NGÔN NGỮ DLP^A VÀ MỘT SỐ ỨNG DỤNG

Võ Thị Như Lý*
Trường Cao đẳng Công thương miền Trung
Ngày nhận bài: 03/4/2020; Ngày nhận đăng: 08/01/2021

Tóm tắt

Trong bài báo này, chúng tôi xem xét cú pháp và ngữ nghĩa của ngôn ngữ lập trình logic DLP^A cùng các tính chất ngữ nghĩa của ngôn ngữ DLP^A, được gọi đơn giản là chương trình logic dạng tuyển kết tập. Chúng tôi cũng trình bày một số ứng dụng điển hình sử dụng ngôn ngữ DLP^A chạy trên hệ thống lập trình DLV.

Từ khóa: *Lập trình logic, ngôn ngữ DLP^A, hệ thống DLV*

1. Giới thiệu

Trong suốt những thập kỷ qua, một mô hình lập trình mới là "Lập trình logic" đã ra đời. Lập trình logic (LP) chủ yếu dựa trên ý tưởng lập trình khai báo, ở đó các chương trình không được tạo ra từ các câu lệnh cũng như từ các hàm mà được tạo ra chủ yếu dựa trên tập các vị từ. Lĩnh vực nghiên cứu LP được nhiều nhà khoa học quan tâm và đã được áp dụng vào việc biểu diễn và xử lý tri thức phức tạp trong lĩnh vực trí tuệ nhân tạo và các lĩnh vực nổi lên khác như quản trị tri thức và tích hợp thông tin. Hiện nay LP được mở rộng theo nhiều hướng khác nhau, trong đó ngôn ngữ DLP^A – một sự mở rộng của LP, cho phép các hàm kết tập xuất hiện trong các quy tắc của chương trình logic.

Hàm kết tập có ý nghĩa rất đáng kể, nó cho phép mô hình hóa một cách tự nhiên và ngắn gọn về nhiều vấn đề. Nó làm tăng khả năng diễn đạt các thuộc tính, thường nảy sinh trong các ứng dụng thế giới thực, được mã hóa theo cách đơn giản và tự nhiên. Trong số đó, có những thuộc tính yêu cầu áp dụng các toán tử toán học (như sum, times, count...) trong một tập các yếu tố thỏa mãn một vài điều kiện, được diễn đạt một cách dễ dàng và tự nhiên bằng ngôn ngữ DLP^A. Ngôn ngữ DLP^A được áp dụng để giải quyết được nhiều bài toán thực tế phức tạp, chẳng hạn bài toán sắp xếp chỗ ngồi, các bài toán tối ưu hóa của lý thuyết đồ thị và nhiều dạng suy luận phỏng đoán.

2. Ngôn ngữ DLP^A

2.1. Cú pháp của ngôn ngữ DLP^A

2.1.1. Tập DLP^A (Armi et.,2003).

Một tập DLP^A là một *tập ký hiệu* hoặc là một *tập nền*. Trong đó:

- *Tập ký hiệu* có dạng {*Vars* : *Conj*}, trong đó *Vars* là một danh sách các biến và *Conj* là hội của các literal thông thường (có thể chứa các literal âm).

- *Tập nền* là tập các cặp có dạng $\langle \bar{t} : Conj \rangle$, trong đó $\bar{t}$ là danh sách các hằng và *Conj* là hội của các literal thông thường nền.

2.1.2. Hàm kết tập/nguyên tố kết tập

(Armi et., 2003).

a) Một *hàm kết tập* có dạng f(S), trong đó S là một tập DLP^A và f là một trong số các hàm #count, #min, #max, #sum, #times.

b) Một *nguyên tố kết tập* có dạng $Lg\ \mathrm{p}_1\ f(S)\ \mathrm{p}_2\ Rg$, trong đó *f*(*S*) là một

* Email: lyvo2019@gmail.com

hàm kết tập, $p_1, p_2 \in \{=, <, \leq, >, \geq\}$, *Lg* và *Rg* là các hạng thức (biến hoặc hằng) và gọi là *chặn*. "$Lg\ p_1$" hoặc "$p_2\ Rg$" có thể không có.

2.1.3. Nguyên tố, literal (Armi et.,2003).

a) Một *nguyên tố* là một nguyên tố thông thường hoặc một nguyên tố kết tập.

b) Một *literal* L là một nguyên tố A hoặc phủ định của nguyên tố A. Nếu A là nguyên tố kết tập thì L là literal kết tập.

2.1.4. Quy tắc DLP^A (Faber et., 2011).

Một *quy tắc* DLP^A *r* là một cấu trúc có dạng:

$a_1 \vee a_2 \vee \ldots \vee a_n :- b_1, b_2, \ldots, b_k, not\ b_{k+1}, \ldots, b_m$

trong đó $a_1, a_2, \ldots, a_n$ là các nguyên tố thông thường, $b_1, b_2, \ldots, b_m$ là những nguyên tố, $n \geq 0$, $m \geq k \geq 0$, $m + n \geq 1$, tuyển $a_1 \vee a_2 \vee \ldots \vee a_n$ là đầu của r và hội $b_1, b_2, \ldots, b_k, not\ b_{k+1}, \ldots, b_m$ là thân của *r*. Ký hiệu $H(r) = \{a_1, ..., a_n\}$ và $B(r) = \{b_1, b_2, ..., b_k, not\ b_{k+1}, ..., not\ b_m\}$. Một quy tắc với phần đầu rỗng (nghĩa là $n = 0$) thì được gọi là ràng buộc toàn vẹn (hay ràng buộc mạnh). Một quy tắc với thân rỗng (nghĩa là $k = m = 0$) được gọi là một sự kiện và ta thường bỏ qua ký hiệu ":−".

2.1.5. Ràng buộc yếu (Faber et., 2008).

Một *ràng buộc yếu wc* cú pháp có dạng:

$$:\sim L_1 \wedge \ldots \wedge L_k\ [w : l]$$

trong đó $k \geq 1$ và các L_i ($i = 1,..., n$) là các literal, còn $weight(wc) = w$ và $layer(wc) = l$ là các hằng hoặc biến nguyên dương và *w*, *l* có thể được bỏ qua và mặc định là bằng 1.

Ràng buộc yếu cho phép ta biểu diễn một số các bài toán tối ưu theo một cách tự nhiên và đơn giản. Trong khi các ràng buộc chuẩn (ràng buộc toàn vẹn, ràng buộc mạnh) luôn phải được thỏa mãn, ràng buộc yếu biểu diễn một mức độ mong muốn nào đó, tức là chúng có thể được thỏa mãn khi có thể nhưng chúng không loại bỏ bất kỳ mô hình nào.

Xét chương trình có các ràng buộc yếu như sau:

$$a \vee b.\quad c :- b.\quad :\sim a.\quad :\sim b.\quad :\sim c.$$

Ở đây, mức độ đánh giá và độ ưu tiên được bỏ qua, giá trị được gán ngầm định là 1. Nếu ta gọi trong DLV, ta thu được kết quả sau:

Best Model: {*a*}

Chú ý rằng các tập trả lời của chương trình {$a \vee b, c:-b$ } là {*a*} và {*b*, *c*}. Sự xuất hiện của các ràng buộc yếu đã loại bỏ {*b*, *c*} vì nó mâu thuẫn với các ràng buộc yếu (trong khi đó {*a*} chỉ mâu thuẫn một ràng buộc yếu).

2.1.6. Chương trình DLP^A (Faber et., 2008).

Một chương trình DLP^A là một tập các quy tắc DLP^A và các ràng buộc yếu (nếu có).

Để đơn giản và không mất tính tổng quát, ta giả sử rằng thân của mỗi quy tắc gồm nhiều nhất là một nguyên tố kết tập. *Biến toàn cục* của *r* là biến xuất hiện trong nguyên tố thông thường của *r*, *biến cục bộ* là biến chỉ xuất hiện trong hàm kết tập của *r*.

2.2. Ngữ nghĩa của ngôn ngữ DLP^A

2.2.1. Các khái niệm

a) Vũ trụ và cơ sở của chương trình DLP^A P

Cho *P* là chương trình DLP^A, *vũ trụ* của P, ký hiệu U_p chỉ tập các hằng xuất hiện trong *P*, *cơ sở* của P, ký hiệu B_P là tập các nguyên tố thông thường xây dựng từ các vị từ của *P* với hằng trong U_P.

b) Phép thế và hiện hành của chương trình DLP^A P (James, P. D., & Faber, W. (2011)).

Phép thế là một ánh xạ từ tập các biến đến tập U_P các hằng trong *P*. Phép thế từ tập các biến toàn cục của quy tắc *r* đến tập U_P là *phép thế toàn cục* đối với quy tắc *r*. Phép thế từ tập các biến cục bộ của tập

ký hiệu *S* đến tập U_P là *phép thế cục bộ* đối với *S*.

Cho tập ký hiệu *S* không chứa biến toàn cục, *S* = {*Vars*: *Conj*}, *hiện hành* của *S*, ký hiệu *inst*(*S*), là tập nền:

inst(*S*) = {⟨γ(*Vars*) : γ(*Conj*) ⟩ | γ là phép thế cục bộ đối với *S*}.

Một *hiện hành nền* của quy tắc *r* nhận được qua 2 bước:

(1) Phép thế toàn cục σ đối với *r* được áp dụng đầu tiên trên *r*.

(2) Mỗi tập ký hiệu *S* trong σ (*r*) được thay thế bởi hiện hành *inst*(*S*).

Hiện hành của chương trình DLP^A *P*, ký hiệu *Ground*(*P*), là tập tất cả các hiện hành có thể có của các quy tắc của *P*.

c) Thể hiện và phép định giá (Faber et.,2004).

Một *thể hiện* của chương trình DLP^A *P* là một tập các nguyên tố thông thường nền $I \subseteq B_P$.

Việc xác định giá trị chân lý của *A* theo thể hiện *I*, trong đó *A* là literal thông thường nền hoặc hội các literal thông thường nền, ký hiệu là *I*(*A*) được định nghĩa theo cách thông thường.

Bên cạnh việc gán giá trị chân lý cho các literal thông thường nền thì một thể hiện còn cung cấp ngữ nghĩa cho các tập nền, các hàm kết tập và các literal kết tập. Ngữ nghĩa của một tập nền, một hàm kết tập, một nguyên tố kết tập theo một thể hiện, tương ứng là một đa tập, một giá trị và một giá trị chân lý.

Phép định giá *I*(*S*) của tập *S* theo thể hiện *I* là đa tập của hằng đầu tiên của các phần tử trong *S* mà hội của chúng là đúng theo *I*. Chính xác hơn, đặt S_I = {

$\langle t_1,\ldots,t_n\rangle \mid \langle t_1,\ldots,t_n : Conj\rangle \in S \wedge Conj$ là đúng theo I}. *Phép định giá I*(*S*) của *S* theo thể hiện *I* là đa tập

$[\, t_1 \mid \langle t_1,\ldots, t_n\rangle \in S_I \,]$

Phép định giá *I*(*f*(*S*)) của hàm kết tập *f*(*S*) theo *I* là kết quả của việc áp dụng hàm *f* trên *I*(*S*). Nếu đa tập *I*(*S*) không nằm trong miền của *f* thì *I*(*f*(*S*)) = $\perp$.

Một nguyên tố kết tập *A* = *Lg*p_1*f*(*S*) p_2 *Rg* là đúng theo thể hiện *I* nếu:

(i) *I*(*f*(*S*)) ≠ $\perp$, và

(ii) Quan hệ $Lg\ \mathrm{p}_1\ I(f(S))$ và $I(f(S))\ \mathrm{p}_2\ Rg$ đều thỏa mãn;

còn ngược lại thì *A* sai.

Một literal kết tập hiện hành *not A*= not f(S)$\mathrm{p}\,k$ đúng theo I nếu:

(i) *I*(*f*(*S*)) ≠ $\perp$, và

(ii) I(f(S))$\mathrm{p}\,k$ đều thoả mãn;

còn ngược lại thì *A* sai.

Một qui tắc r đúng theo *I* (ký hiệu là: $I \vDash r$) nếu một vài nguyên tố trong phần đầu của quy tắc đúng theo *I* ($\exists h \in H(r): I \vDash h$) và tất cả literal trong thân quy tắc đúng theo *I* ($\forall b \in B(r): I \vDash b$).

(Mô hình của chương trình DLP^A) Một mô hình của chương trình DLP^A *P* là một thể hiện *M* của *P* sao cho mọi quy tắc $r \in$ Ground(*P*) là đúng theo *M*. Mô hình *M* của *P* là mô hình cực tiểu nếu không tồn tại mô hình *N* của *P* sao cho *N* là tập con thực sự của *M*.

2.2.2. Tập trả lời

a) Phép biến đổi Gelfond Lifschitz (Faber et.,2008).

Phép biến đổi Gelfond Lifschitz của chương trình DLP^A nền *P* theo tập $X \subseteq B_P$ là một chương trình DLP^A nền dương P^X thu được từ *P* bằng cách:

- Xoá tất cả quy tắc $r \in P$ mà literal phủ định trong *B*(*r*) là sai theo *X* hoặc literal kết tập là sai theo *X*,
- Xoá các literal kết tập, literal phủ định từ những quy tắc còn lại.

b) Tập trả lời của chương trình DLP^A P là tập $X \subseteq B_P$ sao cho X là tập trả lời của $Ground(P)^X$.

Cho chương trình DLP^A P sau đây:

$d(1):-$

$a \vee b:- c$

$b:-$ *not* a , *not* c , *#count*$\{Y: d(Y)\}>0$

$a \vee c: -$ *not* b, *#sum*$\{Y: d(Y)\}>1$

Xét thể hiện $I = \{b, d(1)\}$, lúc đó P^I như sau:

$d(1):-$

$a \vee b:- c$

$b:-$

Ta nhận thấy I là tập trả lời của P^I, nên nó là tập trả lời của P.

Xem thể hiện $J = \{a, d(1)\}$, lúc đó P^J như sau:

$d(1):-$

$a \vee b:-c$

Ta nhận thấy J là tập trả lời của P^J, vì vậy nó cũng là tập trả lời của P .

Xét tập $K = \{c, d(1)\}$,ta có $P^K = P^J$ nhưng K không phải là tập trả lời của P^K , vì đối với quy tắc $r: a \vee b:-c$, thì $B(r) \subseteq K$ nhưng $H(r) \cap K \neq \varnothing$ không thoả. Thực vậy, có thể chỉ ra rằng chỉ có I và J là các tập trả trả lời của P.

3. Một số ứng dụng

3.1. Bài toán xây dựng đội làm việc cho một dự án

Một đội làm việc cho một dự án được xây dựng từ tập các nhân viên theo các yêu cầu sau đây:

P_1. Đội phải có một số lượng nhân viên cụ thể.

P_2. Trong đội phải có ít nhất các kỹ năng khác nhau theo yêu cầu

P_3. Tổng lương của các nhân viên làm việc trong đội không vượt quá ngân sách đề ra.

P_4. Lương của mỗi nhân viên nằm trong giới hạn cho phép.

P_5. Số lao động nữ trong đội phải đạt mức tối thiểu đã đưa ra

Giả sử rằng các nhân viên được cung cấp một số sự kiện có dạng *emp*(*EmpId, Sex, Skill, Salary*); Qui mô của đội, số lượng tối thiểu các kỹ năng khác nhau, ngân sách, mức lương tối đa và số lượng tối thiểu nhân viên nữ lần lượt được qui định bởi các nguyên tố *nEmp*(*N*), *nSkill*(*N*), *budget*(*B*), *maxSal*(*M*) và *women*(*W*). Từ những thông tin này, thực hiện mã hoá các yêu cầu bằng ngôn ngữ DLP^A ta nhận được chương trình sau:

(r_1)*in*(*I*) $\vee$ *out*(*I*) :- *emp*(*I, Sx, Sk, Sa*).

(r_2):- *nEmp*(*N*),*not #count*{*I* : *in*(*I*)} = *N*.

(r_3):- *nSkill*(*M*), *not #count*{*Sk* : *emp*(*I, Sx, Sk, Sa*), *in*(*I*)} $\geq$ *M*.

(r_4) :- *budget*(*B*), *not #sum*{*Sa, I* : *emp*(*I, Sx, Sk, Sa*), *in*(*I*)} $\leq$ *B*.

(r_5):- *maxSal*(*M*), *not #max*{*Sa* : *emp*(*I, Sx, Sk, Sa*), *in*(*I*)} $\leq$ *M*.

(r_6):- *women*(*W*), *not #count*{*I* : *emp*(*I, f, Sk, Sa*), *in*(*I*)} $\geq$ *W*.

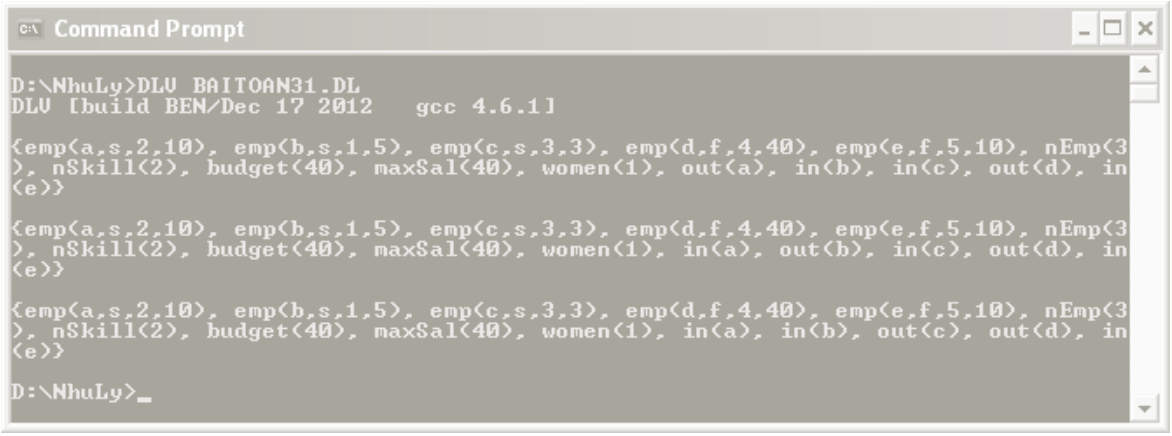
Quy tắc dạng tuyển (r_1) "dự đoán" một nhân viên có trong đội hay không, trong khi 5 ràng buộc (r_2-r_6) tương ứng với 5 yêu cầu P_1 - P_5 . Nhờ vào các hàm kết tập nên việc chuyển đổi các yêu cầu dễ dàng hơn. Đây là một ví dụ làm nổi bật tính hữu ích của biểu diễn tập hợp và đa tập. Việc mã hoá của yêu cầu P_2 đòi hỏi một tập bởi vì ta muốn đếm số kỹ năng khác nhau; hai nhân viên trong đội có kỹ năng giống nhau sẽ được đếm một lần. Ngược lại, P_3 yêu cầu tính tổng các phần tử của đa tập; nếu 2 nhân viên có lương giống nhau thì cả hai giá trị lương phải được cộng tổng hợp cho P_3. Điều này đạt được bằng cách thêm biến I (định danh của mỗi nhân viên) trong *Vars*. Việc định giá của {*Sa, I* : *emp*((*I, Sx, Sk, Sa*), *in*(*I*)} sinh ra tập $S = \{\langle Sa, I\rangle : Sa$ *là lương của nhân viên I trong đội*$\}$. Hàm

tổng *#sum* được áp dụng trên đa tập của thành phần *Sa* đầu tiên của bộ $\langle Sa, I \rangle$ trong *S*.

Giả sử các sự kiện đầu vào là:

emp(*a, s,2,10*). *emp*(*b,s,1,5*).
emp(*c,s,3,3*). *emp*(*d,f,4,40*).
emp(*e,f,5,10*).
nEmp(3). *nSkill*(2). *budget*(40).
maxSal(40). *women*(1).

Thực thi chương trình này bằng hệ thống DLV ta nhận được kết quả như hình sau:



***Hình 1**: Kết quả thực thi chương trình bằng hệ thống DLV*

Chương trình này có 3 tập trả lời sau:

M_1 = {emp(a,s,2,10), emp(b,s,1,5), emp(c,s,3,3), emp(d,f,4,40), emp(e,f,5,10), nEmp(3), nSkill(2), budget(40), maxSal(40), women(1), out(a), in(b), in(c), out(d), in(e)}

M_2 = {emp(a,s,2,10), emp(b,s,1,5), emp(c,s,3,3), emp(d,f,4,40), emp(e,f,5,10), nEmp(3), nSkill(2), budget(40), maxSal(40), women(1), in(a), out(b), in(c), out(d), in(e)}

M_3 = {emp(a,s,2,10), emp(b,s,1,5), emp(c,s,3,3), emp(d,f,4,40), emp(e,f,5,10), nEmp(3), nSkill(2), budget(40), maxSal(40), women(1), in(a), in(b), out(c), out(d), in(e)}

3.2. Bài toán tìm cây khung nhỏ nhất đồ thị

Cho G = <V, E> là một đồ thị có hướng và có trọng số. Tìm cây khung của G và có trọng số nhỏ nhất.

a) Chương trình viết bằng ngôn ngữ DLP^A và không sử dụng hàm kết tập:

```
root(a).
node(a). node(b). node(c). node(d). node(e).
edge(a,b,4). edge(a,c,3). edge(c,b,2). edge(c,d,3).
edge(b,e,4). edge(d,e,5).
in_tree(X,Y,C) v out_tree(X,Y) :-
edge(X,Y,C), reached(X).
:- root(X), in_tree(_,X,C).
:- in_tree(X,Y,C), in_tree(Z,Y,C), X != Z.
reached(X):- root(X).
reached(Y):- reached(X), in_tree(X,Y,C).
:- node(X), not reached(X).
:~ in_tree(X,Y,C). [C:1]
```

Thực thi chương trình này bằng hệ thống DLV ta nhận được kết quả như ở hình sau:

Hình 2. *Kết quả thực thi chương trình của Bài toán đồ thị*

Tập trả lời tìm được là:

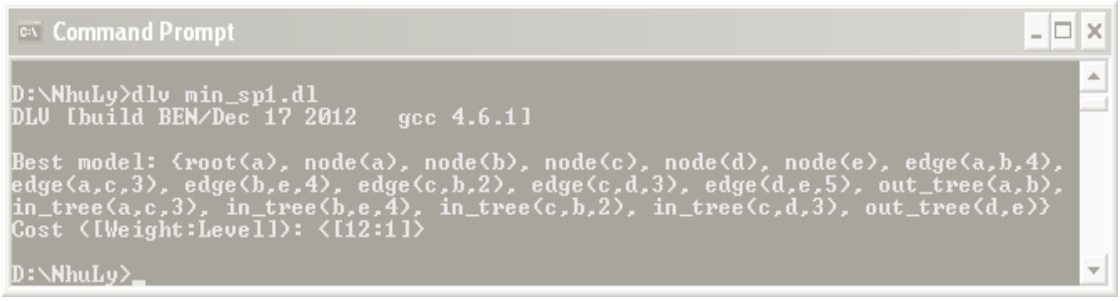
M_1 = {root(a), node(a), node(b), node(c), node(d), node(e), edge(a,b,4), edge(a,c,3), edge(b,e,4), edge(c,b,2), edge(c,d,3), edge(d,e,5), reached(a), out_tree(a,b), in_tree(a,c,3), reached(b), reached(c), in_tree(b,e,4), in_tree(c,b,2), in_tree(c,d,3), reached(e), reached(d), out_tree(d,e)}

Cost ([Weight:Level]): <[12:1]>

b) Chương trình viết bằng ngôn ngữ DLP^A và có sử dụng hàm kết tập:

```
root(a).
node(a). node(b). node(c). node(d). node(e).
edge(a,b,4). edge(a,c,3). edge(c,b,2).
edge(c,d,3). edge(b,e,4). edge(d,e,5).
in_tree(X,Y,C) v out_tree(X,Y) :-
edge(X,Y,C).
%nút gốc của cây không có cung đến
:- root(R), not #count{X : in_tree(X,R,C)}
= 0.
%mỗi nút trong cây chỉ có một cung đến
 :- edge(_,Y,_), not #count{X :
in_tree(X,Y,_)} = 1.
:~ in_tree(X,Y,C). [C:1]
```

Thực thi chương trình này bằng hệ thống DLV ta nhận được kết quả như ở hình sau:



Hình 3. *Kết quả thực thi chương trình của Bài toán đồ thị (dùng hàm kết tập)*

Tập trả lời tìm được là:

M_1 = {root(a), node(a), node(b), node(c), node(d), node€, edge(a,b,4), edge(a,c,3), edge(b,e,4), edge(c,b,2), edge(c,d,3), edge(d,e,5), reached(a), out_tree(a,b), in_tree(a,c,3), reached(b), reached(c), in_tree(b,e,4), in_tree(c,b,2), in_tree(c,d,3), reached€, reached(d), out_tree(d,e)}

Cost ([Weight:Level]): <[12:1]>

Ta nhận thấy kết quả chương trình khi dùng hàm kết tập và khi không dùng hàm kết tập đều cho ra tập trả lời như nhau. Tuy nhiên, khi dùng hàm kết tập thì số lượng quy tắc

phải dùng sẽ được rút ngắn, mã hóa một cách trực tiếp, tự nhiên và sử dụng được các quan hệ kế thừa nên sẽ biểu diễn tri thức tốt hơn. Và các nghiên cứu chỉ ra rằng khi thêm vào hàm kết tập thì không làm tăng độ phức tạp của chương trình logic dạng tuyển (Faber et.,2004).

	∅	$\{M_s\}$	$\{M\}$	$\{A_s\}$	$\{N_s\}$	$\{M_s,A_s, N_s\}$	$\{A\}$	$\{N\}$	$\{M,A,N\}$
Không có phủ định (not)	co-NP	co-NP	co-NP	$\prod_2^P$	$\prod_2^P$	$\prod_2^P$	$\prod_2^P$	$\prod_2^P$	$\prod_2^P$
Phủ định (not)	$\prod_2^P$	$\prod_2^P$	$\prod_2^P$	$\prod_2^P$	$\prod_2^P$	$\prod_2^P$	$\prod_2^P$	$\prod_2^P$	$\prod_2^P$

Bảng 1. *Sự phức tạp của chương trình logic dạng tuyển với hàm kết tập*

Trong đó:

M_s: Kết tập đơn điệu phân tầng

M: Kết tập đơn điệu hoàn toàn (có thể có đệ qui)

A_s: Kết tập không đơn điệu phân tầng

A: Kết tập kháng đơn điệu hoàn toàn

N_s: Kết tập không đơn điệu phân tầng

N: Kết tập không đơn điệu hoàn toàn

Tất cả các dạng kết tập đều có kết quả là $\prod_2^P$ cũng chính bằng chương trình logic dạng tuyển chuẩn.

4. Kết luận

- Bài báo đã biểu diễn tri thức bằng chương trình DLP^A thông qua một số bài toán thực tế và cài đặt các bài toán này trên hệ thống DLV.

- Việc mở rộng chương trình logic dạng tuyển với các hàm kết tập góp phần tạo thêm điểm mạnh trong việc mô hình hóa một cách tự nhiên các tri thức không đầy đủ và làm nổi bật những đặc tính của các ngôn ngữ biểu diễn tri thức mới. Ngoài ra, còn góp phần vào nghiên cứu phương pháp định giá truy vấn đối với chương trình DLP^A❑

TÀI LIỆU THAM KHẢO

Armi, T. D., Faber, W., Ielpa, G., Leone, N., & Pfeifer, G. (2003). *Aggregate Functions in DLV*. Messina, Italy: ASP'03, 274 – 288.

Armi, T. D., Faber, W., Ielpa, G., Leone, N., & Pfeifer, G. (2003). *Aggregate Functions in Disjunctive Logic Programming: Semantics, Complexity, and Implementation in DLV*. Acapulco, Mexico: IJCAI 2003, 847-852.

Faber, W., Leone, N., & Pfeifer, G. (2004). Recursive Aggregates in Disjunctive Logic Programs: Semantics and Complexity. *J.J. Alferes, J. Leite (Eds.), Proceedings of the 9th European Conference on Artificial Intelligence (JELIA 2004), Lecture Notes in AI (LNAI), vol. 3229,* Springer-Verlag, 200–212.

Faber, W., Leone, N., & Pfeifer, G. (2011). *Semantics and complexity of recursive aggregates in answer set programming.* Contents lists available at Science Direct Artificial Intelligence, 278–298.

Faber, W., Pfeifer, G., Leone, N., Armi, T. D., & Ielpa, G. (February 2008). *Design and Implementation of Aggregate Functions in the DLV System*. Department of Mathematics, University of Calabria 87036 Rende (CS), Italy, 12-20.

Faber, W., Eiter, T., Georg, G., Leone, N., Pfeifer, G., Perri, S., & Francesco, S. (2006). *The DLV System for Knowledge Representation and Reasoning*. ACM Transactions on Computational Logic, 499-562.

James, P. D., & Faber, W. (2011). *Logic Programming and Nonmonotonic Reasoning*. Vancouver, Canada: 11th International Conference.

Logic Programming Language DLP^A and its applications

Vo Thi Nhu Ly
Mientrung Industry And Trade College
Email: lyvo2019@gmail.com
Received: April 03, 2020; Accepted: January 08, 2021

Abstract

In this paper, we summarize some syntax, semantics and semantic properties of the Logic Programming Language DLP^A which is also simply called Disjunctive logic program with Aggregate Functions. We also demonstrate some applications in using the Logic Programming Language DLP^A implemented in the Datalog plus Vel (DLV) system.

Keywords: *Logic Programming, DLP^A Language, DLV system.*