교재 8,9장 논리의 추리

5주차: 지식 표현과 논리

학습목표:

지식 표현 언어로서 1 차 논리의 장점을 이해할 수 있다.

자연어 문장을 1차 논리 문장으로 변환 할 수 있다.

학습 내용:

1 차 논리의 개념을 복습한다.

* 구문
* 해석
* 의미론
* 한정 기호의 의미론
* 함의
* 단일화
* 스콜렘화

1차 논리 추론 규칙을 학습한다.

4.3.1 구문

1차 논리의 구문에 필요한 심볼이나 알파벳을 소개한다.

4.3.1.1 심볼

* 논리적 기호: AND, OR, NOT, ALL, EXISTS, IMPLIES, IFF, FALSE, =.
* 비논리적 기호: 나누기
* 상수:
  + 술어: 1-ary, 2-ary, …, n-ary. 이들은 통상적으로 식별자들 이다.
  + 0-ary, 1-ary, 2-ary, …, n-ary. 이들은 통상적으로 식별자들 이다. 0-ary 함수는 개별 상수라고 부르기도 한다.

술어가 참 또는 거짓을 리턴 하는 경우, 함수는 모든 값을 돌려줄 수 있다.

* 변수: 통상적으로 식별자

기호, 술어, 함수를 의미하는 대문자와 변수를 표시하는 소문자 등과 같이 적절한 규칙을 사용하여 술어, 함수 및 변수가 사용된 식별자들을 구별하는 능력이 필요하다.

4.3. 1 .2 항

항은 개별 상수(0-ary 함수), 또는 변수, 또는 n항에 적용된 F(t1 t2 …tn)과 같은 n-ary 함수 등이다.

여기서는 F(t1 t2 …tn)과 (F t1 t2 …tn)의 표기법이 사용된다.

4.3.1.3 원자식

원자식은 FALSE 또는 n 항 P(t1 t2 …tn)과 같은 n-ary 술어이다.

(t1=t2)에서 “=”는 논리적 기호이다. 단 여기서 t1과 t2는 항들이며 원자식이다.

4.3.1.4 리터럴

리터럴은 원자식(양의 리터럴)이거나 원자식의 부정(음의 리터럴)이다. 그라운드(기저) 리터럴은 변수가 없는 리터럴을 의미한다.

4.3.1.5 절

절은 리터럴의 분리이다. 기저절(ground clause)은 변수가 없는 절을 의미한다. 혼(horn) 절은 적어도 하나의 긍정적인 리터럴이 있는 절이다. 한정절(definite clause)은 정확히 하나의 긍정적인 리터럴을 갖는 혼 절이다.

아래의 함축은 혼절 또는 한정절과 동일합니다.

(A IMPLIES B) = ((NOT A) OR B)

(A AND B IMPLIES FALSE) = ((NOT A) OR (NOT B)).

4.3.1.6 공식

* 공식은 다음 중 하나이다.
* 원자식
* 부정, 예를 들면 NOT of formula
* 결합 공식, 예를 들면 AND of formula
* 분리공식, 예를 들면 OR of formula
* 함축, 예를 들면(formula1 IMPLIES formula2)의 형태,
* 등가, 예를 들면(formula1 IFF formula2)의 형태, IFF(if and only if)
* 범용 정량자(전칭 기호): (ALL variable formula: ∀ x P(x)는 「for all x」,
* 존재 정량자(존재 기호): (Exists variable formula: ∃ x p(x))로 표기된다.
* 공식에 변수가 포함되어 있지 않으면 free라 한다. 모든 변수들에 종속 되는 공식은 closed formula라 하고 모든 변수들로부터 자유로운 공식은 open formula라 한다.
* 절로 분리가 가능한 공식을 절 형식(Clausal form)이라 한다. 모든 공식은 절 형식과 등가임을 학습할 것이다.
* 용어들과 공식들을 통칭하여 Form 또는 표현(Expression)이라고 한다.

4.3.1.7 대체(치환)

* s항이 주어진 경우, 변수 x를 위하여 s를 t로 치환한(s[t/x]) 결과는 다음과 같다.
  + - * t, 만약 s가 변수 x일 경우
      * y, 만약 s의 변수 y가변수 x와 상관이 없는 경우
      * F(s1[t/x] s2[t/x] … sn[t/x]) 만약 s가 F(s1 s2 … sn)
* A 공식이 주어진 경우, 변수 x를 위하여 s를 t로 치환한(s[t/x]) 결과는 다음과 같다.
  + - * FALSE, 만약 A가 FALSE일 경우
      * P(t1[t/x] t2[t/x] … tn[t/x]), 만약 A가 P(t1 t2 …tn)일 경우
      * (B[t/x] AND C[t/x]), 만약 A가 (B AND C)이고 다른 결합(connectives)과 유사할 경우
      * (ALL x B), 만약 A가 (ALL x B)일 경우, (EXISTS 에서도 만족함)
      * (ALL y B[t/x]), 만약 A가 (ALL y B)이고 변수 y가 변수 x와 상관이 없는 경우(역시 EXISTS에서도 만족함)

치환 [t / x]는 용어에서 용어로, 그리고 수식에서 수식으로의 사상으로 볼 수 있다. t1 t2 …… tn은 항들이고 x1 x2 …… xn은 변수들인 [t1/x1 t2/ x2 …… tn / xn]을 정의하고 치환하면, x1은 t1으로, x2는 t2로, xn은 tn 으로 동시에 치환합니다. 주의할 점은 동시 치환은 순차 치환과 다르다는 점이다.

7.1 학습 목표

• 학생들은 제한된 클래스의 1 차 논리 문장으로 규칙 사용을 이해해야합니다.

• 학생은 Horn 절의 개념을 잘 알고 있어야합니다.

• 학생들은 다음 추론 알고리즘을 이해하고 구현할 수 있어야합니다.

o 순방향 체인

o 역방향 체인

• 학생들은 PROLOG 프로그래밍 언어의 본질과 그것에 사용 된 추론 방법을 이해해야합니다.

• 학생들은 기본 PROLOG 프로그램을 작성할 수 있어야합니다.

• 학생들은 전문가 시스템 설계와 관련된 아키텍처 및 문제를 이해해야합니다.

7.1 Instructional Objective

• The students should understand the use of rules as a restricted class of first order logic statements

• The student should be familiar with the concept of Horn clause

• Students should be able to understand and implement the following reasoning algorithm

o Forward chaining

o Backward chaining

• Students should understand the nature of the PROLOG programming language and the reasoning method used in it.

• Students should be able to write elementary PROLOG programs

• Students should understand the architecture and issues involved in designing an expert system

이 단원이 끝나면 학생은 다음을 할 수 있습니다.

At the end of this lesson the student should be able to do the following:

• 가능하다면 지식 기반을 일련의 규칙으로 표현하십시오.

• 순방향 / 역방향 연결 알고리즘을 적합하게 적용합니다.

• PROLOG에 기본 프로그램 작성

• 설계 전문가 시스템

5.1. 도출 원리와 논리 프로그램

* 주어진 일차 술어 논리식이 항진식인지 판단할 때 일반적으로 일차 술어 논리식 P(x) 에 대하여 대상영역 D가 무한 집합일 때 ∀ x P(x)가 참인지 아닌지를 결정할 수 없다. 그러나 ∀ x P(x)가 거짓이라는 것을 증명하기 위해서는 어떤 a∈ D에 대해 P(a) 가 거짓임을 나타내면 되므로 항진식인지 판단하기 위하여 모든 x에 대하여 P(x) 가 참이 되는 증명을 하는 것보다 매우 간단하다.
* 어떤 술어 논리식 Q(x) 가 있다고 하자. Q(x) 가 항진식임을 증명하기 위해서는 위에 기술한 것을 고려하여 Q(x) 의 부정 ┓Q(x) 가 항위식임을 증명하면 된다. 여기서 ┓Q(x) 三 P(x) 이고, P(x)를 거짓으로 하는 어떤 a∈ D가 존재하는 술어 논리식 P(x)를 발견하여 ∀ x P(x)가 항위식임을 나타낸다. 이로 인해 Q(x) 가 항진식임을 증명한다. 즉 일차 술어 논리식의 항진성을 증명하는 데에는 대상인 논리식의 부정형을 만들어 그것이 항위식임을 증명함으로써 대상인 논리식이 항진식임을 증명하면 된다. 이 항위식을 전칭기호 ∀만을 정량 기호로 가지는 술어 논리식으로 표현하여 대상 영역의 어떤 값에 의해 술어 논리식이 거짓이 됨을 나타낸다.
* 이와 같이 주어진 술어 논리식이 항진식인지 어떤지를 판정하기 위하여 논리 식의 부정을 취한 술어 논리식이 항위식임을 나타내는 방법, 즉 주어진 술어 논리식이 항진식인지 아닌지를 판정하는 방법을 도출 원리 (resolution principle) 라고 한다.

5.1.1 스콜렘 표준형

* 항진성을 증명하는 대상이 되는 술어 논리식은 일반적으로 그것의 부정형인 항위식에 전칭기호만으로 이루어지는 경우는 없고， 전칭 기호(∀)와 존재 기호(∃)가 포함 된다.
* 극단적인 예지만 다음과 같이 단 하나의 존재 기호(∃)를 포함하는 술어 논리식을 생각해보자.

∀x1, … xn ∃ y R(x1, … xn, y)

이 논리식은 x1, … xn 의 어떠한 값에 대해서도 어떤 값 y= a가 존재하여, R(x1, … xn, y)이 참이 되도록 x1, … xn 가 취하는 값에 의존하여 y값이 결정 되는 것을 의미한다. 이것으로부터 y와 x1, … xn 의 의존관계를 y=f(x1, …, xn)로 나타내기로 한다. 이 때 위에 기술한 논리식은 진위성은 보존되면서 아래와 같이 고쳐 쓸 수 있다.

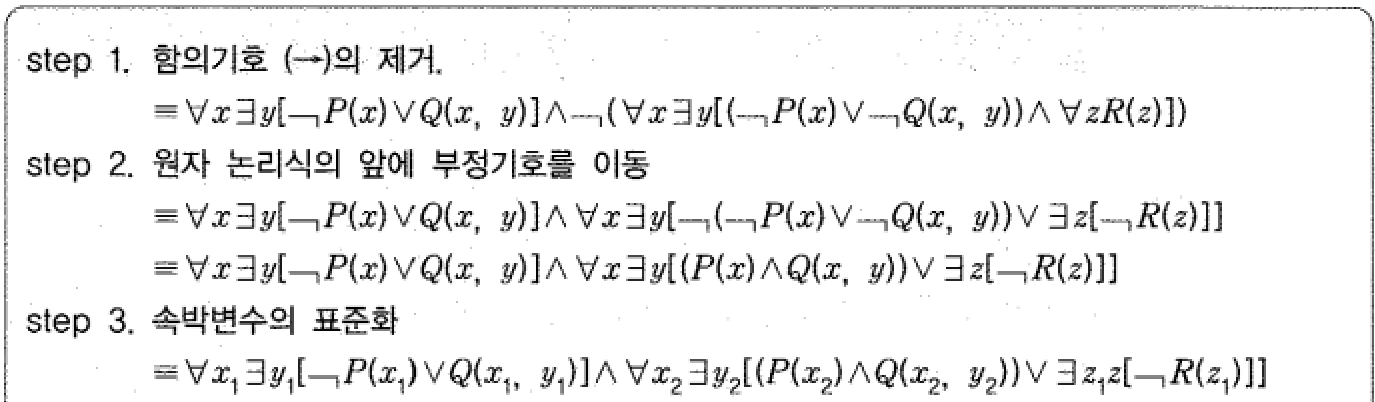
∀x1, … xn R(x1, … xn, f(x1, … xn,))

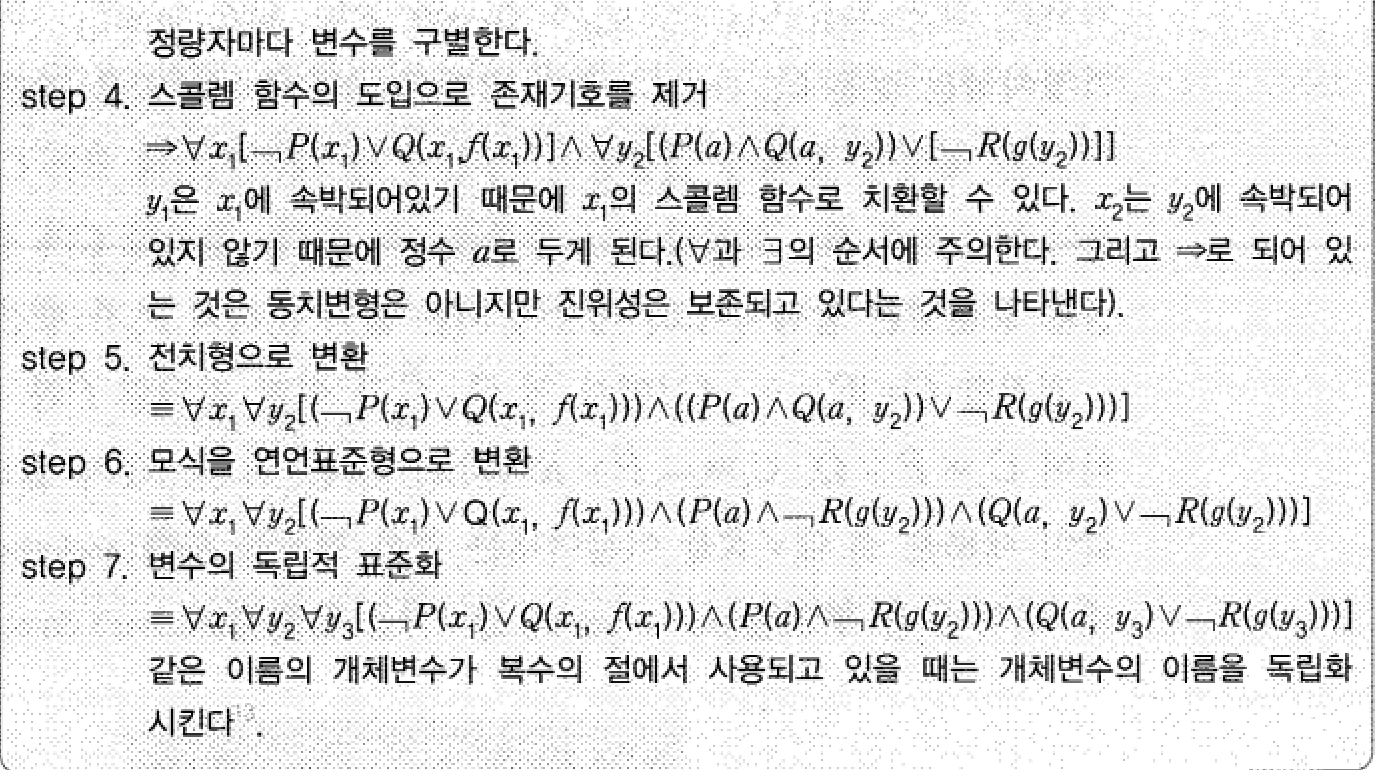
이와 같은 함수 f 를 스콜렘 함수 (Skolem function)이라고 한다. 위에 기술한 것처럼 변형한 술어 논리식은 원래 식과 동치는 아니지만 진위성은 보존되고 있다. 술어 논리식에 대하여 위에 기술한 스콜렘 함수를 이용함으로써 존재기호,∃를 제외한 절형식을 스콜렘 표준형 (Skolem standard form)이라고 한다.

5.1.1.1 스콜렘 표준형으로 변환

술어 논리식의 스콜렘 표준형으로의 변환 방법을 나타낸다. 예로써 다음 술어 논리식을 스콜렘 표준형으로 변환한다.

∀x∃y[P(x)→Q(x, y)]∧┓(∀x∃y[(P(x)→┓Q(x, y))∧∀zR(z)])





단계 7에서 ∀x(P(x) ∧Q(x))≡ ∀xP(x) ∧∀xQ(x) ≡∀x1P(x1) ∧∀x2Q(x2)를 이용 ≡∀x1∀x2(P(x1) ∧ Q(x2))

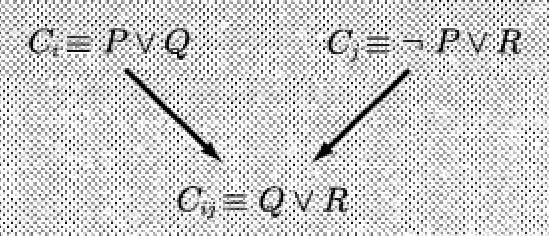
5.1.2 도출 원리에 에 의한 증명

* 도출 원리에 의해 일차 술어 논리식의 항진성을 증명한다. 술어 논리식 P0의 부정형 ┓P0가 아래와 같이 스콜렘 표준형으로 표현되어 있다고 하자.

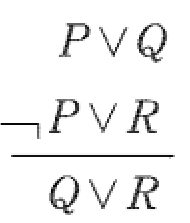
┓P0 = ∀ x1, … ,∀xn[C1∧…∧Cn]

스콜렘 표준형 ∀ x1, … ,∀xn[C1∧…∧Cn]에서 모든 개체변수는 전칭 기호로 묶여 있기 때문에 모식을 구성하고 있는 절로 부터 이루어진 집합 { C1,…, Cn}에 주목한다. 이 집합을 절집합(clause set) 이라고 한다. 절집합C={C1,…, Cn}이 참 이라는 것은 모든 C1,…, Cn 이 동시에 참이 됨을 말한다. 그리고 절집합 C가 거짓이라는 것은 C1,…, Cn 중에 적어도 하나가 거짓임을 말한다. 모든 해석에 대해서 절집합이 거짓이 될 때(다시 말해 모든 C1,…, Cn이 동시에 참이 되지 않을 때 ) , 절집합 C는 충족 불능이 된다. 절집합 C가 충족 불능이라면 스콜렘 표준형은 항위식이고, 대상이 되는 술어 논리식 P0가 항진식임이 증명된다.

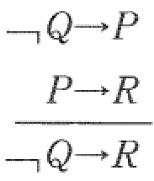
* 절집합 C= { C1,…, Cn }씨에서 절 Ci , Cj 가 임의의 리터럴 P와 부정 ┓P를 포함 하여 Cj≡P∨Q， C i ≡ ┓P∨R이라고 하자. 이 두 개의 절로부터 아래와 같이 새로운 절 Q∨R을 이끌어내는 추론 형식을 도출 (resolution) 이라고 한다. 이때, Ci와 Cj를 부모절 (parent clause), Q∨R을 도출절(resolvent clause) 이라고 한다.



* 도출은 타당한 추론 형식이며 아래와 같이 쓸 수 있다.

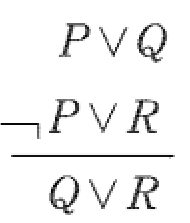


여기서 P∨Q≡ ┓P → Q ≡ ┓Q → P, ┓P∨R ≡ P → R로부터 아래와 같이 변형할 수 있다. ┓P∨R



┓Q → R ≡Q∨R로부터 Q∨R은 P∨Q, ┓P∨R의 삼단논법에 의해 구할 수 있는 논리적 귀결이고 도출은 건전한 추론임을 알 수 있다.

또한 타당한 추론형식인 좌측 식은 삼단논법은 (A → B) ∧ (B →C) →(A → C) = T로 나타낼 수 있으므로 우측과 같이 나타낼 수 있다.



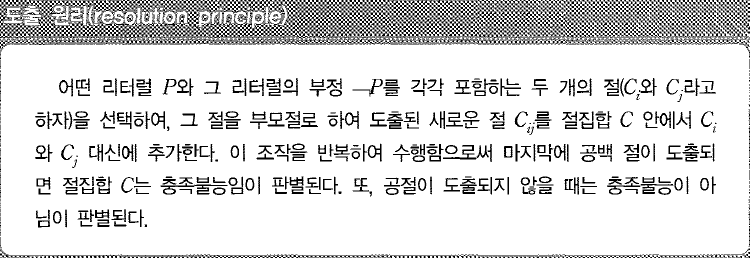
(P ∨ Q) ∧ (┓P ∨ R) → Q∨R ≡ T

위 식은 항진식이므로 그것의 부정은 항위식이 되어 아래와 같이 변형될 수 있다.

┓ ((P ∨ Q) ∧ (┓P ∨ R) → Q∨R ≡(P∨Q) ∧(┓P ∨ R) ∧┓(Q∨R)≡ F

위 식은 삼단논법에서 전제와 결론의 부정과의 논리곱에 의해 구성된 논리식은 항위식이 되는 것도 동시에 나타내고 있다.

절집합에서 도출을 수행한 후 도출절을 절집합에 추가하는 것을 반복함으로써 최종적으로 공백 절(empty clause)을 이끌어낼 수 있다면 그것은 절집합이 충족불능임을 의미하여 주어진 술어 논리식의 항진성을 증명할 수 있다. 여기서 공백 절이란 0개의 리터럴로 이루어지는 절을 가리키며 口의 기호를 이용하여 표시한다.



도출 원리를 이용하여 어떤 논리식이 전제가 되는 복수의 논리식으로부터의 논리적 귀결이 됨을 증명할 수 있다. 예를 들어 논리식 P → R이 두 개의 논리식

1. P→Q
2. Q → R

로부터의 논리적 귀결임을 도출 원리를 이용하여 증명한다.

┓ ((P ∨ Q) ∧ (┓P ∨ R) → Q∨R ≡(P∨Q) ∧(┓P ∨ R) ∧┓(Q∨R)≡ F

위 식에 의해 논리식 P → R이 두 개의 논리식으로부터의 논리적 귀결임을 나타내려면 아래의 식이 충족불능임을 나타내면 된다.

(P→Q) ∧(Q→R) ∧┓(P→R)

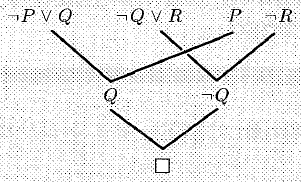
이 논리식의 절집합을 구하여 충족불능임을 도출 원리를 이용하여 나타낸다. 이 논리식을 절 형식으로 변환하면 아래와 같이 된다.

(┓P ∨ Q) ∧(┓Q∨R) ∧P∧┓R

이로부터 절집합은 아래와 같이 된다.

{┓P∨Q, ┓Q∨R, P, ┓R}

이 절집합에 도출 원리를 적용하면 아래 그림에 나타낸 것과 같은 도출 과정을 따라서 공백절이 도출된다. 이것에 의해 이 절집합은 충족불능임이 증명된다. 그림에 보여지는 그래프를 도출 그래프(resolution graph) 라고 하며, 특히 공백절을 가진 도출 그래프를 도출 트리 (resolution tree) 라고 한다.



5.1.3 단일화 치환

앞에서 나타낸 그림과 같이 명제 논리식에 대한 도출을 수행하는 것은 쉽지만, 술어 논리식에 대한 도출은 술어 논리식이 포함하는 개체변수에 새롭게 적절한 처리를 하지 않으면 도출을 수행할 수 없다.

절집합 C의 요소로서 아래의 절이 포함되어 있다고 하자.

Ci≡P(x, f(y))∨Q

Cj≡┓P(a, z))∨R

단, a는 개체정수, x, y, z는 개체변수, f는 함수이며, 개제변수는 전칭기호로 속박되어 있어서 임의의 값을 가질 수 있다고 하자. 이 때, x=a, z=f(y)임을 가정하면, 절 Ci와 Cj로부터 새로운 절을 도출하는 것이 가능하게 된다. 이와 같이 개체변수를 별개의 항으로 치환하는 조작을 단일화(unification) 라고 한다.

예를 들어 치환 s를 다음과 같이 표현하자.

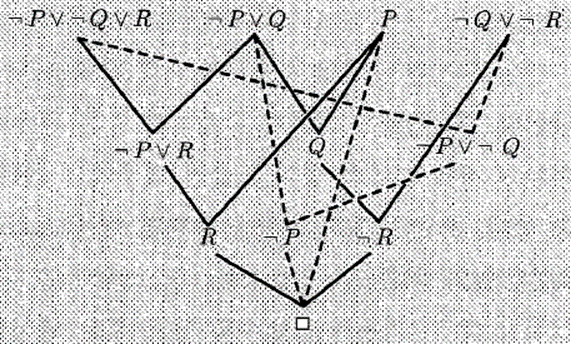
s= (t1/x1, t2/x2, …, tn/xn)

여기서 xi는 i번째 개체변수를 나타내며 t1은 그 개제변수를 치환하는 항을 가리킨다.

단일화 치환은 한번 시행하면 이후의 도출 과정에서의 개체변수는 그 항에 속박된다. 또 단일화 치환이 존재하지 않는 경우에는 도출이 불가능하다. 단일화 치환 s 하에서 공백 절이 도출될 경우에는 “단일화 치환 s 하에서 증명되었다”고 한다. 또한 단일화 치환을 수행할 때 필요최소한의 치환을 수행하는 것을 최일반단일화 치환(mgu: most general unifier) 라고 한다. 이것은 필요 이상의 치환을 수행하지 않고, 가능하면 변수로 남겨 두는 단일화를 말한다. 이로 인해 이후 치환의 자유도를 확보할 수 있게 된다.

5.1.4 도출 제어 전략

도출 원리에 기반하여 증명을 수행할 때 문제가 되는 것은 부모절의 특정한 두 개의 절을 구체적으로 어떻게 선택하면 좋을지 결정할 수 없다는 것이다. 아래 그림은 도출과정에서 서로 다른 증명 예를 나타낸다.



효율이 좋은 도출을 수행할 경우에는 절집합 내의 적절한 두 개의 절을 선택하기 위한 제어 전략이 필요하게 된다. 도출을 위한 전략은 크게 나누어 논리식의 의미를 고려하지 않는 기계적인 제어 전략과 논리식의 의미를 고려한 의미적인 제어전략의 두 가지가 있다. 아래에서 각각에 대하여 설명한다.

5.1.4.1기계적인 제어 전략

* 너비 우선 전략(breadth-first strategy)

도출대상이 되는 절집합에 있는

모든 절에 대하여 도출을 수행하는 전략. 도출된 절이 새롭게 도출대상이 되는 절집합에 포함된 경우에는 아직 도출에 사용되지 않은 절을 우선적으로 도출을 수행해간다. 이 전략은 도출대상이 되는 절을 너비 우선 탐색과 같은 방침으로 처리를 수행해간다.

* 성형도출(linear resolution)

도출대상이 되는 절집합으로부터 도출에 의해 얻어진 절을 하나의 부모절로 하여 도출을 반복하여 진행해 가는 전략. 이 전략은 도출대상이 되는 절을 깊이우선탐색과 같은 방침으로 처리를 수행해간다.

5.1.4.2 의미적인 제어 전략

* 지지집합 전략(set-of-support strategy)

도출대상이 되는 절집합 C는 충족가능한 부분 H와 불명확한 부분 S(지지집합(set of support)으로 부른다)의 두 부분으로 나눌 수 있음을 이용하여 무의미한 부모절의 조합을 방지하는 전략.

절집합 C는 아래에 나타내는 일반적인 형식의 논리식으로부터 구축된다.

(P1∧P2∧ … ∧Pn) →Q

위의 논리식에서 조건부 P1∧P2∧…∧Pn으로부터 얻어지는 절집합(H에 해당)은 통상적으로 충족가능이고, ┓Q에 대한 스콜렘 표준화에 의해 얻어지는 절집합(S에 해당)과 조합함으로써 무의미한 부모절의 조합을 방지한다.

* 의미도출(semantic resolution)

절집합 C에 대하여 어떤 해석 I를 부여하고, 이 해석에 기반하여 C 내의 절을 참이 되는 집합 CT와 거짓이 되는 집합 CF로 나눠서 도출에서의 부모절을 각각 하나씩 선택하는 전략.

5.1.5 도출 원리에 의한 문제 해결

도출은 주어진 논리식이 참이 되는가를 증명하는 추론의 하나이기 때문에 문제를 논리식으로 표현할 수 있으면 문제 해결에 도출 원리를 이용할 수 있다.

문제 해결에 도출 원리를 이용하는 것은 전제가 되는 지식이 논리식 P1∧P2∧ … ∧Pn으로 표현되었을 때， 그로부터 논리식 Q가 성립하는지를 증명함으로써 실현된다. 즉，

P1∧P2∧ … ∧Pn→Q

가 항진식임을 증명하는 것과 같으며 이것은 바로 도출 원리를 이용하여 수행할 수 있다.

예) R(x, y)는 “x는 y의 R이다”라는 관계를 나타내는 술어 논리식이라고 하고, 다음과 같은 지식을 가지고 있다고 하자.

(a) 자식(영자， 영이)

(b) 자식(영이， 순이)

(c) ∀ x∀ y∀ z[자식(x, y) ∧자식(y, z) →손(x, z)]

이 때， “순이의 손자는 누구인가?" 라는 다음의 질문이 주어졌다고 하자.

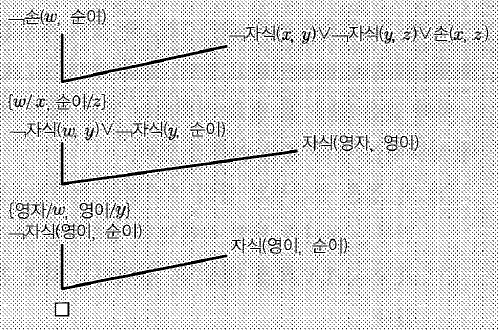
(d) ∃w손(w, 순이)

여기서 문제는 지식 (a), (b), (c)로부터 (d) 가 논리적 귀결로 얻어질 수 있는가 라는 것으로, 논리식 (a)∧(b)∧(c)∧ ┓(d)≡F 와 같은 형태로 표현한다. (a), (b), (c), (d)에 실제 논리식을 넣어서 절집합으로 표현하면 다음과 같이 된다.

{자식(영자, 영이), 자식(영이, 순이), ┓자식 (x, y) ∨┓자식 (y, z) ∨손(x, z),

┓손(W, 순이)}

이 절집합에 도출 원리를 적용함으로써 아래 그림과 같은 도출 트리가 얻어진다. W가 “영자”로 치환됨으로써 “순이”의 손자는 “영자”임을 알 수 있다.



앞에서 문제 해결은 초기상태로부터 목표상태에 이르는 경로를 탐색하는 것이라고 설명하였다. 그와 마찬가지로 논리에 의한 문제 해결은 도출에서 공백 절을 이끌어내는 경로를 탐색하는 것이 된다. 또한 경로의 탐색에서 전략으로는 앞 절, 도출 제어 전략에서 본 것과 같은 전략이 이용된다.

5.2.1 혼절과 추론

도출 원리에 의한 증명의 효율은 도출을 위한 전략에 의존하고 있다. 그러나 대상이 되는 부모절을 보다 간단하게 선택할 수 있다면 도출의 효율은 향상된다. 이것을 의도하여 형식으로 표현한 절을 혼(Hom) 절이 라고 한다. 혼 절이 어떻게 정의되는가에 대하여 설명한다.

* 혼절의 정의

절은 원자 논리식의 논리합으로 구성되는 논리식이고 원자 논리식은 참의 리터럴 혹은 거짓의 리터럴로부터 구성된다. 교환법칙을 이용하여 거짓의 리터럴을 왼쪽에, 참의 리터럴을 오른쪽으로 몰아서 표현하면 논리식은 다음과 같이 된다.

┓A1 ∨ ┓A 2 ∨…∨ ┓An∨B1∨B2∨---∨Bm

이 때 위 식에서 m의 값이 1 인 즉 참의 리터럴을 하나밖에 포함하지 않는 절을 혼 절 (Horn clause) 이라고 한다. 예를 들어, ┓A1∨┓A 2∨B1는 혼 절이다.

* 이와 같은 절을 혼 절로 정의하는 것은 매우 자의적이지만, 그 배경에는 아래와 같은 이유가 있다.

1. 혼 절은 전건부가 참의 리터럴의 논리곱으로 결합된 형태로 나타내고, 후건부는 하나의 참의 리터럴로 나타내는 함의의 형태로 표현할 수 있다. ┓A1∨┓A2∨B1를 혼 절이라고 하면 혼 절은 다음과 같이 고쳐 쓸 수 있다.

A1∧A2→B1

이 형식으로 표현된 지식은 직관적으로 이해하기 쉽고, 또 기술하기도 용이하다.

함의를 사용하여 혼 절을 표현할 때는 기존 표기법과는 함의의 방향을 역으로 해서 다음과 같이 전건부와 후건부의 위치를 바꾼 표기법을 이용한다.

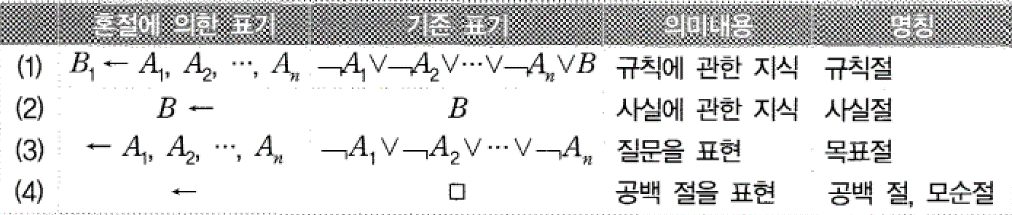
B1←A1∧A2

여기서 B1을 머리부(헤드, head) 라고 하고, A1∧A2를 본체(바디, body) 라고 한다.

1. 혼 절을 이용한 추론은 전방향 추론 혹은 역방향 추론을 통하여 이루어지며, 이들의 추론 과정은 매우 명확하여 사람이 이해하기 쉽다.
2. 혼 절을 이용함으로써 효율이 좋은 선형 도출을 수행할 수 있다.

* 혼 절의 표현 형식

혼 절은 위 정의로부터 네 가지 기본적인 형식 중 하나로 표현된다. 네 가지 기본적인 혼절의 표기와 일반적인 절의 표기의 대응을 표에 나타냈다. 그리고 혼 절로 이루어지는 집합을 혼 집합(Horn set) 이라고 한다.



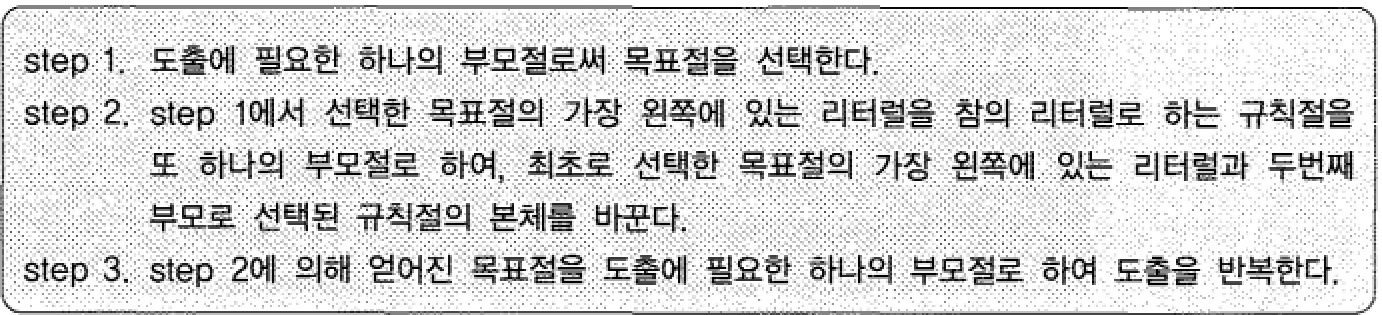
┓ ((P ∨ Q) ∧ (┓P ∨ R) → Q∨R ≡(P∨Q) ∧(┓P ∨ R) ∧┓(Q∨R)≡ F

5.2.2 혼 절에 의한 도출과 논리 프로그래밍

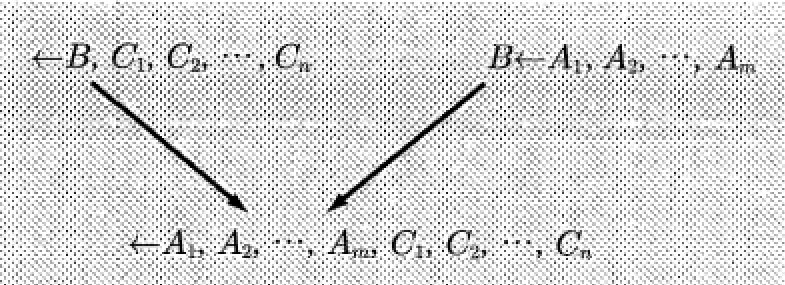
5.2.2.1 SLD 도출

혼 집합의 충족불능성을 효율적으로 증명하는 도출 전략으로써 SLD 도출(selective linear definite resolution) 방법 이 있다.

* SLD 도출 알고리즘



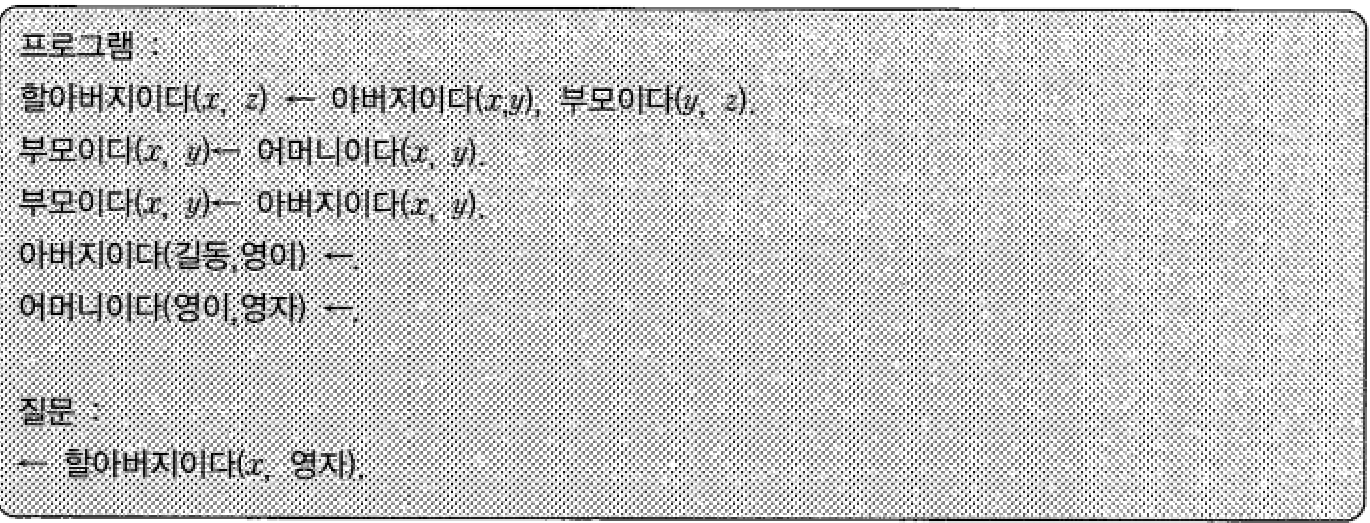
알고리즘을 그림으로 표현하면 아래와 같다. SLD 도출에서 부모절의 한쪽은 목표절이며 그것은 질문을 나타내고 있다. 그리고 나머지 한쪽의 부모절은 규칙절 또는 사실절이며 그것은 지식을 나타내고 있다. 즉， 주어진 질문과 사전에 준비된 지식으로부터 충족불능이 도출된다면 주어진 질문은 그 지식을 이용하여 회답이 가능하다고 판단할 수 있다.

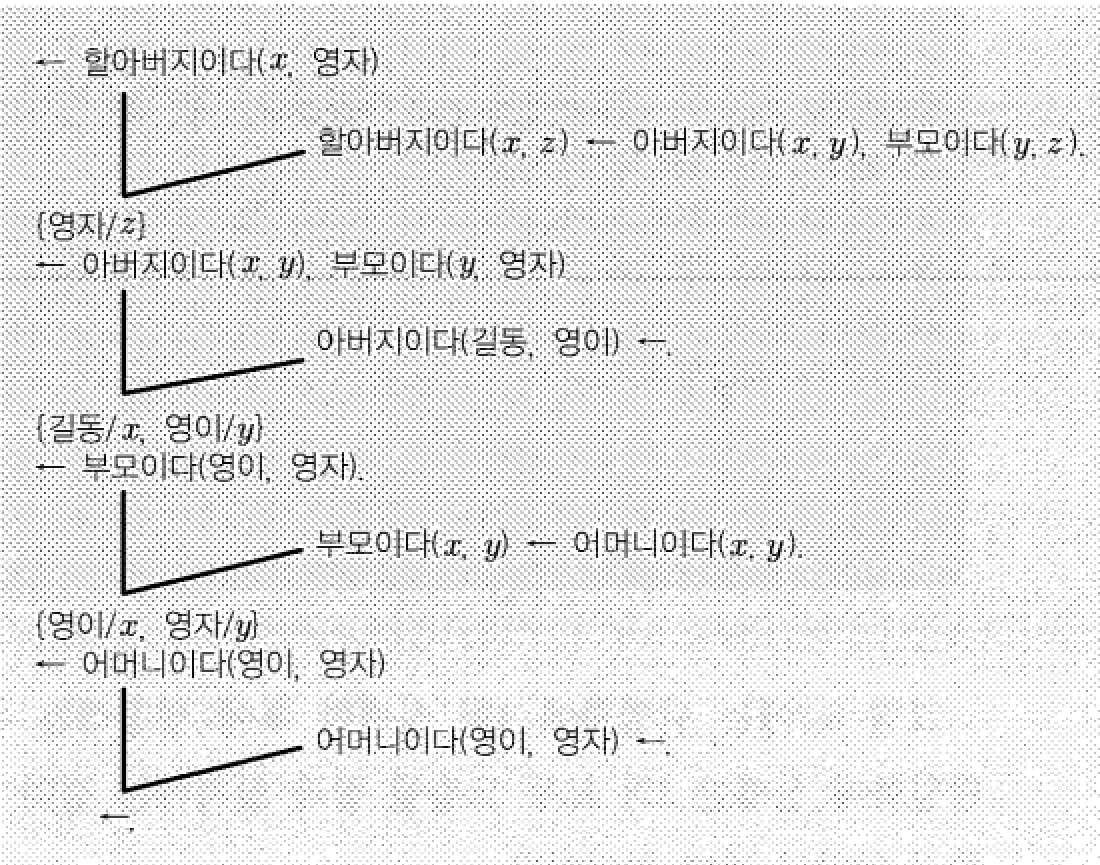


5.2.2.2 논리 프로그래밍

Prolog(programming in Logic)는 혼 절에 의한 프로그램의 기술과 SLD 도출의 메커니즘을 처리 시스템으로 가지고 있는 논리형 프로그래밍 언어이다.

Prolog 프로그램이 다음과 같이 주어졌으면 그 실행 프로세스를 그림에 보였다.



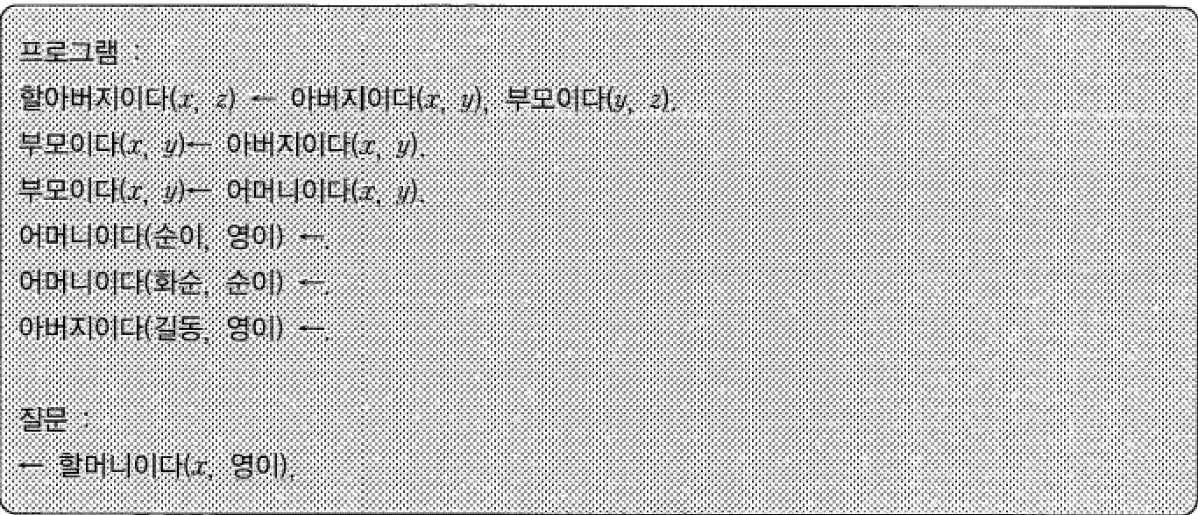


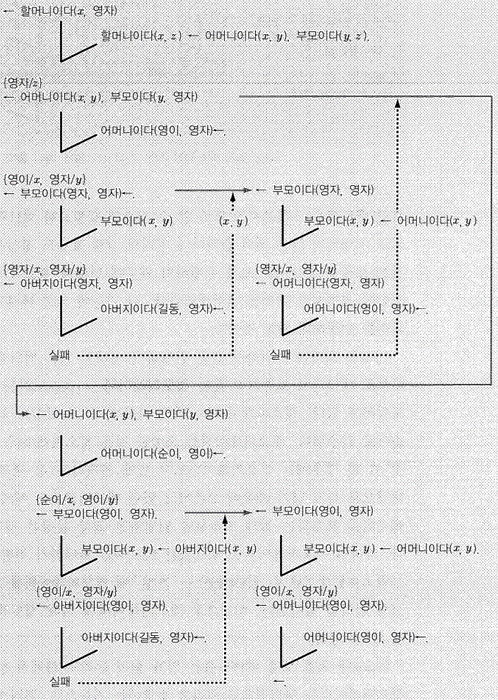
Prolog는 같은 머리부를 가진 절이 프로그램 중에 기술되어 있는 경우에 먼저 기술된 절을 우선적으로 처리한다. 따라서 “부모이다 (x, y)" 를 머리부로 가진 절에 대해서는 위 그림 에 나타낸 것처럼 “부모이다 (x, y)←어머니이다 (x, y)" 가 “부모이다(x, y)←아버지이다 (x, y)" 보다 우선적으로 처리되고 있다.

* 백트랙(backtrack)

Prolog에서는 도출 과정에서 단일화가 실패하면 실패하기 직전의 절까지 되돌아와서 다른 부모 후보가 되는 절과 도출을 수행하여 처리를 계속하는 메커니즘을 가지고 있다. 이것을 백트랙(backtrack) 이라고 하며 Prolog의 특정 중 하나로 되어 있다. 백트랙은 단일화를 수행할 때 실제로 선택되는 절에 대해 다른 후보가 되는 절을 메모리에 보존하고 있으면서 선택된 절의 이후 처리에서 단일화에 실패했을 때 보존하고 있던 다른 후보로의 가능성을 조사하는 기능이다. 이것은 도출을 수행하는 처리가 종형 탐색으로 수행되고 있음을 나타내며, 백트랙은 탐색 문제와 마찬가지로 단일화가 성공하는 경로를 발견하는 행위로 간주할 수 있다.

아래 프로그램을 실행시킬 때의 처리과정을 아래 그림에 보였다.





그림에는 단일화에 3회 실패한 후에 성공하는 처리과정이 나타나 있다. 단일화에 실패한 2회 모두 도출될 절과 일치하는 절이 지식(프로그램) 안에 존재 하지 않았던 것이 원인이다.

단일화 실패에 의한 백트랙은 다음의 경우에 일어난다.

1. 단일화를 수행할 때 머리부가 일치하는 절이 존재하지 않는 경우.
2. 단일화를 수행한 머리부의 본체 절이 단일화에 실패한 경우.
3. 도출된 절이 거짓인 경우.
4. 한번 단일화에 성공했지만 이후의 도출 과정에 의해 백트랙이 일어나서 부모절의 대체 후보가 그 절에 존재하는 경우.

* 백트랙의 오류

백트랙은 혼 집합으로 표현된 프로그램으로부터 해를 모두 탐색하는데 매우 유용한 가능이지만, 다음과 같은 프로그램이 있을 경우 잘못된 해를 발견하게 된다. 그림에 처리과정을 보였다.

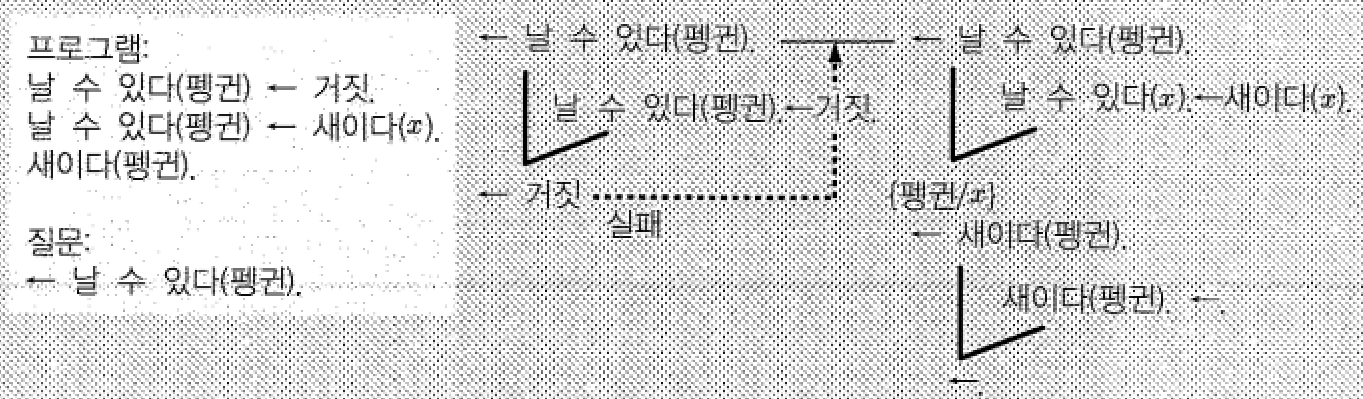
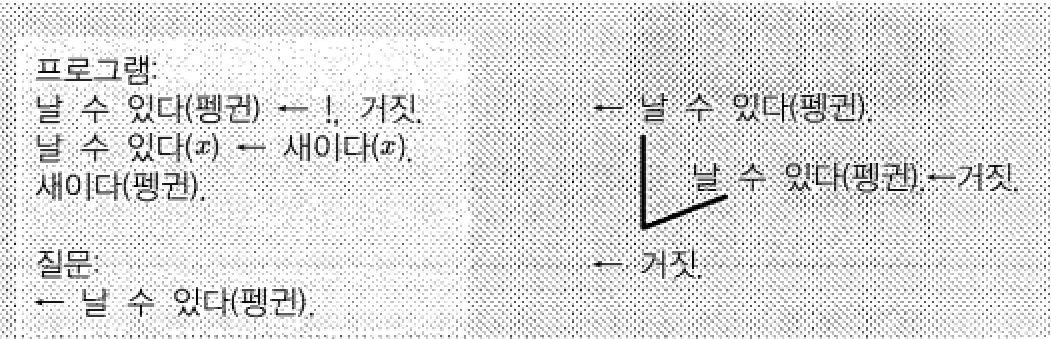


그림 을 보면 펭귄이 날 수 있다는 것은 한번 “거짓”으로 판별되고 있는데도 불구하고 절이 거짓이 된 경우에는 백트랙이 일어나기 때문에 최종적으로 펭귄도 날 수 있다는 해를 출력하게 된다. 이와 같은 결과가 발생하는 이유는 본체가 다른 머리부에 “날 수 있다”를 가진 절을 백트랙에 의해 적용해버리기 때문이다. 여기서 만약 머리부에 “날 수 있다”를 가진 절을 한번 단일화에 실패하면 백트랙을 수행하지 않도록 할 수 있으면 불필요한 백트랙을 수행하지 않게 되고 기대하지 않은 해의 출력이나 처리의 실행 효율도 향상시킬 수 있다. 백트랙은 도출을 종형 탐색으로 수행하기 때문에 단일화 후보가 되는 절을 모두 메모리에 기억시켜 처리를 진행해버린다. 이 기능에 의한 폐해가 그림과 같은 처리를 수행하게 되는 것이다.

* 프롤로그의 컷오퍼레이터(cut operator)

Prolog에서 백트랙의 단점을 보완하기 위하여 컷오퍼레이터 (cut operator) 라는 기능을 추가하였으며 이는 한번 단일화에 실패한 절은 백트랙을 수행하기 위한 후보로 메모리에 보존하지 않는 기능이다. 컷오퍼레이터는 “!" 로 나타내며 그 자신도 인수를 가지지 않는 술어로 간주된다. 컷오퍼레이터를 포함한 절은 컷오퍼레이터 이후의 술어가 “거짓”이 된 경우에는 컷오퍼레이터보다 앞에 존재하는 절 내의 술어는 백트랙의 대상으로 하지 않기 때문에 보존하고 있던 메모리로부터 제거한다. 또한 컷오퍼레이터를 포함하는 절의 머리부와 단일화한 질문 문장은 강제적으로 실패 시킨다. 따라서 컷오퍼레이터는 필요 없는 백트랙을 회피하기 위한 제어를 수행한다.

아래 그림의 절 “날 수 있다(펭귄)←거짓”에 컷오퍼레이터를 도입한 프로그램과 실행 결과를 나타낸다. 이것으로 컷오퍼레이터에 의해 백트랙이 제어되고 있음을 알 수 있다.

Prolog에 의한 도출 메커니즘은 일차 술어 논리의 틀에서 실현되고 있었지만, 컷오퍼레이터는 자의적으로 처리를 변경하는 기능으로, 휴리스틱 지식이 도입되어 있다. 이것은 Prolog가 일차 술어 논리의 틀을 넘어서 프로그래밍 언어로서의 처리 효율화 기능을 가지고 있다고 말할 수 있다.