제 19 회 한국정보처리학회 춘계학술발표대회 논문집 제 10 권 제 1 호 (2003. 5)

Prolog 언어를 사용한 집합 일치화의 구현

김인영*, 신동하**
*상명대학교 일반대학원 컴퓨터과학과
**상명대학교 소프트웨어학부
e-mail : esirizad@smu.ac.kr

An Implementation of Set Unification Using Prolog

Inyoung Kim*, Dongha Shin**
* Dept. of Computer Science, Sangmyung University
** Division of Computer Software, Sangmyung University

요 약

본 논문은 "집합 일치화 문제(set unification problem)"를 논리 언어 Prolog를 사용하여 구현한다. 집합 일치화 문제는 고전적 논리 언어(logic languages)의 일치화 문제(unification problem)에서 집합을 표현할 수 있도록 확장한 것으로 최근 연구되고 있는 "집합 제한 논리 언어(set constraints logic languages)"를 구현하기 위하여 풀어야 하는 문제이다. 본 논문에서는 최근 A. Dovier 연구팀이 제안한 집합 일치화 문제의 풀이(solver)를 소개하고, 이 풀이를 논리 언어 Prolog를 사용하여 구현하는 방법을 기술한다. Prolog 언어는 비결정성(nondeterminism)을 표현할 수 있는 기능과 리스트(list)라는 자료 구조를 제공하는 기능 때문에 다른 어떤 언어에서보다 쉽게 집합 일치화 문제 풀이를 구현할 수 있다. 본 연구의 결과는 집합 제한 논리 언어의 수행기(interpreter) 개발에 직접 이용될 수 있다.

1. 서 론

본 논문은 "집합 일치화 문제(set unification problem)"[1]를 논리 언어[3] Prolog[2]를 사용하여 구현하는 방법을 기술한다. 집합 일치화 문제는 고전적 논리 언어(logic languages)[3]에서 다루는 일치화 문제(unification problem)[4]를 집합의 표현까지 확장한 것으로 최근 연구되고 있는 "집합 제한 논리 언어(set constraints logic languages)"[5]를 구현하기 위하여 풀어야 하는 문제이다. 본 논문에서는 최근 A. Dovier 연구팀이 연구한 집합 일치화 문제의 풀이(solver)[1]를 소개하고, 이 풀이를 논리 언어 Prolog를 사용하여 구현하는 방법을 기술한다. Prolog 언어[2]는 비결정성(nondeterminism)을 표현할 수 있고 리스트(list)라는 자료구조가 제공되기 때문에 다른 어떤 언어보다 쉽게 집합 일치화 문제 풀이를 구현할 수 있다.

본 논문의 2절에서는 집합 일치화 문제를 표현하기 위한 언어(language)의 구문(syntax) 및 의미(semantics)를 정의한다. 본 논문에서 다루는 집합 일치화 문제는 "Ab Cl 일치화" 문제로 알려져 있는 데 이 문제에서 사용되는 언어는 집합을 나타내는 함수 심볼 "{· | · }"과 공집합을 표현하는 심볼 "∅"를 사용하고 집합 등식(equation)의 의미를 정의하기 위하여 두 개의 공리(axiom)인 Ab(Absorption)와 Cl(Commutative on left)을 사용한다. 본 논문의 3절에서는 2절에서 정의한 집합 일치화 문제의 풀이(solver) 알고리즘에 대하여 기술한다. 이 알고리즘은 A. Dovier 연구팀이 제안하고 완전성(completeness)을 증명한 알고리즘으로 비결정적(nondeterministic)으로 기술되어 있다. 집합 일치화 알고리즘은 2절에서 기술하는 공리 Ab 및 Cl 에 바탕을 두고 있다. 근원적으로 "Ab Cl 일치화" 문제는 NP-complete 문제이다. 4절에서는 3절에서 기술된 일치화 풀이를 논리 언어 Prolog를 사용하여 구현하는 방법을 기술한다. 논리 언어는 3절에서 기술한 알고리즘과 같이 비결정적으로 기술된 알고리즘을 쉽게 구현할 수 있다. 5절에서는 본 연구에서 구현한 집합 일치화를 바탕으로 구현 예정인 "집합 제한 논리 언어(set constraints logic languages)"에 대하여 간단하게 기술하고 앞으로 연구 계획을 기술한다.

2. 집합 언어

"Ab Cl 일치화" 문제에서 다루는 언어는 집합을 나타내는 함수(function) 심볼 "{· |· }"과 공집합을 표현하는 심볼 "∅"를 사용하고 이 외에도 사용자가 정의

하는 함수 심볼을 사용할 수 있다. 또한 텀(term)의 동등(equality)을 표현하기 위해서는 술어(predicate) 심볼 "="을 사용한다. 그리고 다른 논리 언어에서와 마찬가지로 변수는 보통 영문의 대문자로 표현한다. 집합 $\{X \mid Y\}$의 의미는 $\{X\} \cup Y$를 의미하고 집합 $\{t_1 \mid \{t_2 \mid \ldots \{t_n \mid t\} \ldots\}$는 t가 $\varnothing$이면 $\{t_1, t_2, \ldots, t_n\}$으로 표현한다. 예를 들어 $\{1, 2\} = \{X, Y\}$는 하나의 집합 등식이다.

앞에서 정의한 언어를 사용하여 표현된 집합 등식은 아래 두 개의 공리(axiom)인 Ab(Absorption)와 Cl(Commutative on left)을 만족한다. 즉 아래 두 공리는 집합 일치화의 의미(semantics)를 표현한다.

공리 Ab: $\{X \mid \{X \mid Z\}\} = \{X \mid Z\}$
공리 Cl: $\{X \mid \{Y \mid Z\}\} = \{Y \mid \{X \mid Z\}\}$

3. 집합 일치화 알고리즘

본 절에서는 A. Dovier 연구팀이 제안하고 완전성을 증명한 집합 일치화 알고리즘을 기술한다. 본 연구는 이 알고리즘을 바탕으로 구현되었다. 집합 일치화 알고리즘은 변수가 없는 경우 $O(n^2)$의 시간 복잡성(time complexity)을 가지며 변수가 사용되는 경우 NP-complete임이 증명되어 있다.

알고리즘은 크게 두 부분으로 구성되어 있다. 첫째 부분은 주어진 집합 등식들 중 하나를 선택하는 부분이고 둘째 부분은 앞에서 선택된 등식을 다시쓰는(rewrite) 과정이다. 첫째 과정은 결정적(deterministic)으로 기술되고 둘째 과정은 비결정적(nondeterministic)으로 기술된다.

첫째 부분:
```
set_unify(E){
  Ens:=E; Eaux:=∅; Es:=∅;
  while(Ens!=∅ || Eaux!=∅){
    if(Eaux!=∅) e:=pop(Eaux);
    else Ens에서 하나의 e를 제거;
    set_unify_actions(E, e);
  }
}
```

둘째 부분:
```
set_unify_actions(E,e){
  case e of
  /* X는 변수 */
  X=X → Ens:=Ens;
  /* t는 변수 아님 */
  t=X → Ens:=Ens∧(X=t);
  /* 함수 occur check */
  X=f(t1,...,tn) && X occurs in f(t1,...,tn)
    → fail
  /* 집합 occur check */
  X={t0,...,tn|t} && X occurs in t0,...,tn
    → fail
  /* 대치 */
  X=t && X does not occur in t →
    Es:=Es[X/t]∧(X=t);
    Ens:=Ens[X/t];
    Eaux:=Eaux[X/t];
  /* 집합 다시 쓰기 */
  X={t0,...,tn|X} && X not occur in t0,...,tn→
    push(X={t0, ..., tn|N}, Eaux);
  /* 다른 함수 */
  f(s1,...,sm)=g(t1,...,tn) && f≠g → fail
  /* 같은 함수 */
  f(s1,...,sn)=f(t1,...,tn) →
    Ens:=Ens∧(s1=t1∧...∧sn=tn);
  /* 집합 일치 */
  {t|s}={t'|s'} →
    AbClstep(E,{t|s}={t'|s'});
}

AbClstep(E,e){
  e:={t|s}={t'|s'};
  if(tail(s) and tail(s') not same var){
    choose one among
  1. Ens:=Ens∧(t=t'); push(s=s',Eaux);
  2. Ens:=Ens∧(t=t'); push({t|s}=s',Eaux);
  3. Ens:=Ens∧(t=t'); push(s={t'|s'},Eaux);
  4. push(s={t'|N},Eaux);
     push({t|N}=s',Eaux);
  } else {
    let X≡tail(s)≡tail(s');
    let t≡t0, ..., tm and t'≡t'0, ..., t'n;
    select arbitrarily i in {0,...,n};
    choose one among
  1. Ens:=Ens∧(t0=t'i);
     push({t1,...,tm|X}=
       {t'0,...,t'i-1,t'i+1,...,t'n|X},Eaux);
  2. Ens:=Ens∧(t0=t'i);
     push({t0,...,tm|X}=
       {t'0,...,t'i-1,t'i+1,...,t'n|X},Eaux);
  3. Ens:=Ens∧(t0=t'i);
     push({t1,...,tm|X}=
       {t'0,...,t'n|X},Eaux);
  4. push(X={t0|N},Eaux);
     push({t1,...,tm|N}=
       {t'0,...,t'n|N},Eaux);
  }
}
```

위의 AbClstep에서는 함수 tail을 사용하고, 2절에서 기술한 "Ab Cl 공리"와 동등한 의미를 가지는 "공리 E"를 사용한다. 함수 tail은 집합 텀(set term)을 계산하는데 사용하는 것으로써 다음과 같이 정의된다.

```
tail(∅)=∅
tail(X)=X        /*X는 변수*/
tail({S|T}) = tail(t)
```

공리 E는 다음과 같이 정의된다.

```
∀Y1Y2V1V2{Y1|V1} = {Y2|V2} ↔
                (Y1=Y2 ∧ V1=V2) ∨
                (Y1=Y2 ∧ V1={Y2|V2}) ∨
                (Y1=Y2 ∧ {Y1|V1}=V2) ∨
                ∃K(V1={Y2|K}∧V2={Y1|K})
```

4. Prolog 언어를 사용한 구현

본 절에서는 SICStus Prolog[9]를 사용하여 3절에서 기술한 집합 일치화 알고리즘을 구현하는 방법을 설명한다. 먼저 집합의 표현을 위한 자료 구조를 간단하게 설명한 후, 주요 술어의 구현 내용을 기술한다. 본 절의 끝에서는 시험 데이터를 사용하여 구현된 프로그램을 시험한 결과를 보인다.

4.1. 자료 구조

본 연구에서는 기존 연구[6, 7, 8]와는 달리 집합 일치화 구현 시 집합을 표현하기 위하여 Prolog 언어의 리스트(list)를 사용하였다. 리스트는 비수치의 프로그래밍(non-numeric programming)에서 광범위하게 사용되는 단순한 자료 구조이다[2]. 리스트는 빈 리스트 []와 비어 있지 않은 리스트 [H | T]로 표현할 수 있다. 여기서 H는 리스트의 첫 번째 요소이고, T는 리스트의 첫 번째 요소를 뺀 리스트이다. 본 연구에서는 2절에서 정의한 구문(syntax) 중 공집합을 표현한 심볼 "∅"과 함수 심볼 "{· |· }"을 리스트 []과 [· |·]로 표현하였다. 예를 들어 집합 {a, b, c}를 리스트로 표현하면 [a, b, c] 또는 [a | [b, c]]가 되며 집합 {X | {X | Z}}는 [X | [X | Z]]로 표현된다.

4.2. 주요 술어

본 연구에서 사용된 주요 술어는 solver/3 와 equal/6 이다.

4.2.1 solver/3

solver/3는 집합 일치화 알고리즘 부분에서 첫번째에 해당되는 부분으로서 주어진 집합 등식들 중에서 제일 왼쪽에 있는 하나를 선택하는 부분이다. 첫 번째 인수로 집합 등식들의 리스트를 주고, 두 번째 인수로 계산된 대답 대치(computed answer substitution)[3]를 받고, 세 번째 인수로 제한(constraints)을 받는다. solver/3는 solver_sub/5를 호출한다. solver_sub/5는 첫 번째 인수로 집합 등식들 중 제일 왼쪽에 있는 집합 등식을 solver/3에서 받고, 두 번째 인수로 빈 리스트 []를 주고, 세 번째 인수는 두 번째 인수로 받은 빈 리스트 []와 구성(composition)[3]하여 만들어진 계산된 대답 대치를 주고, 네 번째 인수로 집합 등식들의 리스트에서 제일 왼쪽에 있는 집합 등식을 뺀 나머지 집합 등식들로 이루어진 리스트를 solver/3에서 받고, 다섯 번째 인수로 제한을 받는다. solver_sub/5는 집합 등식 리스트들 중 제일 왼쪽부터 집합 등식 리스트가 빈 리스트 []일 때까지 집합 등식을 수행시킨다. solver_sub/5는 선택된 집합 등식을 수행시키기 위해 다시 쓰는(rewrite) 과정을 호출하게 되는데, 여기서 다시 쓰는 과정은 equal/6이다.

4.2.2 equal/6

equal/6는 비결정적으로 선택된 등식을 다시 쓰는(rewrite) 과정이다. equal/6에서 첫 번째 인수와 두 번째 인수를 받는데, 이 두 인수는 일치화(unification) 될 집합 텀(set term)들 이다. 세 번째 인수부터 여섯 번째 인수까지는 solver_sub/5의 두 번째 인수부터 다섯 번째 인수까지 와 같다. equal/6는 변수 처리 부분, occur check[3] 및 함수 일치화 실패 부분, 함수 처리 부분 그리고 AbClstep 부분 이렇게 네 부분으로 나눠서 설명할 수 있다.

- 변수 처리: equal/6의 첫 번째 인수와 두 번째 인수가 문자적으로 동등(literal equality)한 변수이면 제한(constraints)에서 삭제하고, 문자적으로 동등한 변수가 아니면 세 번째 인수와 "첫 번째 인수=두 번째 인수"라는 것을 구성(compose)하여 네 번째 인수에 넘긴다. equal/6의 첫 번째 인수에 변수가 아니고, 두 번째 인수가 변수일 경우, 첫 번째 인수와 두 번째 인수를 바꾸어 다시 equal/6을 호출한다. equal/6의 첫 번째 인수가 변수이고, 두 번째 인수가 첫 번째 인수를 집합 변수로 가지면, 그 집합 변수를 새로운 집합 변수로 바꾼다. 마지막으로 equal/6의 첫 번째 인수가 변수일 때 첫 번째 인수에 두 번째 인수를 배정한다.

- occur check 및 함수 일치화 실패: occur check은 함수 occur check와 집합 occur check로 나누어 구현하였다. 함수 occur check는 equal/6의 첫 번째 인수가 변수이고, 두 번째 인수가 변수가 아닌 텀일 때, 첫 번째 인수가 두 번째 인수의 하위 텀(sub term) 안에 존재하면 실패로 한다. 집합 occur check는 equal/6의 첫 번째 인수가 변수이면서 두 번째 인수인 집합 내에 존재하면 실패로 한다. 함수 일치화 실패 부분은 equal/6의 첫 번째 인수와 두 번째 인수가 함수 텀일 때, 두 텀의 함수 심볼이 문자적으로 동등하지 않으면 실패로 한다.

- 함수 처리: equal/6의 첫 번째 인수와 두 번째 인수가 변수가 아닌 함수 텀 일 때, 두 텀의 함수 심볼이 문자적으로 동등하면, 그 하위 텀을 equal_sub/6의 첫 번째와 두 번째 인수로 주어 호출한다. equal_sub/6은 하위 텀들을 서로 왼쪽부터 순서대로 equal/6의 첫 번째와 두 번째 인수로 주어 equal/6을 호출한다.

- AbClstep: AbClstep은 위 3절에서 정의한 "공리 E"를 이용하여 구현하였다. 이 부분은 tail/2를 이용하여 equal/6의 첫 번째 인수와 두 번째 인수의 tail을 얻은 후 비교하여 다른 경우와 같을 경우로 나누어 처리한다. tail이 다를 경우는 첫 번째 인수에서 받은 리스트와 두 번째 인수에서 받은 리스트를 "공리 E"와 같이 4가지 방법으로 equal/6를 호출한다. 4가지 방법 모두 equal/6을 두 번 호출하는데, 첫째, 둘째 그리고 셋째 방법은 첫 번째 인수에서 받은 리스트의 첫 번째 요소와 두 번째 인수에서 받은 리스트의 첫 번째 요소를 equal/6의 첫 번째와 두 번째 인수로 해서 equal/6를 호출하고, 첫째 방법은 equal/6의 첫 번째 인수에서 받은 리스트의 첫 번째 요소를 뺀 집합 변수와 두 번째 인수에서 받은 리스트의 첫 번째 요소를 뺀 집합 변수를 equal/6의 첫 번째와 두

번째 인수로 해서 equal/6을 호출하고 둘째 방법은 equal/6의 첫 번째 인수와 두 번째 인수에서 받은 리스트의 첫 번째 요소를 뺀 집합 변수를 equal/6의 첫 번째와 두 번째 인수로 해서 equal/6을 호출하며 셋째 방법은 equal/6의 첫번째 인수에서 받은 리스트의 첫 번째 요소를 뺀 집합 변수와 두 번째 인수를 equal/6의 첫 번째와 두 번째 인수로 해서 equal/6을 호출한다. 넷째 방법은 equal/6의 첫 번째 인수에서 받은 리스트의 첫 번째 요소를 뺀 집합 변수와 두 번째 인수에서 받은 리스트의 첫 번째 요소하고 새로운 집합 변수로 이루어진 리스트를 equal/6의 첫 번째와 두 번째 인수로 해서 equal/6을 호출하고 equal/6의 첫 번째 인수에서 받은 리스트의 첫 번째 요소하고 새로운 집합 변수로 이루어진 리스트와 두 번째 인수에서 받은 리스트의 첫 번째 요소를 뺀 집합 변수를 equal/6의 첫 번째와 두 번째 인수로 해서 equal/6을 호출한다.

tail이 같을 경우에는 equal/6의 두 번째 인수로 받은 리스트에서 임의적으로 한 요소를 선택하여 equal/6의 두 번째 인수로 주는 것만 다르고 거의 비슷하게 equal/6를 호출한다.

4.3. 보조 술어

보조 술어로 사용된 것은 tail/2, select/3, same_var/2, same_not_var/2, tail_check/3, tail_check_not/3, tail_rest/2, member_check/2, member_not/2, append/3, apply/3, sub_compose/3 등이다.

4.4. 동작 시험

동작 시험은 3절에서 기술한 집합 일치화 알고리즘을 고려하여 시험하였다.

- 경우 1: X는 변수
  입력: X=X, X=Y.
  출력: X=Y, Y=X
- 경우 2: t는 변수 아님
  입력: {a,b,c}=X.
  출력: X={a,b,c}
- 경우 3: 함수 occur check
  입력: {X} = {f(X)}.
  출력: no
- 경우 4: 집합 occur check
  입력: X={a,b,X,d}.
  출력: no
- 경우 5: 대치
  입력: X=Y, Y=Z, Z={a}, H={Z}.
  출력: H={{a}}, X={a}, Y={a}, Z={a}
- 경우 6: 집합 다시 쓰기
  입력: X={a,b,c\X}, Y={aa,bb,cc\Y}.
  출력: X={a,b,c_628}, Y={aa,bb,cc_1189},
  set(_628), set(_1189)
- 경우 7: 다른 함수
  입력: {f(a)} = {g(a)}.
  출력: no
- 경우 8: AbClstep중 tail이 다를 때
  입력: {T\S} = {T1\S1}.
  출력: S=S1, T=T1, S1=S, T1=T ;
  S=S, T=T1, S1={T\S}, T1=T ;
  S={T1\S1}, T=T1, S1=S1, T1=T ;
  S={T1_533}, T=T, S1={T_533}, T1=T1 and set(_533)
- 경우 9: AbClstep중 tail이 같을 때
  입력: {a,b,c} = {A,B,C}.
  출력: A=a, B=b, C=c ;
  A=a, B=c, C=b ;
  A=b, B=a, C=c ;
  A=c, B=a, C=b ;
  A=b, B=c, C=a ;
  A=c, B=b, C=a

5. 결론

본 논문은 "집합 일치화 문제(set unification problem)"[1]를 논리 언어[3] Prolog[2]를 사용하여 구현하는 방법에 대하여 기술하였다. 집합 일치화 문제는 고전적 논리 언어(logic languages)[3]에서 다루는 일치화 문제[4]를 집합의 표현까지 확장한 것으로 최근 연구되고 있는 "집합 제한 논리 언어(set constraints logic languages)"[5]를 구현하기 위하여 풀어야 하는 문제이다. 본 논문에서는 최근 A. Dovier 연구팀이 연구한 집합 일치화 문제의 풀이(solver)[1]를 소개하고, 이 풀이를 논리 언어 Prolog를 사용하여 구현하는 방법을 기술하였다. Prolog 언어는 비결정성(nondeterminism)을 표현할 수 있는 장점 때문에 다른 어떤 언어보다 쉽게 집합 일치화 문제 풀이를 구현할 수 있었다. 본 연구의 후속 연구로 본 연구에서 개발된 프로그램을 사용하여 집합 제한 논리 언어를 구현할 예정이다. 집합 제한 논리 언어는 본 논문에서 다룬 집합 일치화 제한 "=" 외에도 "∈" 제한, "$\cup_3$" 제한, "| |" 제한, "set" 제한 등이 포함되어 있다[5].

참고 문헌

[1] A. Dovier, E. Pontelli, and G. Rossi, Set Unification, Rapporto di Ricerca, Dipartimento di Matematica, Universita di Parma, n.310, 2002.

[2] I. Bratko, "PROLOG Programming for Artificial Intelligence", Addison-Wesley , 2000.

[3] C. J. Hogger, "Essentials of Logic Programming", Clarendon Press, 1990.

[4] F. Baader and W. Snyder, Unification theory, "Handbook of Automated Reasoning", Elsevier Science Publishers B. V., 1999.

[5] A. Dovier, C. Piazza, E. Pontelli and G. Rossi, Sets and Constraint Logic Programming, "ACM Transactions on Programming Languages and Systems", 22, 5, 861-931, 2000.

[6] G. Rossi, {log} User's Manual – Version 3.3, Rapporto di Ricerca, Dipartimento di Matematica, Universita di Rarma, n.233, 2000.

[7] G. Rossi, The {log} Programming Language, 1997.

[8] Log Home Page, http://math.unipr.it/~gianfr/setlog.Home.html

[9] SICStus Prolog User's Manual, http://www.sics.se/sicstus/docs/latest/html/sicstus.html