programmatore? (3) come rappresentare ed utilizzare la conoscenza di meta-livello (ossia la conoscenza sulla teoria oggetto e le sue proprieta' - in particolare l'importante relazione di derivabilita') onde accrescere le capacita' deduttive del sistema ed consentire l'espressione di strategie di controllo (o "euristiche")?

Per concludere voglio sottolineare che la comunita' informatica, nel suo attuale protendersi verso la logica, si interessa piuttosto unilateralmente del calcolo predicativo, e trascura teorie piu' espressive come ad esempio la teoria degli insiemi. Non vedo di questo fatto ragioni profonde, ma solo ragioni storiche; penso inoltre che valga la pena di adattare la programmazione dichiarativa a formalismi logici piu' elaborati del calcolo predicativo. Un progetto quinquennale dell'Enidata, finanziato dall'ENI e coordinato con attivita' di ricerca in corso presso varie istituzioni accademiche, in particolare l'Universita' di Catania e la New York University, mi sta fornendo, dal novembre 1985, l'opportunita' di verificare questa mia convinzione.

Bibliografia

[1] Wos L., Overbeek R., Lusk E., Boyle J., Automated reasoning. Introduction and applications. Prentice Hall, 1984.

[2] Bledsoe W.W., Loveland D.W., (Eds.), Automated Theorem Proving: after 25 years. American Mathematical Society, Contemporary Mathematics, Vol.29, Providence (Rhode Island), 1984.

[3] Gallier, J.H., Logic for Computer Science. Foundations of Automated Theorem Proving, Harper & Row, New York, 1986.

[4] Kluzniak F., Szpakowicz S., Prolog for Programmers, Academic Press, 1985.