울산대학교 연구논문집 제21권(자연과학·공학편) 제1호 pp. 55~63, 1990.
UOU Report Vol.21, (Natural Science and Engineeing) No.1 pp, 55~63, 1990.
University of Ulsan

컴퓨터에 의한 자동적인정리증명에 관한 연구

고 재 진
전 자 계 산 학 과

〈요 약〉

이 논문에서는 먼저 자동정리증명에 관한 일반적인 개념을 서술하고, 그것의 응용에 대해서 논한다. 그 다음에 자동정리증명의 역사적 발전과정을 고찰하고, 최신의 여러가지의 자동정리증명 방법들을 비교 검토하고, 그들의 장단점에 대해서 논한다. 그리고, 어떤 정리를 예로 들고 그것의 논리적 표준형에의 변화과정을 서술하고, 컴퓨터에 있는 자동정리증명기에 그 표준형을 입력하여 자동정리 증명하는 예를 보였다. 결론적으로 자동정리증명방법의 개선에 대한 앞으로의 연구방향을 제시하였다.

A Study On the Automatic Theorem Proving By Computer

Koh, Jae-Jin
Dept. of Computer Sciene

〈**Abstract**〉

In this paper first we describe the general concepts of automatic theorem proving and discuss the applications of it. Then we investigate the historical stages of developments in automatic theorem proving methods, examine and compare the various modern automatic theorem proving methods and discuss the merits and the defects of them. Then we describe a sample theorem and the conversion process of it into logical standard forms and show a proof example of the logical standard forms by an automatic theorem prover in computer. Consequently, we proposed the future research directions on the refinements of automatic theorem proving methods.

Ⅰ. 서 론

컴퓨터는 복잡한 수치계산뿐만 아니라, 컴퓨터 기술의 발전에의해서, 인간의 지능을 필요로 하는 문제를 해결하는 데에도 이용하게 되었다. 그런문제중의 하나가 정리증명(theorem proving)이다. 정리증명에 대한 기초이론은 1930년 Herbrand[5]에 의해서 개발되었는데, 그의 방법은 컴퓨터가 출현되기전에는 실용성이 없었다. 1965년 Robinson[7]이 융해원리(resolution principle)를 고안함으로서 컴퓨터에 의한 실용적인 자동정리증명기가 개발되기 시작하였다. 자동정리증명 기술의 발전과 더불어, 이 기술의 응용이 인공지능의 여러 분야에 파급되었다. 자동정리증명 기술은 기호논리(symbolic logic)에 기초를 두고 있고, 이 기호 논리는 문제를 표현하고, 그것의 해답을 얻는데 사용된다.

예를 들어서 다음과 같은 사실이 있다고 하자.

F1 : 공자는 사람이다.

F2 : 모든 사람은 죽게된다.

F1과 F2를 표현하기 위해서는 프레디케이트

(predicate)라는 개념이 필요하다.

P(x) : x는 사람이다.

Q(x) : x는 죽게된다.

∀x : 모든 x에 대해서

따라서, F1과 F2는 다음과 같이 표현된다.

F1 : P(공자)

F2 : (∀x)(P(x)→Q(x))

여기서, → 기호는 imply (~이면, ~이다)를 뜻한다.

그리고, "공자는 죽게되는가?"라는 질문 F3는 다음과 같이 표현된다.

F3 : Q(공자) ?

여기서, F1과 F2의 논리적결과로 F3의 해답은 "예"가 된다. 앞의 예에서, 어떤 논리식(논리적 사실을 표현한 식)이 다른 논리식들로 부터의 논리적 결과로 유도된다는 것을 보았다. 어떤 논리식들로 부터의 논리적 결과로 어떤 논리식이 유도된다는 사실을 기술한 것을 정리(theorem)라고 한다. 정리가 사실이라는 것을 보여주는 것을 정리증명(theorem proving)이라고 한다. 자동정리증명이란 정리증명을 자동적인(기계적인)방법을 써서 하는 것을 말하고, 정리증명을 하는데 드는 인간의 노력을 경감시킬 수 있다. 자동정리증명은 전산학 뿐만아니라, 자연과학, 공학, 사회과학등에 폭 넓게 응용할 수 있고, 어떤 사실들을 수집하고, 그런사실들로 부터 어떤 논리적결과를 유도하여야 하는 분야에는 항상 이용할 수 있다.

II. 1계 논리의 기초개념

이 절에서는 자동정리증명 방법등에서 사용되는 1계 논리의 기초개념에 대해서 서술한다.

<정의 1>항(term)은 다음과 같이 정의 된다.

(1) 상수는 항이다.

(2) 변수는 항이다.

(3) f가 n계 함수이고, t1, ……, tn이 항일때 f(t1, ……, tn)은 항이다.

<정의 2>프레디키트(predicate)는 상수들의 나열을 T(참)나 F(거짓)으로 사상하는 함수이다.

<정의 3>p가 n계 프레디키트이고, t1, ……, tn이 항들일때 P(t1, ……, tn)을 원자논리식(atomic formula)라고 한다.

<정의 4>정형논리식(well formed formulas), 약해서 논리식은 다음과 같이 정의된다.

(1) 원자논리식은 논리식이다.

(2) F와 G가 논리식이라면 ~(F), (F∨G), (F∧G), (F→G), (F↔G)도 논리식이다.

여기서 ~는 not, ∨은 or, ∧은 and, →은 imply, ↔은 equivalence를 의미한다.

(3) F가 논리식이고, x가 F에 있는 자유변수라면, (∀x)F와 (∃x)F도 논리식이다.

여기서 ∀x는 for all x, ∃x는 there exists x such that을 의미한다.

<정의 5>논리식 F의 해석(interpretation)은 다음의 것으로 구성된다.

(1) 정의역 D

(2) 각 상수에 대해서 정의역 D에 있는 원소를 배정한다.

(3) n계 함수에 대해서는 D^n에서 D로 가는 사상을 배정한다.

(4) n계 프레디키트에 대해서는 D^n에서 {T, F}로 가는 사상을 배정한다.

<정의 6>논리식 G가 모순이 없는(consistent) 또는 만족할(satisfiable)필요충분조건은 해석 I가 존재해서, 논리식 G가 해석 I에 의해서 T(참)로 되는 것이다. 이런 경우 해석I를 논리식 G의 모델(model)이라고 하고, 해석I는 논리식G를 만족 시킨다고 한다.

<정의 7>논리식 G가 모순이 있는 (inconsistent) 또는 비만족할(unsatisfiable)필요충분조건은 논리식 G를 만족시킬 해석이 존재하지 않는 것이다.

<정의 8>논리식 G가 유효할(valid)필요충분조건은 G의 모든 해석이 G를 만족시키는 것이다.

<정의 9>논리식 G가 논리식 F1, ……, Fn들의 논리적 결과(logical consequence)일 필요충분조건은 모든 해석I에 대해서 F1∧, ……, ∧Fn이 I에 의해서 참이면, G가 I에 의해서 참인 것이다.

<정의 10>리터랄(literal)이란 원자논리식 또는 ~원자논리식을 의미한다.

<정의 11>논리절(clause)란 리터랄들을 ∨로 연결한 것을 말한다.

<정의 12>논리식 F가 정규형(prenex normal form)일 필요충분조건은 다음과 같다. 논리식 F는 다음의 형태로 되어 있고, (Q1 X1)……(Qn Xn) (M)

여기서 (Qi Xi), i=1, ……, n은 (∀Xi)이거나 (∃Xi)이고, M은 정량자(quantifiers)가 없는 논리식이다.

<정의 13>논리식 F가 스콜램 표준형(Skolem standard form)일 필요충분조건은 다음과 같다.

F는 정규형으로 되어 있고, 정규형의 M은 결합정규형(conjunctive normal form)이고, 존재정량자(existential quantifiers)는 그 앞에 있는 범용정량자(universal quantifiers)들의 스콜렘 함수(Skolem function)로 대치된다.

논리절들의 집합 S는 논리절들을 Λ로 연결하는 것을 의미하기 때문에 논리절들의 집합 S는 스콜렘 표준형으로 표현할 수 있다. 논리절들의 집합의 예를 들면 다음과 같다.

{~P(x, f(x))vR(x, f(x), g(x)), Q(x, g(x))vR(x, f(x), g(x))}

이것은 다음과 같은 스콜렘 표준형으로 표현할 수 있다.

(~P(x, f(x))vR(x, f(x), g(x)) Λ (Q(x, g(x))vR(x, f(x), g(x)))

III. 자동정리 증명 방법의 발전과정

정리증명에 의해서 어떤 문제를 푸는 방법은 1계 논리(first-order logic)를 이용하여, 문제를 스콜렘 표준형의 논리식으로 표현하여서, 그 논리식이 유효한지 또는 모순이 있는지를 증명하는 것이다. 논리식의 유효성 또는 모순성을 증명하는일반적인 결정절차를 발견하려는 시도는 Leibniz(1646~1716) Peano(1900년경), Hibert(1920년경)등에 의해서 시도되어 왔다.그러나, 이런시도는 불가능함이 Church (1936)와 Turing(1936)에 의해서 증명되었다. 즉, 1계 논리의 논리식의 유효성을 증명하는 일반적인 결정 절차는 존재하지 않음이 증명되었다. 그러나 논리식이 유효하다면, 그 논리식이 유효함을 증명하는 절차는 있음을 발견하였다. 정의 8에 의하면 유효한 논리식이란 모든 해석 하에서 그 논리식이 참인 것을 의미한다. 1930년에 Herbrand는 자동정리증명의 중요한 절차를 발견하였다. 그는 주어진 논리식을 거짓으로 만드는 해석을 찾는 알고리즘을 개발하였다. 즉, 주어진 논리식이 유효하다면 그 논리식을 거짓으로 만드는 해석은 존재하지 않고, 그의 알고리즘은 유한 시도후에 정지한다. Gilmore[4]는 주어진 논리식의 유효성을 부정된 논리식(negated formula)의 모순성을 찾음으로서 증명하는 알고리즘을 설계했다. 그의 알고리즘은 부정된 논리식의 instances들을 주기적으로 생성해서 모순성을 시험했다. 그러나, 그의 알고리즘은 비능률적임이 판명되었다. Gilmore의 방법은 Davis와 Putnan[3]에 의해서 개량되었지만 역시 비능률적이었다. 자동정리증명의 중요한 혁신은 Robinson[7]이 용해원리를 도입함으로서 이루어졌다. 용해원리를 이용한 증명절차는 앞의 방법들 보다 능률적이었다. 그후로 용해증명절차의 단점들을 보완한 여러가지의 개량된 용해방법들이 제시되었는데, 의미적 용해(semantic resolution)[8], 자물쇠 용해(lock resolution)[1], 선형용해(linear resolution)[6], 단위용해(unit resolution)[2], 보조집합전략 (set-of-support strategy)[9]등이 그것들이다.

IV. 최근의 자동정리증명 방법

최근의 자동정리증명 방법들은 Robinson의 용해원리 증명방법을 기초로 해서, 그것을 개량한 방법들이다. 그러면 용해원리(resolution principle)서부터 차례대로 소개해 나가기로 한다.

1. 용해원리방법

<정의 14>치환(substitution)이란 다음과 같은 형태의 유한 집합을 말한다.

{t1/v1, ……, t_n/v_n}

여기서, 각 vi는 변수이고, 각 ti는 vi와 다른 항을 의미한다.

<정의 15>치환 θ가 논리표현들의 집합 {E1, ……, Ek}의 통일자(unifier)가 될 필요충분조건은 다음과 같다.

E1θ = E2θ = …… = Ekθ

<정의 16>논리표현들의 집합{E1, ……, Ek}의 통일자 σ가 범용통일자(most general unifier)가 될 필요 충분조건은 다음과 같다. 위의 집합에 대한 각 통일자 θ에 대해서, 치환 λ가 존재해서 θ = σ · λ 가 된다.

용해원리는 다음의 정의에 기초를 두고 있다.

<정의 17>C1과 C2는 서로 공통변수가 없는 2개의 논리절이라고 하고, L1과 L2는 각각 C1과 C2에 있는 리터랄이라고 하자. 만약 L1과 ~L2가 범용통일 σ를 갖는다면, 다음의 논리절을 C1과 C2의 용해된 논리절(binary resolvent)이라 한다.

(C1σ-L1σ) U (C2σ-L2σ)

<예 1>용해 된 논리절의 예

C1 = P(x) v Q(x), C2 = ~P(a) v R(y)

L1 = P(x), L2 = ~P(a)

L1과 ~L2는 범용통일자 σ = {a/x}를 갖기 때문에 C1과 C2의 용해 된 논리절은 다음과 같다.

(C1σ-L1 · σ) U (C2σ-L2σ) = Q(a) v R(y)

용해원리는 논리절들의 집합으로부터 용해된 논리절들을 생성하는 추론규칙(inference rule)이다. 그리고, 이 원리는 모순 있는 논리절들의 집합으로 부터 공논리절(empty clause)을 항상 생성하기 때문에, 증명할 정리를 부정 해서, 그 부정된 정리를 논리절들의 집합으로 변환해서, 이 집합에 용해원리를 적용하면, 자동정리증명을 효율적으로 할 수 있다. 논리절들의 집합 S에 대해서 용해원리를 바로 적용한 것으로 수준포화용해(level—saturation resolution) 방법이 있는데, 다음과 같다.

$S^0 = S$

$S^n = \{C1$과 $C2$의 용해된 논리절들 $\mid C1 \in (S^0 \cup \cdots\cdots \cup S^{n-1})$ and $C2 \in S^{n-1}\}$

위의 절차를 공논리절이 발견될때까지 반복하게 된다. 그러나, 이 방법의 무조건 적용은 많은 불필요한 논리절들을 생성하기 때문에 이런 낭비를 방지할 조치들을 생각하지 않으면 안된다. 이런 조치에는 삭제전략(deletion strategy)이 있는데, 이 전략은 가능하면 생성된 논리절 중에서, 상진리치(tautology)나 포함된(subsumed)논리절들을 제거하는 것이다.

2. 의미적 용해(semantic resolution)

용해방법에 삭제전략을 적용할때는 이미 불필요한 논리절들이 생성된 후이고, 상진리치 시험과 포함시험을 하는데 많은 비용이 들기 때문에, 처음부터 불필요한 논리절들이 생성되지 못하도록 하는 방법이 강구되었는데, 그중의 하나가 의미적 용해이다.

〈예 2〉용해의 예

$S = \{\sim P \vee \sim Q \vee R,\ P \vee R,\ Q \vee R,\ \sim R\}$

S의 모순성을 증명하기위한 용해의 적용은 다음과 같다.

(1) $\sim P \vee \sim Q \vee R$

(2) $P \vee R$

(3) $Q \vee R$

(4) $\sim R$

(5) $\sim Q \vee R$ (1) 과 (2) 의 용해

(6) $\sim P \vee R$ (1) 과 (3) 의 용해

(7) $\sim P \vee \sim Q$ (1) 과 (4) 의 용해

(8) P (2) 과 (4) 의 용해

(9) Q (2) 과 (4) 의 용해

(10) $\sim Q \vee R$ (1) 과 (8) 의 용해

(11) $\sim P \vee R$ (1) 과 (9) 의 용해

(12) R (2) 과 (6) 의 용해

(13) $\sim Q$ (4) 과 (5) 의 용해

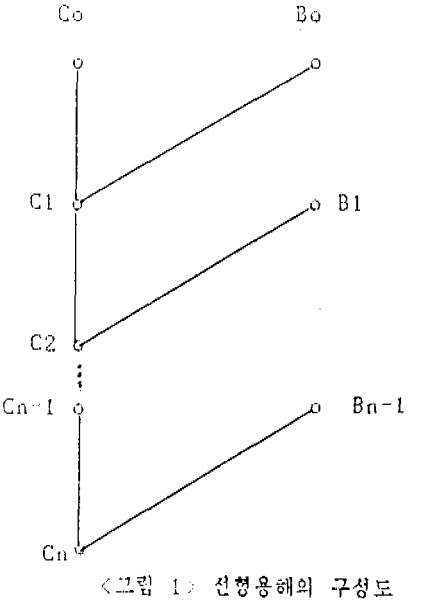
(14) $\sim P$ (4) 과 (6) 의 용해

(15) □(공논리절)(4) 과 (12) 의 용해

예 2에서 생성된 논리절 중에서 (6) 과 (12) 만이 공논리절의 유도에 사용되었다. 생성된 다른 논리절들은 불필요한 것들이다. 의미적 용해에서는 이러한 불필요한 논리절들의 생성을 될 수 있으면 막도록 한다. 그러면 의미적 용해의 개념을 설명하도록 한다. 첫째 증명의 대상이되는 집합 S에 있는 논리절들을 어떤 해석을 이용해서 2개의 그룹 S1과 S2로 나누도록 한다. S1은 해석에 의해서 참이되는 논리절의 그룹이고, S2는 해석에 의해서 거짓이 되는 그룹이다. 그리고, 같은 그룹내의 논리절 사이의 용해를 금지시킨다. 둘째 프레디키트 사이에 순서를 주어서, 용해를 위해서 S1에서 선택된 논리절의 리터랄은, 그 논리절내에서 가장 큰 리터랄 이란 조건을 둔다. 세째 다른 용해순서에 의해서 같은 논리절을 생성하는 논리절의 모임을 충돌(clash)이라고 하는데, 이 충돌개념을 이용하면 중복된 용해를 방지할 수 있다. 위와 같은 3가지의 규제에 의해서 의미적 용해는 원래의 용해 보다 훨씬 능률적으로 용해를 수행할 수 있다.

3. 선형용해(linear resolution)

선형용해는 연결추론(chain reasoning)의 개념을 이용한 것으로서 어떤 논리절을 시발점으로해서, 대응논리절을 선택해서 용해된 논리절을 얻는 과정을 공논리절을 얻을때까지 반복하는 것이다. 이 방법에는 경험적 지혜(heuristics)를 적용할 수 있다는 장점이 있다.



〈그림 1〉 선형용해의 구성도

〈정의 18〉

S : 논리절들의 집합

Co : S에 있는 한 논리절

최상논리절(top clause)를 Co로 하는 S로 부터의 선형유도(linear deduction) Cn은 그림 1과 같은 유도이고, 다음의 조건을 만족한다.

(1) Ci+1, i=0, ……, n−1은 Ci(중앙논리절)와 Bi(옆논리절)와의 융해된 논리절이다.

(2) 각 Bi는 S에 있는 논리절이거나, j<i인 어떤 Cj이다.

〈예 3〉 선형융해의 예

$S=\{P \vee Q,\ \sim P \vee Q,\ P \vee Q,\ \sim P \vee \sim Q\}$

최상논리절 : P v Q

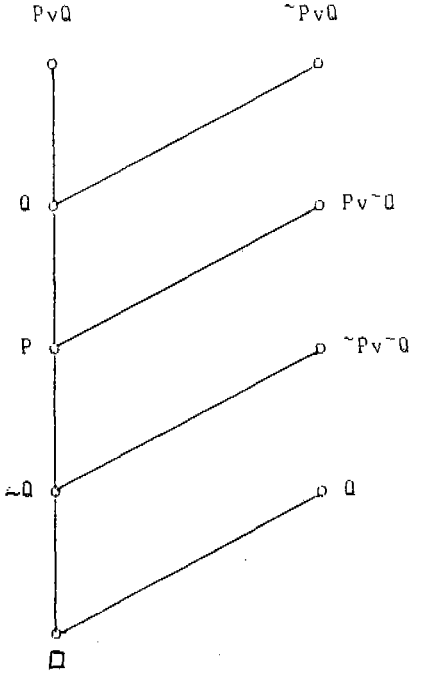


그림2〉 선형융해의 예

4. 개량된 선형융해

선형융해에 다음의 2개의 개념을 첨가한것을 개량된 선형융해라 한다. 첫째는 순서있는 논리절의 개념이고, 둘째는 융해된 리터랄에 대한 정보를 이용하는 것이다. 이 2가지의 개념은 선형융해의 효율을 상당히 향상시킨다. 순서있는 논리절이란 논리절에 있는 리터랄들을 나열된 대로 순서를 주고, 뒷쪽에 있는 리터랄이 앞쪽에 있는 리터랄 보다 순서가 크다는 뜻이다. 다음과 같은 2개의 순서있는 논리절이 있다고 하자.

C1=P v Q

C2=~Q v R

C1의 Q와 C2의 ~Q를 서로 융해하여서 된 논리절은 P v R이 된다. 여기서, C1에 있는 융해되는 리터랄은 순서가 가장 큰 Q이어야 한다.

그리고, 순서있는 융해된 논리절은 P v [Q] v R로 표시하는데, [Q]는 융해된 리터랄을 의미한다.

그리고, 네모친 리터럴 다음에 네모없는 리터럴이 없는 경우에는 네모친 리터럴을 제거할 수 있다.

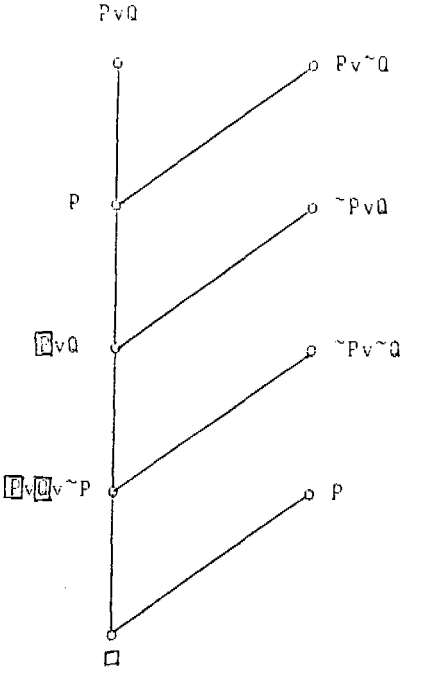
순서있는 논리절과 융해된 리터랄에 대한 정보를 사용하는 유도를 OL—유도(Orderd linear deduction)라고 한다.

〈예 4〉OL -유도의 예

S={P v Q, P v ~Q, ~P v Q, ~P v ~Q}

최상논리절 : P v Q

S에 대한 OL—유도는 그림 3에 있다.



〈그림 3〉 OL -유도의 예

그림 3에서 중앙논리절에서 용해되는 리터럴은 제일 뒤에 있는 리터럴이고, 중앙논리절 [P] v [Q] v ~P에서 [P]와 ~P는 서로 상반되는 짝을 이룬다. 이 경우의 옆 논리절은 앞서있는 중앙논리절 이어야 하고, 이 경우의 옆논리절은 P이다.

V. 컴퓨터에 의한 자동처리증명의 예

이 절에서는 증명해야 할 정리를 서술하고, 그 정리를 1계 논리의 스콜렘 표준형으로 변환하는 과정을 기술하고, Lisp언어로 작성된 단위용해방법의 자동정리증명기를 써서 Pyramid 컴퓨터에서 정리증명을 실행한 예를 기술한다.

(정 리)

단위원소를 갖는 결합법칙이 성립하는 시스템에서, 모든원소의 세곱이 단위원소가 된다면, 그 시스템은 교환법칙이 성립한다.

이 정리를 스콜렘표준형으로 변환하기 위해서 기호화를 해야한다.

이시스템을 S라 하고, S가 만족하는 전제조건들은 다음과 같다.

A1(결합법칙) : x, y, z∈ S이면 $x \cdot (y \cdot z) = (x \cdot y) \cdot z$이다.

A2(단위원소) : 단위원소 e와 모든 x∈ S 에 대해서 $x \cdot e = e \cdot x = x$이다.

그리고, 프레디키트 P(x, y, z)는 $x \cdot y = z$를 나타내도록 한다.

위의 전제조건들을 논리식으로 나타내면 다음과 같다.

A1 : $(\forall x)(\forall y)(\forall z)(\forall u)(\forall v)(\forall w)$
$(P(x,y,u)\ P(y,z,v)\ P(u,z,w) \rightarrow P(x,v,w))$
$\wedge (\forall x)(\forall y)(\forall z)(\forall u)(\forall v)(\forall w)$
$(P(x,y,u)\ P(y,z,v)\ P(x,v,w) \rightarrow P(u,z,w))$

A2 : $(\forall x)P(x,e,x) \wedge (\forall x)P(e,x,x)$

위의 정리를 논리식으로 나타내면 다음과 같다.

T : $(\forall x)P(x,x,e) \rightarrow ((\forall u)(\forall v)(\forall w)(P(u,v,w) \rightarrow P(v,u,w)))$

우리가 증명해야 할 논리식은 다음과 같다.

F : $A1 \wedge A2 \rightarrow T$

우리는 F를 부정한 것에서 모순성을 찾음으로서 증명을 하는데, 이런 증명방법을 반박증명방법(refuation proof method)이라고 한다. f의 부정은 다음과 같다.

$\sim F : \sim(\sim A1 \wedge A2) \vee T) = A1 \wedge A2 \wedge \sim T$

A1, A2~T를 스콜렘표준형으로 나타내면 다음과 같다.

A1 : {$\sim P(x,y,u) \vee \sim P(y,z,v) \vee \sim P(u,z,w) \vee P(x,v,w)$,
$\sim P(x,y,u) \vee \sim P(y,z,v) \vee \sim P(x,v,w) \vee P(u,z,w)$}

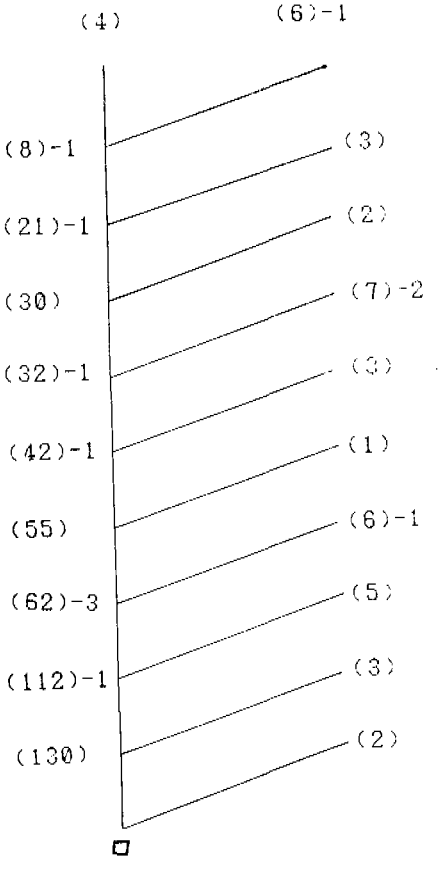
A2 : {$P(x,e,x)$, $P(e,x,x)$}

~T : {$P(x,x,e)$, $P(a,b,c)$, $\sim P(b,a,c)$}

따라서, 우리가 모순성을 증명해야 할 ~F는 다음과 같은 표준 논리절로 구성된다.

(1) P(e, x, x)

(2) P(x, e, x)



<그림 4>단위용해방법에 의한 반박증명

여기서(n1)—n2형식에서 n1은 논리절의 번호이고, n2는 용해되는 리터럴 위치를 나타내는 번호이다. 예를들면(62)—3은 논리절의 번호는 62이고, 용해되는 리터럴의 번호는 3번째 리터럴 이라는 뜻이다.

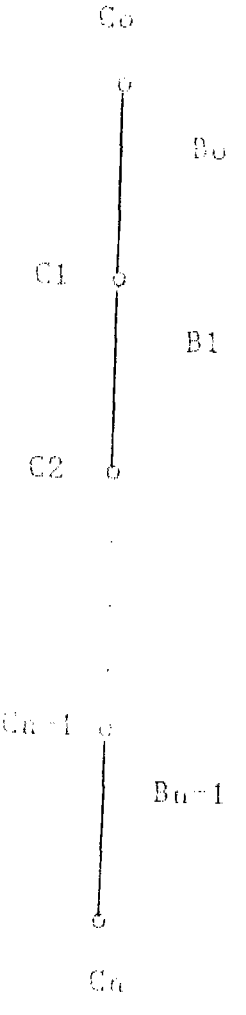
(3) P(x, x, e)
(4) P(a, b, c)
(5) −P(b, a, c)
(6) ~P(x, y, u) v ~P(y, z, v) v ~P(x, v, w) v P(u, z, w)
(7) ~P(x, y, u) v ~P(y, z, v) v ~P(u, z, w) v P(x, v, w)

이것을 단위용해방법을 사용한 자동정리증명기로 Pyramid 콤퓨터에서 반박증명을 실행한 결과는 그림 4에 있다.

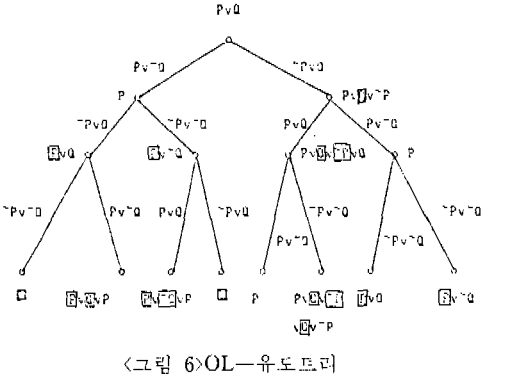
VI. 연구방향

자동정리증명을 효율적으로 수행하도록 하기위한 앞으로의 연구방향은 OL—유도를 어떻게하면 효율적으로 구현하느냐 하는 문제를 연구하는 것이다. 우선 OL—유도에서 최상논리절 Co를 어떻게 선정하느냐 하는 문제는 증명의 대상이되는 논리절의 집합 S의 보조집합(Set—of—support)에 있는 논리절을 선정하는 것이 일반적으로 효율적이다. 그리고, 보조집합에 있는 논리절중에서 구체적으로 어떤 조건을 갖는 논리절을 선택하느냐 하는 문제는 앞으로 연구해야할 과제이다. 그러면, 최상논리절 Co가 선정되었다고 가정하자. 이 경우에 최상논리절 Co와 용해가능한 옆논리절은 여러개 있을수 있다.
Co와 가능한 옆논리절과의 용해된 논리절을 R1, ……, Rm이라 하자. 각 Ri, 1<=i<=m는, 증명으로 유도하는 가능한 중앙논리절들이다. Ri중에 공논리절이 있으면 증명이 끝나지만, 그렇지 않은 경우에는 각 Ri를 중앙 논리절로 해서 그와 용해가능한 옆논리절들을 찾아서 용해를 계속해나가고, 공논리절이 생성되면 증명은 끝난다. 이러한 과정을 간편하게 표현하기 위해서 그림 5와 같은 용해경로를 만들었다.

여기서, Co최상논리절이고, 각 Bo, ……, Bn −1는 옆논리절이고, 각 C1, ……, Cn −1은 중앙논리절이다.
용해경로를 이용하면 OL—유도는 OL—유도 트리(tree)를 형성하게 된다.
<예 5>OL—유도트리의 예
논리절들의 집합 S는 다음과 같다.
(1) P v Q
(2) ~P v Q
(3) P v −Q
(4) ~P v −Q
최상논리절 : P v Q
S에 대한 OL—유도 트리는 그림 6에 있다.



<그림 5>용해경로



<그림 6>OL—유도트리

여기서 P v Q v ~P 형태에서 P와 ~P는 ~P가 제일 뒤에 있고 서로 상방되는 짝이므로 제거할 수 있고, Q는 그 다음에 네모없는 리터럴이 없기 때문에 제거되어서 결국은 □(공논리절)이 된다. OL—유도의 2가지의 중요한 규칙은 첫째 제일뒤에 있는 네모없는 리터럴과 앞서 있는 서로 상반되는 짝이되는 네모있는 리터럴을 동시에 제거할 수 있고, 둘째 네모있는 리터럴 다음에 네모없는 경우에는 그 네모있는 리터럴을 제거할 수 있다.

여기서, OL—반박증명을 찾는 문제는 트리탐색문제로 생각할 수 있다. 최상논리절 Co서부터 시작하여 용해가능한 모든 옆 논리절들에 대해서 용해된 논리절들을 구하고, 각 용해된 논리절들을 중앙논리절로 하여서, 다시 용해 가능한 옆 논리절들에 대해서 용해된 논리절들을 차례대로 구해나가는 방식을 폭 우선(bread—first)방법이라고 한다. 폭 우선 방법은 많은 불필요한 논리절들을 생성할 가능성이 많기 때문에 보다 효율적인 트리탐색방법인 깊이 우선(depth—first)방법을 이용할 수 있다. 이 방법은 트리를 전개할 때 왼편에 있는 노드(node)에 대한 전개를 깊이의 한계까지 전개하고, 그 후에 그 오른편 노드에 대한 전개를 하는 것이다. 깊이 우선 방법은 더 개선할 수가 있는데 그것은 각 노드에서 생성하는 후계 노드(successor node)를 하나만 생성하도록 하는 것이다. 이것을 수정된 깊이 우선(modified depth—first)방법이라고 한다.

(수정된 깊이 우선 방법)

S : 반박증명의 대상이 되는 순서있는 논리절의 집합

<Step 1>최상논리절 Co의 옆 논리절이 될 수 있는 모든 논리절들을 S에서 찾는다. 만약 그런 옆 논리절이 존재하지 않으면 증명없이 종료한다. 그렇지 않으면, $B^1o, \cdots\cdots, B^ro$을 그런 용해가능한 옆논리절라고 하자. Clist는 다음과 같은 짝들의 리스트(list)로 한다. $(Co, B^1o), \cdots\cdots, (Co, B^ro)$

<Step 2>Clist가 공집합이면 증명없이 종료하고, 그렇지 않으면 계속.

<Step 3>(C, B)를 Clist에 있는 첫번째 짝이라 하자. (C, B)를 Clist로 부터 제거한다. 만약 C의 깊이가 깊이한도(depth bound)보다 크다면 step 2로 간다. 그렇지 않으면 계속.

<Step 4>C와 B를 용해해서 생긴 논리절들을 R1, ……, Rm이라 하자. 각 R*i, 1<=i<=m, 는 Ri를 축소(reduce)한 것으로 한다.

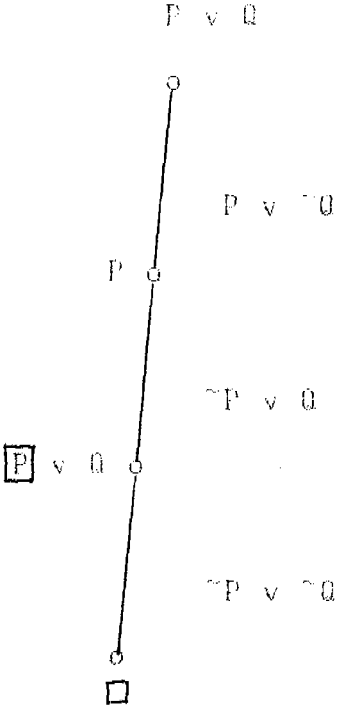
*여기서 축소는 OL—유도의 2가지 규칙을 적용해서 축소하는 것을 의미한다.

<Step 5>어떤 R*i, 1<=i<=m, 가 공논리절이면 반박증명은 성공적으로 종료되고, 그렇지 않으면 계속.

<Step 6>각 i, 1<=i<=m에, 대해서 R*i의 옆 논리절이 될 수 있는 논리절들을 S에서 찾는다. 그런 옆 논리절이 없는 R*i는 제거한다. 그렇지 않으면 이런 옆 논리절들을 Bi1, Bi2, ……, Bisi라 하자. 다음과 같은 짝들을 Clist의 앞부분에 놓고 step 2로 간다.

(R*i, Bi1), ……, (R*i, Bisi)

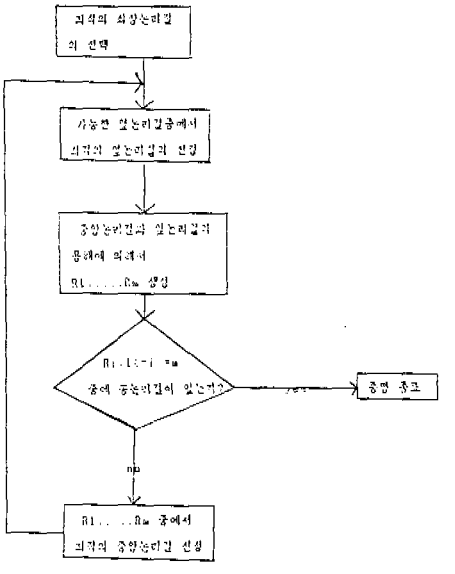
OL—유도트리의 예 5를 수정된 깊이우선방법으로 구현한 유도트리는 그림 7과 같다.



<그림 7>수정된 깊이우선방법의 유도트리

우리는 이런 수정된 깊이 우선방법을 효율적으로 구현할 새로운 전략(strategy)들을 개발하여야 한다.

이 논문에서 제안한 OL—유도의 개선방향은 수정된 깊이우선 방법을 사용하면서, 그림 8에 있는 흐름(flow)에 따른 개선방법을 찾도록 하는 것이다.



<그림 8>개선된 OL—유도의 흐름

여기서, 연구의 대상은 첫째 어떤 기준에서 최적의 최상논리절을 결정하는것, 둘째 어떤 기준에서 최적의 앞논리절을 결정하는것, 세째 어떤 기준에 의하여 최적의 중앙논리절을 결정하느냐 하는 것이다. 지금까지 제안된 전략된 전략들을 살펴보면, 삭제전략, 리터럴 갯수가 적은 논리절 우선전략, 논리절에 대한 평가함수를 써서 평가함수값이 적은 논리절을 우선 선택하는 전략등이 있다. 특히 평가함수를 이용하는 방법에서는 평가함수를 구하는데 많은 통계적 정보가 필요하고, 계산비용이 많이 들기 때문에, 논리절을 평가하는 보다 간편하고, 효율적인 방법을 강구하여야 한다. 이런 문제를 해결하기 위한 연구방향으로는 증명할 정리에 대한 의미적 정보를 증명절차에 이용하는 방법에 대해서 연구하는 것이다.

VII. 결 론

이 논문에서는 자동정리증명에 대한 일반적 개념이 소개됐고, 1계논리의 기초 개념이 서술되었다. 자동정리증명 방법의 역사적 발전과정을 서술했고, 최근의 자동정리증명 방법들을 Robinson의 융해 원리를 시발점으로 기술했다. 최근에 관심을 많이 받고 있는 선형 융해에 대해서 서술하고, 그것의 개량된 형태인 OL—융해에 대해서 중점적으로 기술했다.

한 정리를 예로 들고, 그것을 스콜렘 표준형으로 변환하는 과정을 기술하고, 컴퓨터에서 자동정리증명의 실행 예를 들었다. 연구방향으로는 OL—유도를 어떻게하면 효율적으로 구현하느냐에 초점을 맞추어 전개했다. 그 방향의 하나는 수정된 깊이 우선 방법을 효율적으로 구현하는 방법에 대한 것인데, 논리절의 평가함수를 간결하게 구하는 방법을 모색하는 데 있다. 그런 방법의 하나는 증명할 정리에 대한 의미적 정보를 이용해서 논리절의 평가함수를 간결하게 구하는 방법이다.

참 고 문 헌

1) R.S.Boyer, Locking:a Restriction of Resolution Ph, D Thesis, Univ. of Texas at Austin, Texas 1971.
2) C.L.Chang, The Unit Proof and the Input Proof in Theorem Proving, J.ACM, vol.17, No.4, pp.698~707, 1970.
3) M.Davis and H.Putnam, A Computing Procedure for Quantification Theory, J.ACM, vol.7, No.3 pp.201~215, 1960.
4) P.C.Gilmore, A Proof Method for Quantification Theory:its Justification and Realization, IBM J Res. Develop., pp.28~35, 1960.
5) J.Herbrand, Investgations in Proof Theory:the Properties of the Propositions, in "From Frege to Godel:a Source Book in Mathematical Logic", J. van Heijenoort edition, Harvard Univ. Press, Cambridge, Massachusetts, 1930.
6) D.W.Loveland, A Linear Format for Resolution, Proc. IRIA Symp. Automatic Demonstration, Springer-Verlag, New York, pp.147~162, 1970.
7) J.A.Robinson, A Machine Oriented Logic Based on the Resolution Principle, J.ACM, vol.12, No. 1, pp.23~41, 1965.
8) J.R.Slagle, Automatic Theorem Proving with Renamable and Semantic Resolution, J. ACM, vol.14, No.4, pp.687~697, 1967.
9) L.Wos, G.A.Robinson, and D.F.Carson. Efficiency and Completeness of the Set of Support Strategy in Therem Proving, J.ACM vol.12, No.4, pp.536~541, 1965.