Bollettino
Unione Matematica Italiana

Sezione A – La Matematica nella Società e nella Cultura

Roberta Gori

Ragionando sul fallimento finito e le computazioni infinite usando l'interpretazione astratta

Bollettino dell'Unione Matematica Italiana, Serie 8, Vol. **3-A**—*La Matematica nella Società e nella Cultura* (2000), n.3, p. 351–354.

Unione Matematica Italiana

<http://www.bdim.eu/item?id=BUMI_2000_8_3A_3_351_0>

Bollettino dell'Unione Matematica Italiana, Unione Matematica Italiana, 2000.

Bollettino U. M. I.
La Matematica nella Società e nella Cultura
Serie 8, Vol. III-A, Dicembre 2000, 351-354

Ragionando sul fallimento finito e le computazioni infinite usando l'interpretazione astratta.

Roberta Gori

Eleganza e semplicità sono sicuramente tra le caratteristiche più importanti della programmazione logica. Il linguaggio è basato su un frammento della logica del prim'ordine: la logica delle clausole Horn. Per questo un programma logico può essere considerato come una teoria logica e di conseguenza il suo significato (la sua semantica) può facilmente essere descritto usando le nozioni standard della semantica dichiarativa della logica classica. Van Emden and Kowalski nel loro articolo del 1976, [4], hanno dimostrato l'equivalenza tra il sistema di prova basato sulla risoluzione SLD, la caratterizzazione in termini di modelli (il modello considerato era il minimo modello di Herbrand) e la caratterizzazione di punto fisso. Secondo questa visione il problema di definire la semantica di un programma logico è stato risolto una volta per tutte definendo *la* semantica in termini di modelli che può essere calcolato con una costruzione top-down (l'insieme di successo ground) e con una costruzione bottom-up (il minimo punto fisso di dell'operatore delle conseguenze immediate). Malgrado questa soluzione sembri piuttosto soddisfacente, dovremmo però riflettere su quale sia l'uso principale della semantica di un programma. Il principale utilizzo della semantica è quello di chiarire il significato del programma. Questo concetto è in effetti alla base di tutte le altre applicazioni della semantica come la trasformazione di programmi, la verifica e l'analisi.

Il problema nasce in quanto la semantica dichiarativa scelta non è in grado di caratterizzare molte proprietà importanti della programmazione logica. Intuitivamente questo accade perché la corrispondenza tra l'interpretazione operazionale e quella dichiarativa è ottenuta assumendo un comportamento operazionale piuttosto povero cioé basato solo sulla nozione di insieme di successo. Ma l'insieme di successo non riesce a catturare neanche il principale aspetto della programmazione logica, che è poi la principale caratteristica che differenzia la programmazione logica dalla dimostrazione automatica: le risposte calcolate. Un programma logico infatti *calcola* sostituzioni. Per questo ragionare sul suo comportamento operazionale in termini di minimo modello di Herbrand sembra del tutto inadeguato. Inoltre altre classi di proprietà rilevanti come il fallimento finito non possono essere espresse usando il minimo modello di Herbrand. Per questo motivo, per ragionare correttamente e accuratamente sui programmi logici, è spesso necessario essere in grado di caratterizzare diversi aspetti operazionali della procedura di prova (la risoluzione SLD).

È stata quindi abbandonata l'idea che un programma logico potesse avere una semantica che ne descriveva tutti i vari aspetti e si è fatta strada l'idea che si potessero definire *diverse* semantiche che modellano proprietà diverse della computazione (proprietà osservabili).

Per formalizzare il concetto di correttezza di una semantica rispetto ad una data

proprietà è necessario definire il concetto di equivalenza osservazionale indotto sui programmi dall'avere lo stesso comportamento rispetto alla proprietà scelta.

DEFINIZIONE 1. – *Data una proprietà osservabile x, essa induce un'equivalenza osservazionale sui programmi. $P_1 \approx_x P_2$ se e solo se P_1 e P_2 sono osservazionalmente indistinguibili rispetto alla proprietà x.*

Una volta definita l'equivalenza indotta da una proprietà x, definiamo quando una semantica è corretta rispetto a tale proprietà.

DEFINIZIONE 2. – *Una semantica $\mathcal{S}(P)$ è* corretta *rispetto a $\approx_x$, se $\mathcal{S}(P_1) = \mathcal{S}(P_2)$ implica $P_1 \approx_x P_2$.*

Se vale anche l'implicazione inversa allora la semantica è fully abstract.

Una semantica $\mathcal{S}(P)$ è fully abstract *rispetto a $\approx_x$, se $P_1 \approx_x P_2$ implica $\mathcal{S}(P_1) = \mathcal{S}(P_2)$.*

Oltre ad essere corretta e fully abstract rispetto ad una data proprietà x, una semantica può avere la proprietà di essere composizionale rispetto agli operatori sintattici. Nel caso della programmazione logica particolare rilevanza riveste la proprietà di AND-composizionalità, i.e. $\mathcal{S}(A \wedge B) = \mathcal{S}(A)\ \sigma(\wedge)\ \mathcal{S}(B)$. Questa proprietà ci assicura che la semantica di un goal composto può essere derivata dalla semantica dei goals che lo compongono, e quindi ci permette di definire denotazioni per i soli goals atomici.

Seguendo questo approccio negli ultimi anni sono state definite diverse semantiche capaci di modellare proprietà delle computazioni di successo, come la s-semantica che modella le risposte calcolate, semantiche per modellare i call pattern, i risultanti, etc....

Questa tesi nasce dall'esigenza di definire semantiche che modellino aspetti rilevanti per la programmazione logica che però non possono essere visti come proprietà delle derivazioni di successo: il *fallimento finito* e le *computazioni infinite.*

La prima semantica introdotta in letteratura per modellare il fallimento finito, è stato l'insieme di fallimento finito ground FF_P [1], che è definito come l'insieme di atomi ground che hanno un albero SLD fallito finitamente con una regola di selezione fair. È possibile mostrare con un semplice controesempio che tale semantica non è corretta rispetto all'equivalenza indotta sui programmi dall'avere lo stesso insieme di fallimento finito ($\approx_{FF}$). Successivamente in [3], è stata definita una nuova denotazione per modellare il comportamento del fallimento finito: $NGFF_P$, definito come l'insieme di atomi che hanno un albero SLD fallito finitamente con una regola di selezione fair.

Il primo risultato di questa tesi dimostra che $NGFF_P$ è una semantica corretta per il fallimento finito.

TEOREMA 1. – $NGFF_P = NGFF_Q \Rightarrow P \approx_{FF} Q$.

Si dimostra inoltre che la semantica $NGFF_P$ è anche fully abstract. Si è anche ottenuto un primo importante risultato per dimostrare l'And-composizionalità

della denotazione $NGFF_P$. Completando la base di Herbrand per ideali, si dimostra infatti che l'insieme degli elementi massimali del complemento dell'insieme di fallimento finito non ground (esteso al completamento della base di Herbrand per ideali), $Max(\overline{NGFF_P})$, è And-composizionale.

TEOREMA 2. – *Il goal* $G = \leftarrow A_1, \ldots, A_n$ *ha un albero fallito finitamente in* P *se e solo se* $\nexists$ *un ideale di sostituzioni* $\Theta : A_i\Theta \in Max(\overline{NGFF_P})$ *per* $i = 1, \ldots, n$.

Il precedente risultato implica che anche la semantica $NGFF_P$ è And-composizionale.

Purtroppo però per l'insieme di Fallimento Finito Non-Ground non esisteva alcuna caratterizzazione di punto fisso. La mancanza di una tale caratterizzazione ha fondamentalmente due svantaggi. Il primo è che non esiste un modo per calcolare la semantica $NGFF_P$. Come secondo svantaggio abbiamo che non è possibile applicare al fallimento finito i metodi di analisi e verifica induttiva basati sull'esistenza di una caratterizzazione di punto fisso. Per questo motivo il passo successivo è stato quello di cercare una caratterizzazione di punto fisso per la denotazione $NGFF_P$. Si noti che una tale caratterizzazione non era facile da definire in maniera diretta. Basti pensare che lo stesso insieme di fallimento finito ground FF_P era stato definito solo in maniera dichiarativa. L'idea è stata quella di derivare un operatore di punto fisso per il fallimento finito usando la teoria dell interpretazione astratta.

L'interpretazione astratta è una teoria introdotta da Cousot e Cousot nel 1979 che permette, tramite le inserzioni di Galois, di formalizzare in maniera precisa il concetto di approssimazione tra semantiche. Tramite le funzioni di astrazione e concretizzazione la teoria dell'interpretazione astratta ci fornisce un metodo formale per passare da una semantica più concreta ad una più astratta che contiene quindi meno informazione. Per questo, l'interpretazione astratta è stata ampiamente utilizzata per mettere in relazione semantiche diverse definite in letteratura. Recentemente in [2] è stato definito un framework basato sull'interpretazione astratta nel quale partendo da una semantica molto concreta capace di modellare tutte le derivazioni di successo e data una proprietà di tali derivazioni è possibile definire sistematicamente una semantica corretta rispetto a tale proprietà. La semantica così definita può essere calcolata come punto fisso di un opportuno operatore astratto ottenuto sistematicamente da quello concreto. Seguendo questo approccio l'idea è stata quella di usare l'interpretazione astratta per derivare sistematicamente una semantica di punto fisso che modellasse il fallimento finito partendo da una semantica molto concreta che ci permettesse di osservare tale comportamento.

Per raggiungere questo scopo il precedente framework è stato esteso in modo che la semantica concreta modellasse non solo le derivazioni di successo ma *tutte* le possibili derivazioni SLD (eventualmente infinite) con una regola di selezione fair. Si sono stabilite poi condizioni sufficienti sugli operatori concreti e sulle funzioni di astrazione e concretizzazione perchè la semantica astratta sistematicamente definita e calcolata come massimo punto fisso di un opportuno operatore astratto, avesse tutte le proprietà desiderate come la correttezza

rispetto alla proprietà, la composizionalità rispetto all'AND e la co-continuità dell'operatore astratto.

Una volta definito il framework è stato possibile applicare questa costruzione per derivare una caratterizzazione di punto fisso per l'insieme $NGFF_P$ basata su un operatore co-continuo.

TEOREMA 3. – $NGFF_P = gfp(T_P^{ff}) = \bigwedge_{i \leqslant \omega} T_P^{ff} \downarrow i$.

È interessante notare che tramite questa costruzione sistematica si è anche ottenuto un nuovo risultato di AND-composizionalità per $NGFF_P$ molto più semplice del precedente. Abbiamo quindi affrontato il problema della verifica induttiva di programmi rispetto alla proprietà del fallimento finito. L'idea della verifica induttiva è quella di definire condizioni sufficienti e effettive per provare che un programma si comporti (rispetto ad una data proprietà) come descritto da una specifica. Per ottenere questo scopo l'idea è di provare la relazione $\mathcal{S}(P) \leqslant I$ che esprime la condizione che la semantica del programma soddisfi la specifica I. In particolare, tutte le volte che la semantica di un programma P possa essere calcolata come punto fisso di un operatore T_P, è possibile stabilire condizioni di verifica sufficienti in termini dell'operatore T_P e della specifica I, le quali non richiedono alcun calcolo di punto fisso. In questa tesi, seguendo l'approccio proposto da Ferrand, basato su due diverse specifiche, abbiamo definito, tramite due diverse approssimazioni finite (ottenute con tecniche di interpretazione astratta), condizioni sufficienti ed effettive per dimostrare che un programma P è corretto rispetto agli atomi (non più profondi di una costante k) che falliscono finitamente e rispetto a quelli che hanno successo.

Infine il Framework basato sull'interpretazione astratta è stato applicato per definire una semantica di punto fisso (basata su un operatore co-continuo) che modella le *risposte esatte* di derivazioni infinite. Alcune approssimazioni effettive (ottenute con tecniche di interpretazione astratta) di questa semantica sono state mostrate utili per definire analisi di terminazione per i programmi logici.

BIBLIOGRAFIA

[1] APT K. R. e VAN EMDEN M. H., *Contributions to the theory of logic programming*, Journal of the ACM, **29(3)** (1982), 841-862.

[2] COMINI M., LEVI G. e MEO M. C., *A theory of observables for logic programs*, Information and Computation. To appear.

[3] LEVI G., MARTELLI M. e PALAMIDESSI C., *Failure and success made symmetric*, Proc. North American Conf. on Logic Programming (1990), 3-22.

[4] VAN EMDEN M. H. e KOWALSKI R. A., *The semantics of predicate logic as a programming language*, Journal of the ACM, **23(4)** (1976), 733-742.

Dipartimento di Informatica, Università di Pisa
e-mail: gori@di.unipi.it
Dottorato in Logica Matematica e Informatica Teorica
(Sede amministrativa: Siena) - Ciclo XI
Direttore di ricerca: Prof. Giorgio Levi, Università di Pisa