(Ab)(Cl) 집합 일치화의 구현에 관한 연구

신동하* · 김인영**

An Implementation of (Ab)(Cl) Set Unification

Dongha Shin* · Inyoung Kim**

본 연구는 2004년도 상명대학교 소프트웨어 · 미디어 연구소에서 연구비를 지원받았음

요 약

'집합'은 컴퓨터 프로그램의 설계에 자주 사용되는 도구이다. 이런 이유 때문에 최근 '집합 제한 언어'도 등장하였다. 본 연구에서는 '(Ab)(Cl) 집합 일치화' 문제를 소개하고 '집합 등식 다시쓰기(rewrite)'를 사용하여 집합 일치화를 Prolog 언어를 사용하여 구현하였다. 본 연구에서는 순차적 언어에서는 구현하기 힘들었던 집합 일치화가 Prolog 언어와 같은 논리 언어에서는 비결정성 제어 구조와 리스트 자료 구조를 사용하여 쉽게 구현 가능함을 보였다. 본 연구는 기존 구현이 고가의 상업용 Prolog를 사용한 것과는 달리 GNU 일반 공용 라이센스(GPL)를 가지는 Ciao Prolog를 사용하였기 때문에 누구나 무료로 사용할 수 있다는 장점도 가지고 있다. 현재 본 연구의 결과를 사용하여 '집합 제한 언어'가 구현 중이다.

ABSTRACT

'Set' is a tool that is used frequently in designing computer programs. Because of the reason, 'set constraints languages' have been developed recently. In this research, we introduce '(Ab)(Cl) set unification' problem and implement it using the 'set equation rewriting in Prolog'. In this research we shows that the set unification, that is considered to be difficult to be implemented in procedural languages, can be implemented easily using the non-deterministic control structure and the list data structure in logic language like Prolog. Our research uses the Ciao Prolog with GNU GPL, this is compared with other existing implementations which used expensive commercial Prolog, so anyone can use the result freely. Currently the result is being used for implementing a set constraint language.

키워드

집합(set), 집합 일치화(set unification), 집합 제한 언어(set constraint language), Prolog 언어

Ⅰ. 서 론

최근 '집합(set)'은 컴퓨터 프로그램의 설계 및 개발에 중요한 도구로 사용되고 있으며[1] 집합을 기반으로 하는 프로그래밍 언어인 '집합 제한 언어(set constraint language)'[2][3][4]도 등장하였다. 본

*상명대학교 소프트웨어학부 부교수

**상명대학교 대학원 컴퓨터과학과 석사

접수일자 2004. 4. 15

연구에서는 '집합 제한 언어'를 구현하기 위하여 풀어야 하는 문제인 '집합 일치화(set unification)'[5]의 구현을 다룬다. 본 연구에서는 집합 일치화 문제 중에서 가장 보편적으로 표현되는 문제인 '(Ab)(Cl) 집합 일치화'[6] 문제를 정의하고 '집합 등식'(set equation) 다시쓰기(rewrite) 방법[5][6]을 사용하여 집합 일치화를 Prolog 언어를 사용하여 구현하였다. 본 연구의 의의는 다음과 같다. 첫째, 일반 순차적 언어(procedural language)에서는 쉽게 구현하기 힘들었던 집합 일치화를 Prolog 언어와 같은 논리 언어(logic language)의 특성인 비결정성(non-determinism) 제어 구조와 리스트(list)라는 자료 구조[7][8]를 사용하여 쉽게 구현 가능함을 보였다는 점이고, 둘째, 본 연구의 결과는 논리 언어 기반 집합 제한 언어의 구현에 바로 사용될 수 있다는 점이다. 그리고 셋째, 본 연구에서는 고가의 상업용 Prolog[9][10]를 사용하지 않고 GNU GPL 라이센스[11]를 가지는 Prolog 언어인 Ciao Prolog[12]를 사용하였기 때문에 본 연구의 결과를 누구나 무료로 활용할 수 있다는 점이다.

본 논문의 II장에서는 '(Ab)(Cl) 집합 일치화' 문제의 구문(syntax)과 의미(semantics)를 정의한다. 일반적으로 집합은 집합의 생성을 표현하는 구성자(constructor)인 '{ · | · }'를 사용하여 표현하고 집합의 기본 공리인 '흡수(Ab: Absorption) 공리' 및 '좌측 교환(Cl: Commutative on left) 공리'를 사용하여 의미를 정의한다. 본 논문의 III장에서는 이들 공리에 바탕을 둔 집합 등식 다시쓰기(rewrite) 규칙을 소개한다. 이 규칙은 A. Dovier 연구팀이 제안한 것으로[5] 비결정성(non-determinism)이 내포된 규칙이다. 본 논문의 IV장에서는 다시쓰기 규칙을 바탕으로 집합 일치화 문제를 논리 언어 Prolog를 사용하여 구현하는 방법을 기술한다. Prolog 언어는 다른 언어와는 달리 집합의 표현과 유사한 리스트(list)라는 자료구조를 가지고 있고 비결정성을 쉽게 표현하는 제어구조를 가지고 있다. 본 연구에서 제안한 구현 방법은 이러한 Prolog 언어의 장점을 매우 잘 활용하는 방법이다. A. Dovier 연구팀도 Prolog 언어를 사용하여 집합 일치화를 구현[13]하였지만 Sicstus Prolog[14]라는 고가의 상업적인 언어를 사용하여 구현하였기 때문에 일반인들은 사용할 수 없었다. 본 연구에서는 GNU 라이센스를 가지고 있으면서도 매우 효율적이고 풍부한 기능을 가지고 있는 Ciao Prolog[15]를 사용하여 구현하였기 때문에 구현의 의의는 더 크다고 할 수 있다. 본 논문의 V장에서는 구현된 프로그램의 시험 방법 및 그 결과에 대하여 설명하였다. 마지막으로 VI장에서는 본 논문의 연구 결과 및 의의를 설명하고 앞으로의 연구 계획을 소개하였다.

II. 집합 일치화 문제

'집합 일치화' 문제[5]는 고전적 논리 언어(logic language)[16][17]에서 다루는 '일치화 문제(unification problem)'[18][19]를 집합의 의미를 내포하게 확장한 문제이다. (주: 여기서 소개하는 '(Ab)(Cl) 집합 일치화' 문제 풀이는 NP-complete 임이 증명되어 있다[13].)

1. 구문(Syntax)

'(Ab)(Cl) 집합 일치화' 문제는 아래와 같은 심볼을 사용하여 텀(term)과 등식(equation)을 표현한다.

- 변수 심볼: 영문 대문자로 시작되는 단어
- 함수 심볼: '{ · | · }', 'Φ', 사용자 정의 심볼
- 술어 심볼: '='
- 연결자 심볼: '∧'

집합 구성 함수 심볼을 사용하여 표현한 집합 {X1|X2}의 의미는 {X1}∪X2의 의미이며 여기서 X2는 항상 집합을 나타내는 텀이다. 그리고 편의상 {X1|{X2|{X3|...{Xn|Y}...}}}을 간단하게 {X1,X2,X3,...,Xn|Y}로 표현하기도 한다. 만약 여기서 Y=Φ이면 간단히 {X1,X2,X3,...,Xn}으로 표현한다. 등식(equation)은 '텀1=텀2'로 표현된다. 예를 들어 '{X1|{X2|{X3|Φ}}}={a|{b|{c|Φ}}}'는 하나의 등식이다. '(Ab)(Cl) 집합 일치화' 문제는 등식들을 연결자 '∧'을 사용하여 연결(conjunction)한 것이다. 예를 들면, '{X1|{X2|{X3|Φ}}}={a|{b|{c|Φ}}} ∧ X4=d'는 하나의 집합 일치화 문제의 표현이다.

2. 의미(Semantics)

앞 절에서 정의한 구문으로 표현된 집합은 아래에 표현된 공리 Ab(Absorption)와 공리 Cl(Commutative on left)을 만족한다.

- 공리 Ab: {X|{X|Z}}={X|Z}
- 공리 Cl: {X|{Y|Z}}={Y|{X|Z}}

공리 Ab와 공리 Cl은 집합의 성질을 그대로 나타내고 있으며 이들 두 공리를 합하여 간단히 '(Ab)(Cl) 공리'라고 한다. (Ab)(Cl) 공리를 같은 의미의 다른 표현으로 나타내면 다음과 같다.

{Y1|V1}={Y2|V2} ↔
(Y1=Y2 ∧ V1=V2) ∨
(Y1=Y2 ∧ V1={Y2|V2}) ∨
(Y1=Y2 ∧ {Y1|V1}=V2) ∨
(V1={Y2|N} ∧ V2={Y1|N})

III. 다시쓰기 규칙(Rewriting rule)

'다시쓰기 규칙'은 집합 일치화 문제를 구성하는 각 등식을 풀린 형식(solved form)이 될 때까지 다시 쓰는 규칙이다. 등식 X=t는 만약 X가 t 속에 나타나지 않으면 풀린 형식이라고 한다. 본 절에서 정의하는 다시쓰기 규칙은 A. Dovier 연구팀이 제안한 방식[5]으로 다시쓰기 규칙을 Es, Eaux, Ens라는 3개의 식을 사용하여 표현한다. 여기서 Es는 이미 풀린 등식의 리스트이고, Eaux는 곧 풀어야 할 등식의 리스트이며, Ens는 아직 풀지 않은 등식의 리스트이다. 초기에 Es와 Eaux는 빈 리스트이고 Ens는 주어진 집합 일치화 문제를 가지고 있다. 다시쓰기 규칙은 Eaux에 저장된 등식 중 하나를 제거하여(이때 Eaux가 빈 리스트이면 Ens에서 하나를 제거함) e에 저장한 후 e가 다음 중 하나의 조건을 만족하면 다시쓰기가 이루어진다. 아래의 다시쓰기는 Ens와 Eaux가 빈 리스트가 될 때까지 계속된다.

R1) X=X ↦ (참고: 제거됨)
R2) t=X ↦ Ens=Ens∧(X=t)
R3) X=f(t1,...,tn) ↦ fail (조건: X가 f(t1,...,tn)에 포함되어 있음)
R4) X={t1,...,tn|t} ↦ fail (조건: X가 t1,...,tn에 포함되어 있거나 t가 집합 팀이 아니고 X≠t이면서 X가 t에 포함되어 있음)
R5) X=t ↦ Es=Es[X/t]∧(X=t), Ens=Ens[X/t], Eaux=Eaux[X/t] (조건: X가 t에 포함되지 않음)
R6) X={t1,...,tn|X} ↦ Eaux=Eaux∧X={t1,...,tn|N} (조건: X가 t1,...,tn에 포함되지 않음)
R7) f(s1,...,sn)=g(t1,...,tm) ↦ fail (조건: f≠g)
R8) f(s1,...,sn)=f(t1,...,tn) ↦ Ens=Ens∧(s1=t1∧s2=t2∧...∧sn=tn)
R9) {t|s}={t'|s'} ↦ 다음 중 하나 선택함 (조건: tail(s)와 tail(s')이 같은 변수가 아님)
R9-1) Ens=Ens∧(t=t'), Eaux=Eaux∧s=s'
R9-2) Ens=Ens∧(t=t'), Eaux=Eaux∧{t|s}=s'
R9-3) Ens=Ens∧(t=t'), Eaux=Eaux∧s={t'|s'}
R9-4) Eaux=Eaux∧s={t'|N}∧{t|N}=s')
R10) {t0,...,tm|s}={t1',...,tn'|s'} ↦ {0,...,n}에서 하나를 선택하여 i라고 한 후 다음 중 하나를 선택함 (조건: tail(s)와 tail(s')이 같은 변수임)
R10-1) Ens=Ens∧(t0=ti'), Eaux=Eaux∧{t1,...,tm|X}={t0',...,ti-1',ti+1,...,tn'|X}
R10-2) Ens=Ens∧(t0=ti'), Eaux=Eaux∧{t0,...,tm|X}={t0',...,ti-1',ti+1,...,tn'|X}
R10-3) Ens=Ens∧(t0=ti'), Eaux=Eaux∧{t1,...,tm|X}={t0',...,tn'|X}
R10-4) Eaux=Eaux∧X={t0|N}∧{t1,...,tm|N}={t0',...,tn'|N}

앞의 다시쓰기 규칙에서 tail(s)는 다음과 같이 정의된다.

tail(t) =t (조건: t가 변수이거나 f(t1,...,tn)임)
tail(t2) (조건: t={t1|t2})

다시쓰기가 끝나면 Es에 저장되어 있는 X1=t1∧...∧Xn=tn 형식의 등식이 답 대치(answer substitution)가 된다.

앞의 다시쓰기 규칙 중 R1, R2, R3, R5, R7 및 R8은 집합이 포함되지 않는 기존 논리 언어의 일치화[16][17]를 위한 규칙이며, 집합의 도입과 함께 추가된 규칙은 R4, R6, R9 및 R10이다. 여기서 R4는 집합을 고려한 발생 점검(occur check)을 의미하며, R6은 동일한 의미로의 집합 변형을 의미하고, R9 및 R10은 (Ab)(Cl) 공리의 표현이다.

IV. 구현 방법

본 절에서는 집합 다시쓰기 규칙 중 제일 중요한 규칙인 R9 및 R10을 Prolog 언어를 사용하여 구현하는 방법을 기술한다(다른 규칙의 구현 설명은 생략함). 구현 방법이 비교적 용이한 이유는 Prolog 언어가 비결정성 제어 구조와 리스트 자료구조[7][8]를 제공하기 때문이다.

1. 집합의 표현

본 구현에서 집합은 다음과 같은 재귀적 규칙에 의하여 Prolog 텀으로 표현된다.

- 공집합 Φ는 snil로 표현된다.
- 집합 {X1|X2}는 set(X1,X2)로 표현된다.

예를 들어 집합 '{X1,X2,X3}'는 'set(X1,set(X2,set(X3,snil)))'로 표현된다.

집합 일치화 문제에서 사용되는 Ens, Eaux, Es는 등식의 리스트(list)로 표현된다. 예를 들어 '{X1,X2,X3}={a,b,c} ∧ X4=d'는 '[set(X1,set(X2,set(X3,snil)))=set(a,set(b,set(c,snil))),X4=d]'로 표현된다.

2. R9의 구현

R9를 구현하는 Prolog 술어는 'abcl_rewrite_nine'이며 이 술어의 인수(argument)는 순서적으로 다음과 같다.

i. Ens1: 입력 Ens (다시쓰기 전)
ii. Eaux1: 입력 Eaux (다시쓰기 전)
iii. Es1: 입력 Es (다시쓰기 전)
iv. set(T1,S1)=set(T2,S2): 선택된 집합 등식
v. Ens2: 출력 Ens (다시쓰기 후)
vi. Eaux2: 출력 Eaux (다시쓰기 후)
vii. Es2: 출력 Es (다시쓰기 후)

술어 'abcl_rewrite_nine'은 Prolog의 비결정성 제어 구조를 사용하며 다음과 같은 4개의 클로즈로 구현된다.

```
abcl_rewrite_nine(...) :-                    % R9-1
  Ens2=[T1=T2 | Ens1],
  Eaux2=[S1=S2 | Eaux1],
  Es2=Es1.
abcl_rewrite_nine(...) :-                    % R9-2
  Ens2=[T1=T2 | Ens1],
  Eaux2=[set(T1,S1)=S2 | Eaux1].
  Es2=Es1.
abcl_rewrite_nine(...) :-                    % R9-3
  Ens2=[T1=T2 | Ens1],
  Eaux2=[S1=set(T2,S2) | Eaux1],
  Es2=Es1.
abcl_rewrite_nine(...) :-                    % R9-4
  Ens2=Ens1,
  Eaux2=[set(T1,N)=S2,S1=set(T2,N) | Eaux1],
  Es2=Es1.
```

3. R10의 구현

R10을 구현하는 Prolog 술어는 'abcl_rewrite_ten'이며 이 술어의 인수는 앞의 'abcl_rewrite_nine'의 인수에 아래 2개의 인수가 추가된다.

viii. Select: 4번째 집합 등식 인수의 오른쪽 집합 텀 set(T2,S2)에서 선택된 i번째 텀 (여기서 i는 계속 변하여 불려짐)
ix. Remain: set(T2,S2)에서 i번째 텀이 제거된 텀

여기서 Select 및 Remain은 'abcl_rewrite_ten/9'이 불리기전 다음의 술어를 불러 계산한다. 이 술어의 첫째 인수는 주어지는 집합 텀이며 두 번째 및 세 번째 인수는 Select 및 Remain이다. 이 술어는 모든 가능한 Select와 Remain을 계산하여야 하는데 첫째 인수의 값에 따라 비결정성의 작동에 의하여 가능한 여러 개의 답을 찾아낸다.

```
select(set(Select,Remain),Select,Remain).
select(set(X,Y),Select1,set(X,Remain1)) :-
  select(Y,Select1,Remain1).
```

술어 'abcl_rewrite_ten'은 Prolog의 비결정성 제어 구조를 사용하며 다음과 같은 4개의 클로즈로 구현된다.

```
abcl_rewrite_ten(...) :-                     % R10-1
  Ens2=[T1=Select | Ens1],
  Eaux2=[S1=Remain | Eaux1],
  Es2=Es1.
abcl_rewrite_ten(...) :-                     % R10-2
  Ens2=[T1=Select | Ens1],
  Eaux2=[set(T1,S1)=Remain | Eaux1],
  Es2=Es1.
abcl_rewrite_ten(...) :-                     % R10-3
  Ens2=[T1=Select | Ens1],
  Eaux2=[S1=set(T2,S2) | Eaux1],
  Es2=Es1.
abcl_rewrite_ten(...) :-                     % R10-4
  tail(S1,X),
  Eauxt=[X=set(T1,N) | Eaux1],
  replace_tail(set(T1,S1),N,set(_,S11)),
  replace_tail(set(T2,S2),N,set(T22,S22)),
  Eaux2=[S11=set(T22,S22) | Eauxt].
```

V. 시 험

본 장에서는 앞 장에서 구현한 프로그램의 동작을 시험한다. 시험은 각 다시쓰기 규칙의 구현이 정확한지를 시험할 수 있는 문제를 입력하여 그 대답 대치가 정확한지를 점검하는 방법을 사용한다. 규칙 R6, R9 및 R10을 시험한 결과는 다음과 같다.

```
R6 시험:
?- X=set(a,set(b,set(c,X))),Y=set(d,Y).
Y=set(d,_631), X=set(a,set(b,set(c,_517))) ? ;
no
R9 시험:
?- set(X1,set(X2,set(X3,snil)))=set(a,set(b,snil)).
X1=a, X2=b, X3=b ? ;
X1=a, X2=a, X3=b ? ;
X1=a, X2=b, X3=a ? ;
X1=b, X2=a, X3=a ? ;
X1=b, X2=a, X3=b ? ;
X1=b, X2=b, X3=a ? ;
no
?- set(X1,X2)=set(X3,X4).
X1=X3, X2=X4 ? ;
X1=X3, X4=set(X3,X2) ? ;
X1=X3, X2=set(X3,X4) ? ;
X4=set(X1,_1627), X2=set(X3,_1627) ? ;
no
R10 시험:
?- set(X1,N)=set(X2,N).
X1=X2 ? ;
X1=X2, N=set(X2,_3052) ? ;
X1=X2, N=set(X2,_3047) ? ;
N=set(X1,set(X2,_3062)) ? ;
no
?- set(X1,set(X2,N))=set(X2,N).
X1=X2, N=set(X2,_3286) ? ;
X1=X2, N=set(X2,set(X2,_3290)) ? ;
X1=X2 ? ;
X1=X2, N=set(X2,_3332) ? ;
X1=X2, N=set(X2,_3327) ? ;
X1=X2, N=set(X2,set(X2,_3342)) ? ;
N=set(X1,_3264) ? ;
N=set(X1,set(X2,_3471)) ? ;
N=set(X1,set(X2,_3345)) ? ;
N=set(X1,set(X2,set(X2,_3360))) ? ;
no
```

VI. 결 론

본 논문은 '(Ab)(Cl) 집합 일치화' 문제를 논리 언어인 Prolog 언어를 사용하여 구현하는 방법에 대해 기술하였다. '집합 일치화' 문제는 고전적 논리 언어(logic languages)에서 다루는 '일치화 문제(unification problem)'에 집합의 의미를 확장한 것으로 '집합 제한 언어'를 구현하기 위해서 풀어야 할 문제이다. 본 논문에서는 '(Ab)(Cl) 집합 일치화' 문제의 의미(syntax)와 구문(semantics)를 정의하고, A. Dovier 연구팀이 제안한 집합 등식 다시쓰기(rewrite) 규칙을 소개하고, 이 규칙을 Ciao Prolog를 사용하여 구현하는 방법을 기술하였다. 본 연구로 인해 집합 일치화를 논리 언어의 특성인 비결정성(non-determinism) 제어 구조와 리스트(list)라는 자료 구조를 사용하여 쉽게 구현 가능함을 보였을 뿐만 아니라 고가의 상업용 Prolog를 사용하지 않고 GNU 일반 공용 라이센스(GPL)를 가지는 Ciao Prolog를 사용하여 구현하였기 때문에 누구나 무료로 활용할 수 있다. 본 연구의 후속 연구로 '집합 제한 논리 언어(set constraints logic language)'를 구현할 예정이다. 본 연구의 결과로 '집합 제한 논리 언어'를 구현하는데 바로 사용될 수 있다.

참고문헌

[1] A. Dovier, Computable Set Theory and Logic Programming, PhD Thesis TD-1/96, Universita degli Studi di Pisa, dip. di Informatica, 1996. March.

[2] Pat M. Hill and John W. Lloyd, The Godel Programming Language, MIT Press, 1994.

[3] Escher NG Manual, http://users.unimi.it/~ddl/vega/manual/escher_ng/

[4] Mozart Documentation, http://www.mozart-oz.org/documentation/index.html

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

[6] 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.

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

[8] M. F. Clocksin and C. S. Mellish, "Programming in Prolog, fourth edition", Springer-Verlag, 1994.

[9] SICStus Prolog Homepage, http://www.sics.

se/isl/sicstuswww/site/index.html
[10] Quintus Prolog Homepage, http://www.sics.se/quintus/
[11] GNU General Public License, http://www.gnu.org/copyleft/gpl.html
[12] The CIAO Prolog Development System WWW Site, http://clip.dia.fi.upm.es/Software/Ciao/
[13] A. Dovier, E.G. Omodeo, E. Pontelli, and G. Rossi, {log}: A Language for Programming in Logic with Finite Sets, "The Journal of Logic Programming", 28(1), 1-44, 1996.
[14] SICStus Prolog User's Manual, http://www.sics.se/sicstus/docs/latest/html/sicstus.html
[15] The CIAO Prolog System Manual, http://clip.dia.fi.upm.es/Software/Ciao/ciao_html/ciao_toc.html
[16] Chin-Liang Chang and Richard Lee, "Symbolic Logic and Mechanical Theorem Proving", Academic Press, 1973.
[17] V. Sperschneider and G. Antoniou, "Logic: A Foundation for Computer Science", Addison-wesley, 1991.
[18] F. Baader and W. Snyder, Unification theory, "Handbook of Automated Reasoning", Elsevier Science Publishers B. V., 1999.
[19] J. W. Lloyd, "Foundations of Logic Programming", Springer-Verlag, 1984.

저자소개

신동하(Dongha Shin)

1980년 경북대학교 전자공학과 전자계산기전공 졸업(학사)
1982년 서울대학교 대학원 전자계산기공학과 졸업(석사)
1994년 University of South Carolina 컴퓨터과학과 졸업(박사)
1982년~1996년 한국전자통신연구원 책임연구원
1991년~1994년 University of South Carolina, TA
1997년~현재 상명대학교 소프트웨어학부 부교수
※관심분야: 프로그래밍 언어, 컴파일러, 계산이론, 인공지능, 내장형 소프트웨어, 공개 소프트웨어 프로젝트

김인영(Inyoung Kim)

2002년 상명대학교 가정교육학과 및 소프트웨어학과 졸업(학사)
2004년 상명대학교 대학원 컴퓨터과학과 졸업(석사)
2002년~2003년 상명대학교 연구 조교
※관심분야: 집합 제한 언어, 내장형 소프트웨어, 공개 소프트웨어 프로젝트