Prolog 언어를 사용한 집합 제한 논리 언어의 구현

김인영* · 신동하**
*상명대학교 일반대학원 컴퓨터과학과
**상명대학교 소프트웨어학부

An Implementation of Set Constraints Logic Language Using Prolog

Inyoung Kim*, Dongha Shin**
*Dept. of Computer Science, Sangmyung University
**Division of Computer Software, Sangmyung University
E-mail : esirizad@smu.ac.kr

요 약

본 논문은 "집합 제한 논리 언어(set constraints logic language)"를 논리 언어 Prolog를 사용하여 구현하는 방법을 기술한다. "집합 제한 논리 언어"는 "집합 이론(set theory)"을 사용하여 프로그래밍 할 수 있는 새로운 파라다임(paradigm)의 프로그래밍 언어이다. 본 논문에서는 최근 A. Dovier 연구팀이 제안한 "집합 제한 문제 풀이(set constraints problem solver)" 방법을 설명하고 이 풀이를 논리 언어 Prolog를 사용하여 구현하는 방법을 기술한다. Prolog 언어는 비결정적(nondeterministic)으로 표현되는 프로그램을 쉽게 구현할 수 있고 리스트(list)라는 자료 구조를 제공하기 때문에 쉽게 "집합 제한 문제 풀이"를 구현할 수 있다. 본 연구에서는 개발한 언어의 유용성을 보이고자 여러 가지 응용 분야에 본 언어를 적용하여 보았다.

ABSTRACT

In this paper, we describe an implementation method of "set constraints logic language" using the logic language Prolog. "Set constraints logic language" is a programming language with a new paradigm that uses the "set theory" in programming. In this paper, we explain "set constraints problem solver" that has been proposed by A. Dovier and his researchers and we describe an implementation method of this solver using Prolog. We can easily implement the "set constraints problem solver" in Prolog, since Prolog easily implements nondeterministic problems and provides a data structure called list. We have applied the language to several application fields to show the usefulness of the language.

키워드
Set Constraints Logic Language, Set Constraints Problem Solver, Prolog

Ⅰ. 서 론

본 논문은 "집합 제한 논리 언어(set constraints logic language)"[1][2][3]를 논리 언어[4] Prolog[5]를 사용하여 구현하는 방법을 기술한다. 최근 연구[1][2][3][6]가 활발히 이루어지고 있는 "집합 제한 논리 언어"는 "집합 이론(set theory)"[7]을 사용하여 프로그래밍 할 수 있는 새로운 파라다임(paradigm)의 프로그래밍 언어이다. 본 논문은 "집합 제한 논리 언어"를 표현하기 위한 언어의 구문(syntax) 및 의미(semantics)를 설명하고, 최근 A. Dovier 연구팀이 제안한 "집합 제한 문제 풀이(set constraint problem solver)" 방법을 설명한다. 본 논문은 "집합 제한 문제 풀이"를 Ciao Prolog 언어를 사용하여 구현한다. Prolog 언어는 비결정적(nondeterministic)으로 표현할 수 있고 리스트(list)라는 자료 구조를 제공하기 때문에 다른 언어보다 매우 쉽게 "집합 제한 문제 풀이"를 구현할 수 있

다[8].

본 논문의 2장에서는 "집합 제한 논리 언어"를 표현하기 위한 언어의 구문 및 의미를 설명한다. 3장에서는 2장에서 정의한 구문과 의미를 가지고 "집합 제한 문제 풀이"에 대하여 설명한다. 이 "집합 제한 문제 풀이"는 A. Dovier 연구팀이 제안하고 그 완전성(completeness)을 증명한 것으로써, 비결정적(nondeterministic)인 부분과 결정적(deterministic)인 부분으로 나누어 기술되어있다. 4장에서는 3장에서 설명된 "집합 제한 문제 풀이"를 논리 언어 Prolog를 사용하여 구현한 방법을 기술하고, 그 동작을 시험해본다. 5장에서는 본 연구의 결과 및 앞으로 연구 계획을 기술한다.

II. 집합 언어의 구문과 의미

"집합 제한 논리 언어"에서 다루는 "집합 언어(set language)"는 함수 심볼(function symbol), 술어 심볼(predicate symbol), 변수 심볼(variable symbol)로 표현된다. 함수 심볼은 집합을 나타내는 함수 심볼 "{_|_}", 공집합을 표현하는 함수 심볼 "Φ", 그리고 사용자가 정의하는 함수 심볼이 있다. 술어 심볼은 각 제한들(constraints)을 표현하기 위한 "=","∈","∉","≠","∪3","¬∪3","∥","∦" 그리고 "set(·)"이 있다. 그리고 변수 심볼은 보통 영문의 대문자로 표현한다. 집합 {X|Y}의 의미는 {X}∪Y를 의미하고 집합 {t1 | {t2 | ...{tn | t}...}는 t가 "Φ"이면 {t1, t2, ..., tn}으로 표현한다[8]. 우리의 집합 제한 논리 언어로 구현하는 프로그램(program)은 제한들의 연결(conjunction)로 구성된다. 예를 들어, "X={a,b,c}∧Y∈X∧b≠Y."은 하나의 집합 제한 논리 언어로 구현한 프로그램이다.

앞에서 정의한 언어를 사용하여 표현된 집합 제한 중 집합 등식(e.g. {X|Y}={X|Z})은 아래 두 개의 공리(axiom)인 Ab(Absorption)와 Cl(Commutative on left)을 만족한다. 즉 아래 두 공리는 집합 일치화의 의미를 표현한다[8].

공리 Ab : {X | {X | Z}} = {X | Z}

공리 Cl : {X | {Y | Z}} = {Y | {X | Z}}

III. 집합 제한 문제 풀이

본 장에서는 A. Dovier 연구팀이 제안하고 완전성을 증명[1][7][9]한 "집합 제한 문제 풀이"를 기술한다. 집합 제한 문제 풀이는 제한들 중 하나를 선택하여 맞는 제한 규칙(constraint rule)을 적용하는 부분과 선택된 제한(constraint)을 해당 제한 규칙에 의해 다시 쓰는(rewriting) 부분으로 구성된다. 전자는 결정적으로 기술되고, 후자는 결정적인 부분과 비결정적인 부분으로 기술된다.

전자는 다음과 같이 기술된다. C는 제한을 나타내는 것으로 해당 제한 규칙에 적용된다. STEP은 주어진 C가 문제 해결된 형식(solved form)이 될 때까지 반복한다. ";"은 분리(disjunction)을 나타내는 것으로 ";"은 "∧"과는 반대의 의미이다.

STEP(C) : eq(C); in(C); nin(C); neq(C); set(C);
union(C); ununion(C); joint(C); disjoint(C);

다음은 후자인 각 제한의 규칙들을 기술한다. 제한은 "=","∈","∉","≠","∪3","¬∪3","∥","∦" 그리고 "set(·)"으로 총 9가지이다. 각 규칙은 "↦"을 중심으로 왼쪽 부분이 해결할 문제이고, 오른쪽 부분은 그 문제를 다시 쓰는 부분이다. C'은 "∧"으로 연결된 제한들이고, X, Y, Z는 변수이고, t, ti, s, si는 일반 텀들(generic terms)이고, f, g는 함수 심볼들(function symbols)이고, N, N1, N2는 새로 생성된 변수이다[1].

"=" 제한은 두 집합 또는 텀(term)의 동등함을 제한한다. "=" 제한 규칙은 아래와 같다. E9와 E10 부분은 2장에서 정의한 공리 Ab와 공리Cl을 사용하여 기술되었다.

E1) $X=X\wedge C' \mapsto C'$

E2) $t=X\wedge C'$, t is not a variable $\mapsto X=t\wedge C'$

E3) $X=f(t_1,...,t_n)\wedge C'$, $f\neq\{\_|\_\}$, $X\in vars(t_1,...,t_n)$ $\mapsto$ fail

E4) $X=\{t_0,...,t_n|t\}\wedge C'$, t is Φ or variable, $X\in vars(t_0,...,t_n) \mapsto$ fail

E5) $X=t\wedge C'$, $X\notin vars(t)$, t is a set term or $set(X)\notin C' \mapsto X=t\wedge C'[X/t]$

E6) $X=\{t_0,...,t_n|X\}\wedge C'$, $X\notin vars(t_0,...,t_n)$ $\mapsto X=\{t_0,...,t_n|N\}\wedge set(N)\wedge C'$

E7) $f(s_1,...,s_m)=g(t_1,...,t_n)\wedge C'$, $f\neq g \mapsto$ fail

E8) $f(s_1,...,s_n)=f(t_1,...,t_n)\wedge C'$, $f\neq\{\_|\_\}$ $\mapsto s_1=t_1\wedge...\wedge s_n=t_n\wedge C'$

E9) $\{t|s\}=\{t'|s'\}\wedge C'$, tail(s),tail(s') are not the same var $\mapsto C'\wedge$ any of

$E9_1$) $t=t'\wedge s=s'$

$E9_2$) $t=t'\wedge\{t|s\}=s'$

$E9_3$) $t=t'\wedge s=\{t'|s'\}$

$E9_4$) $s=\{t'|N\}\wedge\{t|N\}=s'\wedge set(N)$

E10) $\{t_0,...,t_m|X\}=\{t'_0,...,t'_n|X\}\wedge C' \mapsto C'\wedge$ any of

$E10_1$) $t_0=t'_j\wedge\{t_1,...,t_m|X\}=\{t'_0,...,t'_{j-1},t'_{j+1},...,t'_n|X\}$

$E10_2$) $t_0=t'_j\wedge\{t_0,...,t_m|X\}=\{t'_0,...,t'_{j-1},t'_{j+1},...,t'_n|X\}$

$E10_3$) $t_0=t'_j\wedge\{t_1,...,t_m|X\}=\{t'_0,...,t'_n|X\}$

$E10_4$) $X=\{t_0|N\}\wedge\{t_1,...,t_m|N\}=\{t'_0,...,t'_n|N\}\wedge$ set(N) for any j in {0, ..., n}

"∈" 제한은 한 원소가 한 집합 안의 원소임을 제한한다. "∈" 제한 규칙은 아래와 같다.

M1) $s\in\Phi\wedge C' \mapsto$ false

M2) $r\in\{s|t\}\wedge C' \mapsto C'\wedge$ any of

$M2_1$) r=s

$M2_2$) $r\in t$

M3) $t\in X\wedge C' \mapsto X=\{t|N\}\wedge set(N)\wedge C'$

"$\not\in$" 제한은 위의 "$\in$" 제한의 부정(negation)임을 제한한다. "$\not\in$" 제한 규칙은 아래와 같다.

MN1) $s\not\in\Phi\wedge C' \mapsto C'$
MN2) $r\not\in\{s\mid t\}\wedge C' \mapsto r\neq s\wedge r\not\in t\wedge C'$
MN3) $t\not\in X\wedge C'$, $X\in vars(t) \mapsto C'$

"$\neq$" 제한은 위의 "=" 제한의 부정임을 제한한다. "$\neq$" 제한 규칙은 아래와 같다.

EN1) $f(s_1,...,s_m)\neq g(t_1,...,t_n)\wedge C' \mapsto C'$
EN2) $f(s_1,...,s_n)\neq f(t_1,...,t_n)\wedge C'$, $f\neq\{\_\mid\_\}$, $n>0$
$\mapsto C'\wedge$any of
$EN2_1$) $s_1\neq t_1$
...
$EN2_n$) $s_n\neq t_n$
EN3) $s\neq s\wedge C'$, s is a constant or a variable
$\mapsto$ fail
EN4) $t\neq X\wedge C'$, t is not a variable $\mapsto X\neq t\wedge C'$
EN5) $X\neq f(t_1,...,t_n)\wedge C'$, $f\neq\{\_\mid\_\}$, $X\in vars(t_1,...,t_n)$
$\mapsto C'$
EN6) $X\neq\{t_1,...,t_n\mid t\}\wedge C'$, $X\in vars(t_1,...,t_n) \mapsto C'$
EN7) $X\neq\{t_1,...,t_n\mid X\}\wedge C'$, $X\not\in vars(t_1,...,t_n)$
$\mapsto C'\wedge$any of
$EN7_1$) $t_1\neq X$
...
$EN7_n$) $t_n\neq X$
EN8) $\{s\mid r\}\neq\{u\mid t\}\wedge C' \mapsto C'\wedge$any of
$EN8_1$) $N\in\{s\mid r\}\wedge N\not\in\{u\mid t\}$
$EN8_2$) $N\in\{u\mid t\}\wedge N\not\in\{s\mid r\}$

"$\cup_3$" 제한은 첫 번째 인수와 두 번째 인수의 합집합(union)이 세 번째 인수라는 것을 제한한다. "$\cup_3$" 제한 규칙은 아래와 같다.

U1) $\cup_3(s,s,t)\wedge C' \mapsto s=t\wedge C'$
U2) $\cup_3(s,t,\Phi)\wedge C'$, $s\neq t \mapsto s=\Phi\wedge t=\Phi\wedge C'$
U3) $\cup_3(\Phi,t,X)\wedge C'$ or $\cup_3(t,\Phi,X)\wedge C'$, $t\neq\Phi$
$\mapsto X=t\wedge C'$
U4) $\cup_3(s_1,s_2,\{t_1\mid t_2\})\wedge C'$, $s_1\neq s_2$
$\mapsto \{t_1\mid t_2\}=\{t_1\mid N\}\wedge t_1\not\in N\wedge C'\wedge$any of
$U4_1$) $s_1=\{t_1\mid N_1\}\wedge t_1\not\in N_1\wedge\cup_3(N_1,s_2,N)$
$U4_2$) $s_2=\{t_1\mid N_1\}\wedge t_1\not\in N_1\wedge\cup_3(s_1,N_1,N)$
$U4_3$) $s_1=\{t_1\mid N_1\}\wedge t_1\not\in N_1\wedge s_2=\{t_1\mid N_2\}\wedge t_1\not\in N_2$
$\wedge\cup_3(N_1,N_2,N)$
U5) $\cup_3(\{t_1\mid t_2\},t,X)\wedge C'$ or $\cup_3(t,\{t_1\mid t_2\},X)\wedge C'$,
$t\neq\{t_1\mid t_2\}, t\neq\Phi \mapsto \{t_1\mid t_2\}=\{t_1\mid N_1\}\wedge t_1\not\in N_1$
$\wedge X=\{t_1\mid N\}\wedge t_1\not\in N\wedge C'\wedge$any of
$U5_1$) $t_1\not\in t\wedge\cup_3(N_1,t,N)$
$U5_2$) $t=\{t_1\mid N_2\}\wedge t_1\not\in N_2\wedge\cup_3(N_1,N_2,N)$
U6) $\cup_3(X,Y,Z)\wedge Z\neq t\wedge C'$, $X\neq Y$
$\mapsto \cup_3(X,Y,Z)\wedge C'\wedge$any of
$U6_1$) $N\in Z\wedge N\not\in t$
$U6_2$) $N\in t\wedge N\not\in Z$
$U6_3$) $Z=\Phi\wedge t\neq\Phi$
U7) $\cup_3(X,Y,Z)\wedge X\neq t\wedge C'$ or $\cup_3(Y,X,Z)\wedge X\neq t$
$\wedge C'$, $X\neq Y \mapsto \cup_3(X,Y,Z)\wedge C'\wedge$any of
$U7_1$) $N\in X\wedge N\not\in t$
$U7_2$) $N\in t\wedge N\not\in X$
$U7_3$) $X=\Phi\wedge t\neq\Phi$

"$\neg\cup_3$" 제한은 위의 "$\cup_3$" 제한의 부정임을 제한한다. "$\neg\cup_3$" 제한 규칙은 아래와 같다.

UN1) $\neg\cup_3(s_1,s_2,s_3)\wedge C' \mapsto C'\wedge$any of
$UN1_1$) $N\in s_3\wedge N\not\in s_1\wedge N\not\in s_2$
$UN1_2$) $N\in s_1\wedge N\not\in s_3$
$UN1_3$) $N\in s_2\wedge N\not\in s_3$

"$\parallel$" 제한은 두 집합의 교집합(intersection)이 공집합이라고 제한한다. "$\parallel$" 제한 규칙은 아래와 같다.

D1) $\Phi\parallel t\wedge C'$ or $t\parallel\Phi\wedge C' \mapsto C'$
D2) $X\parallel X\wedge C' \mapsto X=\Phi\wedge C'$
D3) $\{t_1\mid t_2\}\parallel X\wedge C'$ or $X\parallel\{t_1\mid t_2\}\wedge C'$
$\mapsto t_1\not\in X\wedge X\parallel t_2\wedge C'$
D4) $\{t_1\mid s_1\}\parallel\{t_2\mid s_2\}\wedge C'$
$\mapsto t_1\neq t_2\wedge t_1\not\in s_2\wedge t_2\not\in s_1\wedge s_1\parallel s_2\wedge C'$

"$\nparallel$" 제한은 위의 "$\parallel$" 제한의 부정임을 제한한다. "$\nparallel$" 제한 규칙은 아래와 같다.

DN1) $s\nparallel t\wedge C' \mapsto N\in s\wedge N\in t\wedge C'$

"set(·)" 제한은 인수가 집합 또는 집합 변수라고 제한한다. "set(·)" 제한 규칙은 아래와 같다.

S1) $set(\Phi)\wedge C' \mapsto C'$
S2) $set(\{t\mid s\})\wedge C' \mapsto set(s)\wedge C'$
S3) $set(f(t_1,...,t_m))\wedge C'$, $f\neq\{\_\mid\_\}$, $f\neq\Phi \mapsto$ error

IV. 구현 및 동작시험

본 장에서는 Ciao Prolog[10]를 사용하여 3장에서 기술한 "집합 제한 문제 풀이"를 구현하는 방법을 기술한다. 우선 집합을 표현하기 위한 자료구조를 간단히 설명한 후, 주요 술어의 구현 방법을 기술하고, 마지막으로 구현된 프로그램의 동작을 시험한다.

1. 자료구조

본 연구에서는 기존 연구[11][12][13]와는 달리 집합을 표현하기 위해서 Prolog 언어의 리스트(list)와 연구자가 정의한 함수를 사용하였다. 일반적인 집합 표현은 set([...],_)으로 나타내고, 공집합은 set([],[]) 또는 []([](empty list)로 나타낸다. set([...],_)에서 첫 번째 인수는 원소들(elements)을 가지는 리스트이고, 두 번째 인수는 집합 또는 집합 변수이다. 예를 들어, {a|{b|{c|Z}}}은 set([a],set([b],set([c],Z))) 또는 set([a,b,c],Z)으로 표현된다. 이때 Z는 집합 변수이다. 제한 술어(constraint predicate) 구현에서 "="은 eq(_,_), "$\in$"은 in(_,_), "$\not\in$"은 nin(_,_), "$\neq$"은 neq(_,_), "$\cup$3"은 union(_,_,_), "$\neg\cup$3"은 ununion(_,_,_), "$\parallel$"은 disjoint(_,_), "$\nparallel$"은 joint(_,_), 그리고 "set(·)"은

set(_)으로 바꾸어 구현하였다. set([...],_)이라는 자료구조를 사용함으로써 eq(_,_)구현시 규칙 E9와 규칙 E10에서 술어 tail을 사용하지 않고 쉽게 구현할 수 있었다.

2. 주요 술어

본 연구에서 사용된 주요 술어에는 solver/2, eq/4, in/4, nin/4, neq/4, union/5, ununion/5, disjoint/4, joint/4 그리고 set/3이 있다. 위의 주요 술어들은 3장에서 기술된 "집합 제한 문제 풀이"를 대부분 반영하여 구현하였다. 제한 규칙 중 eq/4를 제외한 나머지 제한 규칙들은 다른 제한 규칙을 사용하기 때문에 eq/4를 먼저 구현하고, 특히 많이 사용되는 in/4와 nin/4를 그 다음으로 구현하는 것이 좋다.

2.1 solver/2

solver/2는 제한 리스트(constraints list)를 받아서 제한 리스트 중 하나의 제한을 선택하여 해당 제한 규칙에 적용시키는 부분으로 결정적으로 구현하였다. solver/2의 두 인수는 둘 다 리스트로 구성된다. 첫 번째 인수는 초기값을 받는 부분이고, 두 번째 인수는 문제 해결된 형식(solved form)을 받는 부분이다. 두 리스트는 다시 세 부분으로 나누어지는데, 첫 번째는 문제 해결된 형식 리스트(solved form list)이고, 두 번째와 세 번째는 문제 해결할 형식 리스트이다. 세 번째는 스택(stack)[6]처럼 사용된다. 첫 번째와 세 번째 리스트는 초기값으로 [](empty list)를 주고, 두 번째 리스트는 풀어야할 제한 리스트를 준다. solver/2에서는 solver_sub/2를 호출하여 두 번째와 세 번째 리스트가 빈 리스트가 될 때까지 solver_sub/2 호출을 반복한다. solver_sub/2는 선택된 제한을 해당 제한 규칙에 적용시킨다.

set/3을 제외한 각 제한 규칙의 세 번째 인수와 네 번째 인수는 solver_sub/2에서 받은 부분과 돌려줄(return)부분이다. set/3은 두 번째 인수와 세 번째 인수이다.

2.2 eq/4

eq/4는 첫 번째 인수와 두 번째 인수가 동등함을 나타낸다. eq/4는 변수처리, occur check, 함수처리 그리고 AbCl처리 부분으로 나누어 구현되었다.

2.3 in/4

in/4는 첫 번째 인수가 두 번째 인수의 원소가 됨을 나타낸다. in/4는 공집합처리, 변수처리, 집합처리로 나누어 구현되었고, 다시 쓰는(rewriting) 과정에서 eq/4를 사용한다.

2.4 nin/4

nin/4는 첫 번째 인수가 두 번째 인수의 원소가 되지 않음을 나타낸다. nin/4는 in/4와 같이 세 부분으로 나누어 구현되었고, 다시 쓰는 과정에서 neq/4를 사용한다.

2.5 neq/4

neq/4는 첫 번째 인수와 두 번째 인수가 동등하지 않음을 나타낸다. neq/4는 변수처리, 함수처리, 집합처리 부분으로 나누어 구현되었고, 다시 쓰는 과정에서 in/4와 nin/4를 사용한다.

2.6 union/5

union/5는 첫 번째 인수와 두 번째 인수의 합집합이 세 번째 인수임을 나타낸다. union/5는 동등처리, 공집합처리, 집합처리, 변수처리로 나누어 구현되었고, 다시 쓰는 과정에서 eq/4, neq/4, in/4, nin/4를 사용한다.

2.7 ununion/5

ununion/5는 한 원소가 세 번째 인수의 원소이고 첫 번째 인수와 두 번째 인수의 원소가 아니거나, 첫 번째 인수의 원소이고 세 번째 인수의 원소가 아니거나, 두 번째 인수의 원소이고 세 번째 인수의 원소가 아님을 나타낸다. ununion/5는 다시 쓰는 과정에서 in/4, nin/4를 사용한다.

2.8 disjoint/4

disjoint/4는 첫 번째 인수와 두 번째 인수의 교집합이 공집합임을 나타낸다. disjoint/4는 공집합처리, 변수처리, 집합처리로 나누어 구현되었고, 다시 쓰는 과정에서 eq/4, neq/4, nin/4를 사용한다.

2.9 joint/4

joint/4는 첫 번째 인수와 두 번째 인수의 교집합이 공집합이 아님을 나타낸다. joint/4는 다시 쓰는 과정에서 in/4를 사용한다.

2.10 set/3

set/3은 첫 번째 인수가 집합 또는 집합 변수임을 나타낸다. set/3은 공집합처리, 집합처리로 나누어 구현되었다.

3. 동작 시험

동작 시험은 여러 응용분야 중 NP-complete 문제인 지도 색칠하기(*map coloring*) 문제[1][3]와 간단한 예를 가지고 한다.

3.1 지도 색칠하기 문제

지도 색칠하기 문제는 "집합 제한 논리 언어"를 사용하여 수행시켜볼 수 있다. 예는 남한의 5도(province)로 한다. 지역은 경기도(GG), 강원도(GW), 충청도(CC), 전라도(JL) 그리고 경상도(GS)이고, 인접한 지역으로 경기도는 강원도와 충청도이고, 강원도는 경기도, 충청도, 경상도이고, 충청도는 경기도, 강원도, 전라도, 경상도이고, 전라도는 충청도와 경상도이고, 경상도는 강원도, 충청도, 전라도이다. 색은 빨강, 파랑, 초록을 사용하였다. 두 가지 색 사용시 수행결과는 "no"가 나오고 세

가지 색 사용시 수행결과는 아래와 같다. 아래의 결과로 남한의 5도를 색칠할 수 있는 최소한의 색의 수는 3가지임을 알 수 있다.

```
-입력: eq(Regions,set([GG,GW,CC,JL,GS],[])),
       eq(Regions,set([red,blue,green],[])),
       nin(GG,set([GW,CC],[])),
       nin(GW,set([GG,CC,GS],[])),
       nin(CC,set([GG,GW,JL,GS],[])),
       nin(JL,set([CC,GS],[])),
       nin(GS,set([GW,CC,JL],[])).
 출력:
 GG=red,GW=blue,CC=green,JL=blue,GS=red,
 Regions=set([red,blue,green,blue,red],[]) ;
 GG=red,GW=green,CC=blue,JL=green,GS=red,
 Regions=set([red,green,blue,green,red],[]) ;
 GG=blue,GW=red,CC=green,JL=red,GS=blue,
 Regions=set([blue,red,green,red,blue],[]) ;
 GG=blue,GW=green,CC=red,JL=green,GS=blue,
 Regions=set([blue,green,red,green,blue],[]) ;
 GG=green,GW=red,CC=blue,JL=red,GS=green,
 Regions=set([green,red,blue,red,green],[]) ;
 GG=green,GW=blue,CC=red,JL=blue,GS=green,
 Regions=set([green,blue,red,blue,green],[])
```

3.2 간단한 예

```
-입력: eq(X,set([a,b,c],[])), in(Y,X), neq(b,Y).
 출력: Y=a,X=set([a,b,c],[]) ; Y=c,X=set([a,b,c],[])
-입력: in(X,Y), nin(X,set([a,b],[])).
 출력: Y=set([X],_385),set(_385),neq(X,a),
       neq(X,b)
-입력: union(set([a,b],[]),set([c],[]),X),
       disjoint(X,Y).
 출력: nin(c,Y),nin(b,Y),nin(a,Y),set(Y),
       X=set([a],set([b],set([c],[])))
-입력: in(X,set([2,4,6],[])), multi(X,X,Y).
 출력: Y=4,X=2 ; Y=16,X=4 ; Y=36,X=6
```

V. 결 론

본 논문은 "집합 제한 논리 언어(set constraints logic language)"를 논리 언어 Prolog를 사용하여 구현하는 방법을 기술하였다. "집합 제한 논리 언어"는 "집합 이론(set theory)"을 사용하여 프로그래밍 할 수 있는 새로운 컴퓨터 프로그래밍 언어이다. 본 논문에서는 "집합 제한 문제 풀이(set constraint problem solver)" 방법을 설명하고, 이 풀이를 논리 언어인 Ciao Prolog를 사용하여 구현하는 방법을 기술하였다. 본 연구는 Prolog 언어의 자료 구조 중 하나인 리스트(list)와 본 연구자가 정의한 자료 구조를 사용하여 쉽게 구현할 수 있었다. 현재는 "집합 제한 문제 풀이"에 대해서만 구현되어있는데, 후속 연구로 집합 계산뿐만 아니라 Prolog 언어에서와 같은 심볼(symbol) 계산도 자유롭게 표현할 수 있는 언어로 확장 개발할 예정이다.

참고 문헌

[1] 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.

[2] A. Dovier, C. Piazza, E. Pontelli, and G. Rossi, On the Representation and Management of Finite Sets in CLP-languages, Conference and Symposium on Logic Programming, 40-54, The MIT Press, 1998.

[3] 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.

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

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

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

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

[8] 김인영, 신동하, Prolog 언어를 사용한 집합 일치화의 구현, 춘계학술대회논문집, 10/1, 한국정보처리학회, 2003. 5.

[9] R. Carmona, A. Dovier and G. Rossi, Dealing with Infinite Intensional Sets in CLP, Joint Conference on Declarative Programming, 1997.

[10] CIAO Prolog System Manual, http://clip.dia.fi.upm.es/Software/Ciao/ciao_html/ciao_toc.html

[11] G. Rossi, {log} Users Manual Version 3.3, Rapporto di Ricerca, Dipartimento di Matematica, Università di Parma, n.233, 2000.

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

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