Università degli Studi della Calabria

Dipartimento di Matematica

Dottorato di Ricerca in Matematica ed Informatica

XX Ciclo

Settore Disciplinare INF/01 INFORMATICA

Tesi di Dottorato

Answer Set Programming with Functions

Susanna Cozza

Supervisore	**Coordinatore**
Prof. Nicola Leone	Prof. Nicola Leone

Anno Accademico 2007 - 2008

Answer Set Programming with Functions

Susanna Cozza

Dipartimento di Matematica,
Università della Calabria
87036 *Rende, Italy*
email : cozza@mat.unical.it

Sommario

L'Answer Set Programming (ASP) [Gelfond and Lifschitz, 1988; 1991] è un formalismo che si è affermato, in questi ultimi anni, come potente strumento di programmazione dichiarativa per la rappresentazione della conoscenza e per il ragionamento non monotono. La semantica su cui è basato (denominata *semantica answer set*) estende la semantica dei modelli stabili (usata per i programmi logici 'normali') all'ambito della Programmazione Logica Disgiuntiva (PLD), in cui oltre alla negazione non monotona è consentito anche l'uso della disgiunzione nella testa delle regole.

Nell'ambito dell'ASP, un determinato problema computazionale viene rappresentato tramite un programma PLD che può avere zero o più modelli alternativi chiamati *answer sets*, ognuno dei quali corrisponde ad una possibile visione del dominio modellato.

I linguaggi ASP consentono di rappresentare, in maniera semplice e naturale [Eiter *et al.*, 2000], varie forme di ragionamento non monotono, planning, problemi diagnostici e, più in generale, problemi di elevata complessità computazionale [Baral and Gelfond, 1994; Lobo *et al.*, 1992; Wolfinger, 1994; Minker, 1994; Lifschitz, 1996; Eiter *et al.*, 1999; Baral, 2003].

Dopo i primi sistemi sperimentali sul calcolo dei modelli stabili [Bell *et al.*, 1994; Subrahmanian *et al.*, 1995], attualmente sono disponibili un buon numero di sistemi che supportano in maniera efficiente l'ASP [Gebser *et al.*, 2007a; Leone *et al.*, 2006; Janhunen *et al.*, 2006; Lierler, 2005; Lin and Zhao, 2004; Simons *et al.*, 2002; Anger *et al.*, 2001; East and Truszczyński, 2001; Egly *et al.*, 2000]. Tra questi, quelli di maggiore successo sono: DLV [Leone *et al.*, 2006], `GnT` [Janhunen *et al.*, 2006] (estensione PLD del più conosciuto Smodels [Simons *et al.*, 2002]), e più di recente *clasp* [Gebser *et al.*, 2007a]. Tutto ciò, se da un lato ha consentito l'impiego dell'ASP in contesti reali, dall'altro ha messo in evidenza le limitazioni dei sistemi attualmente disponibili.

Uno dei principali limiti è dato dall'inadeguato supporto a termini complessi quali *funzioni*, *liste*, *set* ed in generale costrutti che consentano, in maniera diretta, il ragionamento su strutture dati ricorsive e domini infiniti. Infatti, anche se la semantica answer set è stata definita per linguaggi del primo ordine 'generali' quindi con anche termini funzionali, i sistemi ASP esistenti, sono essenzialmente basati su linguaggi senza funzioni. L'esigenza di estendere la PLD con un adeguato supporto ai termini funzionali è percepita fortemente dalla comunità scientifica ASP, come testimoniato dai numerosi contributi pubblicati recentemente su questo tema [Lin and Wang, 2008; Simkus and Eiter, 2007; Baselice *et al.*, 2007; Bonatti,

2004; Syrjänen, 2001]. Tuttavia, non è stato ancora proposto un linguaggio sufficientemente soddisfacente dal punto di vista linguistico (che abbia espressività elevata) ed anche adatto ad essere integrato nei sistemi ASP esistenti. Infatti, attualmente nessun sistema ASP consente un uso effettivo di termini funzionali. I simboli di funzione o sono del tutto banditi dal linguaggio oppure sono soggetti a forti vincoli sintattici che non ne consentono l'uso in ambito ricorsivo.

Obiettivo di questo lavoro di tesi è il superamento di tali limitazioni. Il contributo del lavoro è sia teorico che pratico ed ha portato all'implementazione di un sistema ASP che supporta adeguatamente i termini complessi: funzioni ma anche liste ed insiemi.

Il principale contributo teorico riguarda la definizione formale di una nuova classe di programmi PLD: i programmi *finitamente ground* ($\mathcal{FG}$). Tale classe supporta in maniera completa i simboli di funzione (essi sono ammessi anche in caso di regole ricorsive) e gode di alcune rilevanti proprietà computazionali. Viene infatti dimostrato che, per tutti i programmi appartenenti alla classe, la valutazione secondo un approccio bottom-up, consente il calcolo effettivo degli answer set e di conseguenza, sia le query ground che le query non ground sono decidibili, qualsiasi sia la forma di ragionamento scelto (coraggioso o cauto). Un ulteriore risultato riguarda l'espressività di questa classe: viene dimostrato che, qualsiasi funzione calcolabile può essere espressa tramite un programma $\mathcal{FG}$. Chiaramente, questo implica che il problema di riconoscere se un programma appartiene o meno alla classe, risulta essere indecidibile (in particolare, semidecidibile). Poichè per alcuni utenti/applicazioni è necessario avere una garanzia ''a priori'' della terminazione del programma, si è ritenuto utile identificare una sottoclasse dei programmi $\mathcal{FG}$, per la quale anche il problema del riconoscimento sia decidibile. Alcune condizioni sintattiche sono quindi state individuate come caratterizzanti per la nuova classe: i programmi *a dominio finito* ($\mathcal{FD}$).

La classe dei programmi $\mathcal{FG}$ è, per alcuni aspetti, complementare alla classe dei programmi *finitari* [Bonatti, 2004]. La prima segue un approccio bottom-up, mentre la seconda top-down. Le due classi risultano essere incomparabili, nel senso che esistono dei programmi che appartengono alla prima ma non alla seconda e viceversa. Al fine di rendere valutabili secondo l'approccio bottom-up i programmi finitari, ampliando così la classe dei programmi $\mathcal{FG}$, si è fatto ricorso alla tecnica dei *Magic Sets*. Tale tecnica, nata nell'ambito delle Basi di Dati Deduttive a scopo di ottimizzazione, consiste in un metodo di riscrittura che simula la valutazione top-down di una query, modificando il programma originale e ag-

giungendo nuove regole che limitano la computazione ai dati rilevanti ai fini della query. Un opportuno adeguamento dell'algoritmo di riscrittura Magic Sets è stato definito per le query finitamente ricorsive (le query il cui programma 'rilevante' è finitario ed è positivo). Dato un programma $\mathcal{P}$ ed una query ground finitamente ricorsiva $\mathcal{Q}$, i seguenti risultati sono stati dimostrati riguardo il programma riscritto $RW(\mathcal{Q}, \mathcal{P})$: la sua dimensione è lineare rispetto alla dimensione dell'input; risulta essere del tutto equivalente a quello originario ai fini della risposta alla query e soprattutto, tale programma è $\mathcal{FG}$, quindi può essere valutato bottom-up (contrariamente al programma originale).

Oltre ai contributi teorici sinora descritti, questa tesi presenta anche i risultati di un'attività più pratica, di tipo implementativo e di sperimentazione. Tale attività è stata svolta utilizzando DLV come sistema ASP di riferimento ed ha avuto come obiettivo l'estensione del linguaggio con simboli di funzione e più in generale termini complessi quali liste e set.

Un primo passo in questa direzione è stata la realizzazione del supporto a funzioni esterne (plug-in). L'utente del sistema può definire dei propri particolari predicati esterni (identificati tramite il prefisso '$\#$'), implementati come funzioni C++ ed utilizzabili all'interno di un programma DLV. Ad ogni predicato esterno $\#p$ di arità n deve essere associata almeno una funzione C++ p', detta 'oracolo'. Tale funzione, della cui definizione è responsabile l'utente, deve restituire un valore booleano e deve avere esattamente n argomenti. Un atomo ground $\#p(a1, \ldots, an)$ sarà vero se e solo se il valore restituito dalla funzione $p'(a1, \ldots, an)$ è vero. Gli argomenti di una funzione oracolo possono essere sia di input che di output: ad argomenti di input corrispondono termini costanti o a cui è già stato assegnato un valore (termini 'bound') nel relativo predicato esterno, ad argomenti di output corrispondono termini variabili non legati ad alcun valore (termini 'unbound').

L'introduzione delle funzioni esterne nel sistema DLV ha come conseguenza la possibilità di introdurre nuove costanti simboliche nel programma logico (Value Invention) e quindi rendere eventualmente infinito l'universo di Herbrand. Di conseguenza, per i programmi con Value Invention, non è più garantita la terminazione. Sono state quindi individuate un insieme di condizioni sintattiche tali da garantire la proprietà di 'istanziazione finita' e quindi la terminazione, per un programma logico con funzioni esterne. Un riconoscitore di programmi logici che rispettano tali condizioni è stato inoltre implementato ed integrato in DLV.

Sfruttando le possibilità offerte dai predicati esterni, è stato realizzato il sup-

porto a termini complessi di tipo funzionale. In pratica, in presenza di termini funzionali, una regola DLV viene 'riscritta' in maniera tale da tradurre la presenza del simbolo di funzione, nell'invocazione di un opportuno predicato esterno; questo si occupa poi della costruzione del termine complesso o, viceversa, dell'estrazione degli argomenti da esso. Il risultato ottenuto è che i programmi $\mathcal{FG}$, descritti in precedenza, sono pienamente supportati dal sistema. Un riconoscitore sintattico per i programmi a dominio finito consente di individuare ''a priori'' quei programmi la cui terminazione non è garantita. A discrezione dell'utente, è possibile disabilitare tale riconoscitore e sfruttare in pieno l'espressività dei programmi $\mathcal{FG}$, rinunciando però alla garanzia di terminazione. Il linguaggio effettivamente supportato dal sistema, oltre ai termini funzionali, prevede anche altre due tipologie di termini complessi: liste e set. Realizzati sfruttando lo stesso framework usato per le funzioni, liste e set sono stati corredati da una ricca libreria di funzioni di manipolazione che rendono ll loro uso molto agevole. Grazie a tali estensioni, il linguaggio di DLV è diventato sempre più adatto alla codifica immediata e compatta di problemi di rappresentazione della conoscenza.

In sintesi, i contributi principali di questo lavoro di tesi sono:

- la definizione formale della classe dei programmi finitamente ground; cioè una nuova classe di programmi logici disgiuntivi che consente l'uso dei simboli di funzione (eventualmente con ricorsione) e che gode di rilevanti proprietà computazionali quali: l'elevata espressività, la computabilità bottom-up degli answer set e quindi la decidibilità del ragionamento, anche in caso di query non ground;

- l'utilizzo della tecnica dei Magic Sets per la riscrittura di programmi finitari positivi che non risultano appartenere alla classe dei programmi finitamente ground, ma sui quali è comunque possibile valutare bottom-up delle query ground, grazie ad un'opportuna trasformazione del programma stesso;

- la realizzazione di un sistema il cui linguaggio, oltre a supportare pienamente la classe dei programmi finitamente ground, consente l'integrazione nel programma logico di sorgenti computazionali esterne e l'utilizzo agevole di termini complessi quali liste e set;

- la comparazione del nostro lavoro, in particolar modo con i programmi finitari, ma anche con altri approcci proposti in letteratura per l'estensione dei sistemi ASP con simboli di funzione.