10 PROGRAMMAZIONE LOGICA CON VINCOLI

1

Jaffar & Lassez, Constraint logic programming, Proc. ACM Symp. POPL 1987

Jaffar & Maher, Constraint logic programming; A Survey, Journal of Logic Programming 19-20 (1994)

PROGRAMMAZIONE LOGICA CON VINCOLI: MOTIVAZIONI

Constraint Logic Programming (CLP)

- la programmazione logica pura è stata estesa in numerose direzioni
  - programmazione funzionale
  - programmazione orientata-ad-oggetti
  - ....

CLP invece che ADALOG

- lo stesso Prolog contiene delle primitive (= operazioni su domini "interpretate"), che non hanno una semantica dichiarativa e non si comportano come vere relazioni.

- alcune estensioni di Prolog (i.e. Prolog II) aggiungono nuovi oggetti ai termini, come gli alberi razionali (soluzioni di equazioni cicliche), con relazioni come l'uguaglianza e la disuguaglianza

PROGRAMMAZIONE LOGICA CON VINCOLI: MOTIVAZIONI

- perché non importare nella programmazione logica
    - il paradigma dei sistemi di soddisfacimento di vincoli (dell'A.I.)
    - particolari metodi ed algoritmi di soluzione per vincoli su domini particolari (della Ricerca Operativa, della Computer Algebra)

- perché, in fondo, il meccanismo del rispondere con delle sostituzioni (o equazioni) è limitativo
    - il paradigma diventa più generale e potente se permettiamo risposte implicite, rappresentate per l'appunto da vincoli

$$X < Y \;\&\; Y \geq 2.75$$

sul dominio dei reali non è rappresentabile con una sostituzione

COME SI ARRIVA A CLP
PARTE 1

(ovvero: l'algoritmo di unificazione ridimensionato)

- la programmazione logica classica si basa sul calcolo dell'mgu tra termini (in realtà calcolo delle soluzioni dell'equazione $t_1 = t_2$)
- l'unificazione fa due cose
  - funziona da algoritmo di decisione sulla risolubilità dell'equazione (cioè decide se esistono soluzioni)
  - semplifica l'equazione, trasformandola in forma risolta e rappresentando in forma esplicita (insiemi di) soluzioni.
- la seconda cosa non è realmente essenziale ma è semplicemente un fatto di "presentazione dei risultati"
  - le soluzioni determinate a partire dalla forma risolta (o equivalentemente) dal corrispondente mgu idempotente sono esattamente le stesse dell'equazione iniziale
- separiamo concettualmente le due funzioni
  - un algoritmo che determina la risolubilità (essenziale!)
  - un algoritmo di semplificazione di insiemi di equazioni risolubili (un semplice miglioramento dell'implementazione!)

COME SI ARRIVA A CLP
PARTE 2: LA VENDETTA

(ovvero: l'universo di Herbrand dimenticato)

- la programmazione logica classica ha come dominio dei dati l'Universo di Herbrand
  - calcola lì sopra con l'algoritmo di unificazione, ma, appoggiandoci
    "calcola costruendo insiemi di equazioni risolubili"

- rimpiazziamo l'Universo di Herbrand con altro (altri) dominio (domini)

  e

- rimpiazziamo il "calcolare costruendo insiemi di equazioni risolubili" con
  "calcolare costruendo insiemi di vincoli (sui nuovi domini) risolubili"

- così si ottiene CLP(?): lo schema
  - uno dei domini può anche essere il vecchio universo di Herbrand con uguaglianza come vincoli
    - la programmazione logica classica è un caso particolare
  - i problemi elencati nelle motivazioni sono tutti risolubili elegantemente all'interno di questo schema

IL LINGUAGGIO

una clausole CLP

$$a \leftarrow \underbrace{c_1, c_2, \dots, c_n}_{\text{vincoli}} \,\Box\, b_1, \dots, b_m .$$

- $c_1, c_2, \dots, c_n$ (i vincoli) sono formule atomiche i cui predicati sono interpretati su una particolare struttura algebrica (many sorted) $\mathcal{R}$
- non sono definiti da altre clausole!
- $\mathcal{R}$ può contenere il tradizionale universo di Herbrand (un particolare sort)
- l'insieme dei predicati interpretati contiene l'uguaglianza (definita per tutti i sorts)

$$
\begin{aligned}
p(X, Y, Z) \leftarrow\; & Z = X * X1 + Y * Y1, \\
& X1 \geq 0, \\
& Y1 \geq 0 \,\Box \\
& q(X, Y, X1, Y1).
\end{aligned}
$$

unico sort, numeri reali

LA REGOLA DI RISOLUZIONE

un goal CLP

$$\leftarrow c_1, \dots, c_m \,\square\, b_1, \dots, b_n.$$

(con $c_1 \wedge \dots \wedge c_m$ risolubile in $\mathcal{R}$)

- la risoluzione con vincoli

$$\leftarrow c_1, \dots, c_m \,\square\, g_1, \dots, g_k, \dots, g_n.$$

$$a^1 \leftarrow c^1_1, \dots, c^1_{m_1} \,\square\, b_1, \dots, b_{m_1}$$

$$\leftarrow c_1, \dots, c_m, g_k = a^1, c^1_1, \dots, c^1_{m_1} \,\square\, g_1, \dots, g_{k-1}, \overbrace{b_1, \dots, b_{n_1}} g_{k+1}, \dots, g_n$$

  - $c_1 \wedge \dots \wedge c_m \wedge g_k = a^1 \wedge c^1_1, \dots, c^1_{m_1}$ deve essere risolubile in $\mathcal{R}$

- risoluzione senza unificazione

- unificazione (nella sua sola versione di procedura per decidere la risolubilità) rimpiazzata da un risolutore di vincoli su $\mathcal{R}$

LA DERIVAZIONE

- si estendono naturalmente le definizioni date nel caso dei programmi logici positivi
  (regola di selezione, varianti delle clausole, etc.)

- una derivazione $(G_0, G_1, \ldots, G_k)$ termina con successo se
  G_k ha la forma

  $$\leftarrow c_1, \ldots, c_m \blacksquare$$

- (cioè è composto solo da vincoli)
  - le formula $c_1 \wedge \ldots \wedge c_m$ viene detta <u>vincolo di risposta</u> (answer constraint)
    (generalizzazione della sostituzione di risposta calcolata)

- una derivazione $G_0, \ldots, G_k$ <u>fallisce finitamente</u> se non ha avuto successo e G_k non ha nessun risolvente nel programma

VINCOLI DI RISPOSTA: UN ESEMPIO

$$
\begin{aligned}
& p(X, Y, Z) \leftarrow \\
& \quad Z = X * X1 + Y * Y1, \\
& \quad X1 \geq 0, \\
& \quad Y1 \geq 0, \\
& \quad q(X, Y, X1, Y1).
\end{aligned}
$$

$$
q(X, Y, X, Y) \leftarrow
$$

$$
\begin{aligned}
& \leftarrow p(0.2, 0.1, Z) \\
& \leftarrow p(0.2, 0.1, Z) = p(X', Y', Z'), \\
& \quad Z' = X' * X_1' + Y' * Y_1', \\
& \quad X_1' \geq 0, \\
& \quad Y_1' \geq 0, \\
& \quad q(X', Y', X1', Y_1'). \\
& \leftarrow p(0.2, 0.1, Z) = p(X', Y', Z'), \\
& \quad Z' = X' * X_1' + Y' * Y_1', \\
& \quad X_1' \geq 0,\ Y_1' \geq 0,\ q(X', Y', X_1', Y_1') = q(X^2, Y^2, X^2, Y^2)
\end{aligned}
$$

è il vincolo di risposta: il risolutore di vincoli nel dominio dei reali ci dice che è risolubile

- può anche essere così astuto (ma è solo un fatto di implementazione!) da semplificare il vincolo e restituire (una versione informale delle restrizioni alle variabili del goal iniziale)

$$Z = 0.05$$

VINCOLI DI RISPOSTA: UN ALTRO ESEMPIO


```
p(X,Y,Z) <-
    Z = X*X1 + Y*Y1,
    X1 > Y1,
    q(X,Y,X1,Y1).
q(X,Y,X,Y) <-
```

$\leftarrow p(0.2, 0.1, Z)$

$\leftarrow Z = 0.05$ (con il risolutore di vincoli astuto)

$\leftarrow p(X, Y, Z)$

$\leftarrow Z = X*X + Y*Y,\ X > Y$

- solo una parte del vincolo di risposta calcolato può essere risolto in forma esplicita

DAI VINCOLI DI OUTPUT AI VINCOLI DI INPUT

- insiemi di vincoli rappresentano insiemi possibilmente infiniti di soluzioni eventualmente in forma esplicita

- lo stesso meccanismo può essere utilizzato per specificare proprietà soddisfatte dal goal, inserendo insiemi di vincoli nel goal iniziale.

- esempio

```
p(X,Y,Z) ←
      Z = X*X1 + Y*Y1,
      X1 > Y1,
      q(X,Y,X1,Y1).
q(X,Y,X,Y) ←

      ← X>Y, p(X,Y,Z)
            |
      ← X>Y, Z = X*X + Y*Y
```

UN ALTRO ESEMPIO

$$\begin{array}{l} \text{sumto}(0,0). \\ \text{sumto}(N,S) :- \; N \geq 1,\; N \leq S,\; N1 = N-1,\; S1 = S-N \;\blacksquare \\ \qquad\qquad\qquad\qquad \text{sumto}(N1, S1). \end{array}$$

- definisce la relazione sumto $(n, 1+2+\cdots+n)$ sui naturali

- un'interrogazione

  ?- $S \leq 3 \;\blacksquare\; \text{sumto}(N,S)$

- 3 risposte $(N=0, S=0)$, $(N=1, S=1)$, $(N=2, S=3)$

- la derivazione corrispondente alla terza risposta

?- $S \leq 3 \;\blacksquare\; \text{sumto}(N,S)$.

$\downarrow$

?- $S \leq 3,\; N \geq 1,\; N \leq S,\; N1 = N-1,\; S1 = S-N \;\blacksquare\; \text{sumto}(N1,S1)$.

$\downarrow$

?- $S \leq 3,\; N \geq 1,\; N \leq S,\; N1 = N-1,\; S1 = S-N,\; N1 \geq 1,\; N1 \leq S1,$
$N2 = N1-1,\; S2 = S1-N1 \;\blacksquare\; \text{sumto}(N2,S2)$.

$\blacksquare$

?- $S \leq 3,\; N \geq 1,\; N \leq S,\; N1 = N-1,\; S1 = S-N,$
$N1 \geq 1,\; N1 \leq S1,\; N2 = N1-1,\; S2 = S1-N1,$
$N2 = 0,\; S2 = 0$.

$\Downarrow$

$(N=2,\; S=3)$

?- $S \leq 3,\; N \geq 1,\; N \leq S,\; N1 = N-1,\; S1 = S-N,$
$N1 \geq 1,\; N1 \leq S1,\; N2 = N1-1,\; S2 = S1-N1,$
$N2 \geq 1,\; N2 \leq S2,\; N3 = N2-1,\; S3 = S2-N2 \;\blacksquare$
$\text{sumto}(N3, S3)$.

il vincolo è insoddisfacibile e quindi la computazione termina

- vincoli per specificare sia l'input che l'output
- i vincoli vengono creati dinamicamente durante l'esecuzione
- l'insieme dei vincoli raccolti ad ogni passo è testato per verificarne la soddisfacibilità prima di continuare la valutazione

DOMINI DI VINCOLI E LINGUAGGI

- algebre dei termini (equazioni) (alberi finiti, dominio di Herbrand) — PROLOG + quasi tutti gli altri
- equazioni e disequazioni su alberi razionali — PROLOG II
- aritmetica lineare sui reali (i vincoli non lineari sono ritardati) — CLP(R)
- algebre Booleane — CHIP, PROLOG III, BNR-PROLOG
- aritmetica lineare sui razionali — CHIP, PROLOG III
- domini finiti (aritmetica lineare su sottoinsiemi limitati di interi) — CHIP, CLP(FD), BNR-PROLOG
- stringhe — PROLOG III
- aritmetica sui reali (trattata in modo numerico) — BNR-PROLOG

UN ESEMPIO VERO

problema di Dirichlet per l'equazione di Laplace in 2 dimens.

```
laplace([H1, H2, H3|T]) ← av(H1, H2, H3),
                          laplace([H2, H3|T]).
laplace([_,_]) ←
av([TL,T,TR|T1], [ML,M,MR|T2], [BL,B,BR|T3]) ←
        B+T+ML+MR-4*M=0,
        av([T,TR|T1], [M,MR|T2], [B,BR|T3]).
av([_,_], [_,_], [_,_]).
```

- il programma riempie una matrice (lista di liste) che dà le temperature di una superficie in punti discreti
- l'input è una approssimazione della matrice che contiene i valori dei quattro bordi
- ogni temperatura è calcolata come la media dei 4 vicini

```
← laplace([
     [0, 0, 0, 0, 0],
     [100, R, S, T, 100],
     [100, U, V, W, 100],
     [100, X, Y, Z, 100],
     [100, 100, 100, 100, 100], ]).

  ⟹   ← R=57.143,
          S=47.321,
          T=57,143
          ⋮
```

- la risoluzione in stile prolog-logica accumula vincoli che sono le varie istanze dell'equazione in av
- il risolutore di vincoli calcola "poi" le soluzioni

ALCUNI DOMINI DI VINCOLI

1. aritmetica sui reali

$\Sigma = \{ \{0,1\}, \{+,*\}, \{=,<,\leq\} \}$

$\mathcal{A}$ = dominio dei reali, con l'interpretazione naturale ai funzioni e relazioni

$(\mathcal{A}, \mathcal{L})$ aritmetica sui reali $\mathfrak{R}$

- i vincoli sono equazioni e disequazioni del tipo

$$X*X + X+X + X*Y \geq Y+Y+Y$$

(coefficienti solo 0 o 1)

$$X*X + 2X + X*Y \geq 3Y$$

- se la segnatura non contiene $*$

$(\mathcal{A}', \mathcal{L}')$ aritmetica lineare sui reali $\mathfrak{R}_{lin}$

- se $\mathcal{A}$ è ristretto ai razionali

$\mathfrak{Q}_{lin}$

- se aggiungiamo in Σ la relazione $\neq$

$\mathfrak{R}_{lin}^{\neq}$ e $\mathfrak{Q}_{lin}^{\neq}$

- se restringiamo Σ a $\{\{0,1\}, \{+\}, \{=\}\}$

$\mathfrak{R}_{linEqn}$ equazioni lineari sui reali

ALCUNI DOMINI DI VINCOLI

2. Il dominio dei vincoli di Herbrand

Σ costanti e funzioni del programma + =

$\mathcal{A}$ ~~int~~ dominio dei termini finiti (alberi finiti) con la pre-interpretazione di Herbrand

$(\mathcal{A}, \mathcal{L})$ dominio dei vincoli di Herbrand FT (termini finiti) (programmazione logica pura)

- i vincoli sono equazioni fra termini

$$x = g(y)$$

$$\exists z \;\; x = f(z, z) \wedge y = g(z)$$

(si può omettere il quantificatore, perché tutte le variabili che occorrono solo nei vincoli sono implicitamente quantificate esistenzialmente)

3. Il dominio dei vincoli degli alberi razionali

- stesso Σ di FT
- $\mathcal{A}$ dominio degli alberi razionali (soluzioni di equazioni senza occur check)

~~RT~~ RT

ALCUNI DOMINI DI VINCOLI

6. Il dominio dei vincoli dei Booleani

$\Sigma = \{\{0,1\}, \{\neg, \wedge, \vee, \oplus, \Rightarrow\}, \{=\}\}$

$\mathcal{A}$ = dominio {true, false}
con l'interpretazione usuale per gli operatori logici
(not, and, or, or disgiuntivo, implicazione)

BOOL

esempio di vincoli

$\neg(X \wedge Y) = Y$

$\neg(X \wedge Y) \oplus Y$ rappresenta $\neg(X \wedge Y) \oplus Y = 1$

7. il dominio finito di vincoli

$\Sigma = \{\phi; \{+\}, \{\{\in [m,n]\}_{m \le n}, =, \neq, \le\}\}$

intervalli sugli interi

$\mathcal{A}$ = dominio degli interi, con tutti i simboli di Σ interpretati nel modo naturale

$\mathcal{L}$ è l'insieme dei vincoli generati da Σ sotto la condizione che ogni variabile è soggetta ad un vincolo di intervallo

FD domini finiti

esempio di vincolo

$X \in [1,5] \wedge Y \in [0,7] \wedge X \neq 3 \wedge X + 2Y \le 5 \wedge X + Y \le 9$

IL RISOLUTORE DI VINCOLI

- FT (Herbrand)
  - è noto un algoritmo lineare (unificazione)
  - in Prolog viene fatto con l'unificazione su RT (alberi razionali), che, pur se quasi lineare, permette di fatto di economizzare nell'ispezione dei termini

- R_{LinEqn}
  - l'eliminazione Gaussiana è quadratica

- R_{Lin}
  - esistono algoritmi polinomiali, ma non si usano
  - si usa il Simplesso pur essendo esponenziale
    - deve essere esteso perché quello originale lavora con numeri nonnegativi e relazioni di ugua.

- FD
  - il problema è quasi sempre NP-hard

- BOOL
  - il problema è NP completo

- FEAT (feature trees)
  - analogo ad RT

APPLICAZIONI DI PROGRAMMAZIONE LOGICA CON VINCOLI

- modelli di sistemi complessi
  - i vincoli permettono una interpretazione dichiarativa delle relazioni di base
  - le regole (clausole) permettono di definire relazioni più complesse
  - CLP è essenzialmente usato come linguaggio di specifica
  - simulazione del sistema

- problemi di ricerca combinatoria
  - le regole forniscono il meccanismo per enumerare
  - i vincoli permettono di ridurre lo spazio di ricerca