상향식 추상 해석을 이용한 논리 프로그램의 AND/OR 병렬 수행 모델 개선

(Improving AND/OR Parallel Execution Model of Logic Programs using Bottom-up Abstract Interpretation)

창 병 모† 최 광 무†† 한 태 숙††
(Byeong-Mo Chang) (Kwang-Moo Choe) (Taisook Han)

요 약 본 논문에서는 상향식 추상 해석(bottom-up abstract interpretation)을 이용하여 논리 프로그램의 병렬 수행 모델인 AND/OR 프로세스 모델을 개선한다. 이를 위하여 기존의 상향식 추상 해석을 깊이 k 추상화를 기초로하여 프로그램의 각 절의 성공 패턴(success pattern)에 대한 근사값을 구할 수 있도록 확장하였다. 확장된 추상 해석은 각 절의 성공 패턴에 대한 근사값를 구할 수 있으며 이를 이용하면 어떠한 subgoal은 실행중에 실패할 것이라는 사실을 컴파일 시간에 결정할 수 있다. 본 연구에서는 이러한 성질을 이용하여 AND/OR 프로세스 모델을 개선하며 이 모델에서 AND 프로세스는 자식 OR 프로세스가 실패할 것이 확실한 경우에 자식 OR 프로세스를 생성하지 않는다. 이 방법론은 다른 하향식 (병렬) 수행에도 쉽게 적용 가능하다.

ABSTRACT This paper improves the AND/OR Process Model, a parallel execution model of logic programs, using a bottom-up abstract interpretation. We first extend a bottom-up abstract interpretation framework so that it can approximate the program's success patterns of program clauses by depth k abstraction. By the ability of the extended framework to approximate possibly non-ground success patterns for clauses, it can be statically determined whether some subgoals will fail during execution. We improve the AND/OR Process Model using this property, in which AND processes do not create child OR processes if it is statically determined that they will fail during execution. This approach can also be easily applied to other top-down (parallel) execution models

1. 서 론

논리 프로그램 언어는 그 언어의 논리성 및 간결성등으로 인하여 심볼 계산 분야에서 주요한 언어로 이용되고 있다. 또한 논리 언어는 자체의 비결정성(nondeterminism)으로 인해서 내재된 병렬성을 갖는다. 따라서 논리 언어는 프로그램이 가지고 있는 병렬성을 자연스럽게 표현할 수 있으며 이를 병렬 컴퓨터에서 효과적으로 구현할 수 있다는 측면에서 많은 관심을 받고 있다. 논리 언어의 내재된 병렬성을 이용하기 위해서 여러 병렬 수행 방법들이 제안되었으며 [2, 5, 6, 8, 12] 대표적인 모델로써 Conery에 의해서 제안된 AND/OR 프로세스 모델 [5, 6]을 들 수 있다. AND/OR 프로세스 모델은 논리 프로그램에 내재된 두 종류의 병렬성 즉 AND 병렬성과 OR 병렬성을 추구하고 있다. 어떤 절을 위한 AND 프로세스는 절 내의 각각의 몸체(body) 애톰에 대해서 OR 프로세스를 생성함으로써 AND-병렬성을 추구하고, 어떤 애톰을 위한 OR 프로세스는 각 후보 절 들에 대해서 AND 프로세스를 동시에 생성함으로써 OR-병렬성을 추구한다. 그런데 AND-병렬성은 절 내의 애톰들 사이의 공유 변수로 인해서 OR-병렬성에 비해서 다소 복잡하다. AND 프로세스는 애톰들 사이의 데이터 종속성을 나타내는 데이터 종속성 그래프를 이용하여 수행하며 전진 수행(forward execution)과 후진 수행(backward execution)의 두 페이즈(phase)를 반복하면서 진행한다 [5, 6]

본 연구에서는 AND/OR 프로세스 모델을 기초로 하여 개선된 병렬 수행 모델을 제공한다. 개선된 AND/OR 병렬 수행 모델에서는 최근 관심을 모으고 있는 컴파일 시간 상향식 추상 해석(bottom-up abstract interpretation)을 통하여 AND 프로세스에서 불필요한

†정 회 원 한국전자통신연구소
††종신회원 한국확학기술원 전산학과 교수
논문접수 1993년 7월 26일
심사완료 1993년 11월 16일

자식 OR 프로세스를 생성하지 않음으로써 시스템의 성능을 향상시킨다. 본 연구에서는 [1]에서 제안된 상향식 추상 해석을 [14]의 틀을 기초로하여 프로그램의 각 절의 성공 패턴(success pattern)에 대한 근사값를 구할 수 있도록 확장하였다 확장된 상향식 추상 해석에서는 깊이 k 추상화(depth k abstraction) [15]에 기초한 추상 도메인 상에서 논리 프로그램을 수행함으로써 프로그램의 각 절의 성공 패턴(success pattern)에 대한 추상화된 근사값를 구하고 이를 이용하면 어떤 호출(call)은 실제 수행에서 반드시 실패한다는 것을 결정할 수 있다. 상향식 추상 해석의 이러한 성질을 이용하여 AND 프로세스에서 실패할 것이 확실한 자식 OR 프로세스를 생성하지 않음으로써 AND/OR 프로세스 모델의 전체적인 시스템 성능을 향상시킨다

다음 절에서는 기초적인 기호 및 정의에 대해서 기술한다. 제 3 절에서는 확장된 상향식 추상 해석에 대해서 기술하고 제 4 절에서는 이를 이용한 개선된 AND/OR 병렬 수행 모델에 대해서 기술한다. 제 5 절에서는 개선된 병렬 수행 모델에 대해서 토론하고 제 6 절에서 결론을 맺는다.

2. 기호 및 정의

2.1 기초적인 기호

$(\prod, \sum, Var)$는 프레디키트(predicate) 심볼의 유한 집합 $\prod$, 함수 심볼의 유한 집합 $\sum$와 변수의 집합 Var를 나타내는 1차 언어이다. 함수 심볼 $f \in \sum$와 프레디키트 심볼 $p \in \prod$는 애러티(arity)라고 불리우는 자연수와 연관되어 있으며 애러티 n를 갖는 (프레디키트 혹은 함수) 심볼 f는 f/n으로 나타낸다

$\sum$와 Var로 부터 만들어진 모든 텀(term)들의 집합은 $Term(\sum, Var)$ 혹은 간단히 $Term$으로 표시된다.

애톰은 $p(t_1, \quad , t_n)$형태이며 $p/n \in \prod$이고 $t_1, \cdots, t_n \in Term(\sum, Var)$이다. $Atoms$는 $\prod$와 $Term(\sum, Var)$으로부터 만들어진 모든 애톰들의 집합을 나타낸다. $Horn$ 절(clause)은

$$H \leftarrow B_1, \quad , B_n$$

형태로 H는 머리(head)라고 불리는 애톰이고 $B_1, \cdots, B_n$은 몸체(body)라고 불리는 애톰들의 리스트이다 몸체가 없으면, 이 절을 사실(fact)이라고 부르며 그렇지 않으면 규칙(rule)이라고 부른다. 규칙은 그리이스 문자 δ등으로 나타낸다. 질의(query)는 애톰들의 연속으로써 $\leftarrow B_1, \quad , B_n$로 나타낸다

$Atoms$의 원소로 만들어진 절들의 집합을 $Clause(\prod, \sum, Var)$로 나타내며 간단히 $Clause$로 나타내기도 한다. 어떤 텀 t에 나타난 모든 변수의 집합은 $var(t)$로 나타낸다. 함수 gr은 텀 혹은 애톰들의 집합을 그들의 그라운드 인스턴스(ground instance)들의 집합으로 사상하는 함수로 정의한다.

대치(substitution) ϑ는 Var부터 $Term$로의 사상이다.모든 대치들의 집합은 Sub으로 나타낸다. 부분 함수(partial function) mgu는 한 쌍의 텀 혹은 애톰에 대해서 그들의 mgu(most general unifier)를 제공한다. $\theta = mgu(s,t)$는 s와 t가 유니파이 가능함을 의미한다.

mgu개념은 임의의 등식의 집합 $\{A_1 = B_1, \cdots, A_n = B_n\}$에 대해서 다음과 같이 확장 가능하다. 즉 $mgu(A_1, \cdots, A_n>, <B_1, \cdots, B_n>\}$는 등식의 집합 $\{A_1 = B_1, \cdots, A_n = B_n\}$ 에 대한 mgu를 의미한다.

S가 집합이고 $\sim$이 동등 관계(equivalence relation)일 때 $S/\sim$은 $\sim$에 대한 동등 클래스들의 집합을 나타낸다. 어떤 원소 $a \in S$에 대해서 $[a]\sim$는 $\sim$대한 a의 동등 클래스를 나타낸다.

2.2 실제 의미(Concrete Semantics)

[11]에서 소개된 실제 의미는 그라운드(ground) 되지 않는 텀과 애톰도 다룰 수 있게 함으로써 [18]에서 제안된 표준 의미를 확장했다. 이는 대치를 계산하는 논리 프로그램의 능력을 완전하게 특징지울 수 있는 의미를 제공한다.

확장된 Herbrand 유니버스(universe) U_p는 $Term(\sum, Var)/\sim$로써 $\sim$은 변이(variance) 관계를 나타낸다 (즉 $t_1 \sim t_2$ iff $\exists \vartheta_1 \vartheta_2 \mid t_1\vartheta_1 = t_2 \wedge t_2\vartheta_2 = t_1$). 변이 관계는 애톰, 절등으로도 쉽게 확장 가능하다.

확장된 Herbrand 베이스(base) B_p는 $Atoms/\sim$이며 $\sim$는 애톰에 대한 확장된 변이 관계이다. 해석(interpretation) I는 B_p의 부분 집합이다. 해석 I는 $gr(I)$가 p를 위한 Herbrand 모델이면 프로그램 P의 모델이다. 주어진 애톰에 대한 동등 클래스 $[A]\sim$와 유한 집합 V에 대해서 V의 변수를 포함하지 않는 $[A]\sim$내의 대표 A'를 항상 찾을 수 있다

어떤 절 c와 해석 I에 대해서 $<A_1, \cdots, A_n> <<_c I$는 $A_1, \cdots, A_n$가 c와 변수를 공유하지 않고 서로 변수 공유가 없는 I의 몇 원소들의 대표들을 나타낸다. 간단한 표현을 위해서 한 애톰이 그 동등 클래스를 나타낸다고 가정하고 $[A]\sim$ 보다는 A로 표현하겠다.

정의 1 ([11])
$T_p : 2^{B_p} \leftarrow 2^{B_p}$는 프로그램 P에 대한 다음과 같은 변환이다.

$$T_P(I)=\left\{ A\vartheta \;\middle|\; \begin{array}{l} c=A\leftarrow B_1\cdots,\ B_1,\ \cdots,\ B_n\in P \\ <B'_1,\ B'_n>\ll_c I \\ \vartheta=mgu(<B_1,\quad,B_n>,\ <B'_1,\quad,B'_n>) \end{array} \right\}$$

프로그램 P의 의미는 T_P의 최소 고정점 즉 $lfp(T_P)$에 의해서 결정되며 이는 다음과 같이 계산된다.

$$lfp(T_P)=T_P^{\omega}(\phi)=\bigcup_{i=0}^{\infty} T_P^{i}(\phi)$$

또한 $lfp(T_P)$는 P의 모델임이 [11]에서 증명되었다.

2.3 추상 해석

프로그램의 추상해석은 추상 도메인 상에서 그 프로그램의 실제 시맨틱(concrete semantic)에 대한 근사값을 제공한다. 기본적인 아이디어는 프로그램을 무한할 수 있는 실제 도메인에서 수행시키지 않고 추상 도메인에서 실행시킨다는 것이다. 일반적으로 추상해석은 추상 도메인 상에서 고정점 계산(fixed point computation)에 의해서 수행되며 프로그램의 실제 실행에 대한 정보를 제공한다 [7]. 추상 도메인은 보통 유한하거나 유한하지 않으면 무한한 증가 체인(ascending chain)을 허용하지 않는 래티스이다. 이 추상해석은 실제 시맨틱(즉 프로그램의 실제 수행중의 행동)에 대한 안전하고 유한한 근사값을 제공한다.

앞으로 우리는 [7]에서 정의된 추상 해석 표준 틀(standard framework)을 가정한다.이 틀에서는 최소 고정점(least fixpoint)에 의한 시맨틱스(semantics)을 가정한다.

이 표준 틀에서 프로그램 P를 위한 실제 해석은 실제 도메인 E 위에서 정의된 단조(monotonic) 연산자 $E_P: E\rightarrow E$에 의해서 정의되고, 추상 해석은 추상 도메인 D 위에서 정의된 단조(monotonic) 연산자 $D_P: D$에 의해서 정의된다고 가정한다.

정의 2 ([7,14])
추상 해석 $((E,\subseteq), E_P, (D,\leq), D_P, \alpha, \gamma)$은 아래의 조건을 만족하는 완전 래티스(*complete lattice*)$(E,\subseteq)$, 단조 연산자 $E_P: E\rightarrow E$, 완전 래티스(*complete lattice*)$(D,\leq)$, 단조 연산자 $D_P: D\rightarrow D$, 추상화 함수(*abstraction function*) $\alpha: E\rightarrow D$와 실제화 함수(*concretization function*) $\gamma: D\rightarrow E$로 구성된다.

(1) α와 γ는 단조(monotonic)하다,
(2) 모든 $d\in D$에 대해서 $d=\alpha(\gamma(d))$,
(3) 모든 $e\in$에 대해서 $e\subseteq\gamma(\alpha(e))$,
(4) 모든 $d\in D$에 대해서 $E_P(\gamma(d))\subseteq\gamma(D_P(d))$.

위의 조건 (1)-(3)은 $((E,\subseteq),\alpha,(D,\leq),\gamma)$가 Galois insertion을 형성함을 의미한다. 조건 (4)는 안전성 (safeness)조건으로써 D_P가 E_P를 충실하게 근접함을 의미한다.

명제 1 ([7])
$((E,\subseteq), E_P, (D,\leq), D_P, \alpha, \gamma)$이 추상 해석이면 $lfp(E_P)\subseteq\gamma(lfp(D_P))$이다.

3. 상향식 추상 해석

3.1 기본 틀

이 절에서는 [1]에 있는 상향식 추상 해석을 살펴보고 이를 애톰뿐만 아니라 절도 처리할 수 있도록 확장한다. 이 틀에서는 추상화 함수로써 깊이 k 추상화를 가정한다. 이 틀은 다른 추상 도메인으로도 쉽게 확장될 수 있다.

정의 3([15])
레벨 k 섭텀(subterm)은 다음과 같이 정의된다:

(a) 주어진 텀 t에 대해서, t는 t의 레벨 0 섭텀이다.

(b) t의 레벨 k 섭텀이 $f(t_1,\cdots,t_n)$이면, t_i은 t의 레벨 $k+1$ 섭텀이다.

우리는 깊이 k 추상화 [15]를 추상화 함수로 사용한다. 어떤 텀 t의 깊이 k 추상 텀은 t의 모든 레벨 k 섭텀을 새로운 변수(*fresh variable*)로 대치함으로써 얻어진다. $\rho_k(t)$는 어떤 텀 t를 그것의 깊이 k 추상 텀으로 사상하는 함수라고 하자.

예제 1
어떤 텀 $t=f(g(x,a),y,b)$에 대해서 $\rho_1(t)=f(v_1,\ v_2,\ v_3)$이고 $\rho_2(t)=f(g(w_1,\ w_2),y,b)$이다. 여기서 v_1, v_2, v_3, w_1, w_2는 새로 만들어진 변수들이다.

프로그램 P의 추상 유니버스(abstract universe)는 $U_P^A=\rho_k(Term(\sum,Var))$로 정의된다. 프로그램 P의 추상 베이스(abstract base) B_P^A는 $Atoms^A/\sim$로 정의되는데 여기서

$$Atoms^A=\{p(t_1^A\cdots,t_n^A)\mid p/n\in\prod, t_1^A,\cdot,t_n^A\subseteq U_P^A\}$$

이고 $\sim$은 애톰에 대한 변이 관계이다.

추상화 함수 $\alpha_a: B_P^A$는 $\alpha_a(p(t_1,\ \cdots,\ t_n))=[p(\rho_k(t_1),\quad,\rho_k(t_n))]\sim$로 정의된다. 추상화 함수는 다음과 같이 확장가능하다 $\alpha_a: 2^{B_P}\rightarrow 2^{B_P^A}$

$$\alpha_a(I)=\{\alpha_a(A)\mid A\in I\}$$

상응하는 실제화 함수 $\gamma_\alpha(I^\downarrow)$는 다음과 같다.

$$\gamma_\alpha(I^\downarrow) = \{A' \in B_P \mid A^\downarrow \in I^\downarrow,\ \alpha_\alpha(A') = A^\downarrow\}$$

소정리 1

$((p(B_P),\subseteq),\ \alpha_\alpha,\ (p(B^A_P),\subseteq),\ \gamma_\alpha)$는 Galois insertion 이다.

증명. 증명은 [4]를 참조하기 바람. □

추상 대치(abstract substitution)는 변수들의 유한 집합 $V \subseteq Var$를 U^A_P로 사상하는 함수이다. 어떤 대치 $\vartheta = \{t_1/x_1, \cdot, t_n/x_n\}$가 주어지면 α_Φ는 $\alpha_\Phi(\vartheta) = \vartheta^\downarrow$ 즉 $\{\rho_A(t_1)/x_1, \cdot, \rho_A(t_n)/x_n\}$로 정의된다. 상응하는 실제화 함수는 $(\alpha_\Phi, \gamma_\Phi)$가 Galois insertion 이라는 조건을 만족하여야 한다.

추상 연산자 T^α_P는 [1]에서 T_P에 대한 근사값을 구하기 위해서 다음과 같이 정의되었다

정의 4 ([1])

$T^A_P : 2^{B^\downarrow_P} \to 2^{B^\downarrow_P}$

$$T^A_P(I^A) = \left\{ \alpha_\alpha(A\ \vartheta^A) \;\middle|\; \begin{array}{l} c = A \leftarrow B_1, \cdot, B_n \in P, \\ <B^A_1, \cdot, B^A_n> \ll_c I^A, \\ \vartheta^A = mgu^A(<B_1, \cdot, B_n>, <B^A_1, \cdot, B^A_n>) \end{array} \right\}$$

여기서

$mgu^A(<B_1, \cdots, B_n>, <B^A_1, \cdot, B^A_n>) = \alpha_\Phi(\vartheta)$이고,

$\vartheta = (<B^A_1, \cdot, B^A_n>)$이다.

정리 1 ([1])

P가 논리 프로그램이라 하자. 어떤 유한한 n에 대해서 $lfp(T^A_P) = (T^\alpha_P)^n(\phi)$.이다.

증명. T^A_P는 단조(증가)함으로 Knaster-Tarski 정리 [17]에 의해서 $lfp(T^A_P)$가 존재하며 $(2^{B^\downarrow_P},\subseteq)$가 유한함으로 어떤 유한한 n에 대해서 $lfp(T^A_P) = (T^\alpha_P)^n(\phi)$. □

예제 2

P가 다음의 논리 프로그램이라 하자.

δ_1 : $path(X,[X\backslash P]) \leftarrow arc(X,N), path(N,P)$.

δ_2 : $path[X] \leftarrow final(X)$.

$final(f)$.

$are(a,b)$. $arc(a,c)$. $are(b,e)$. $arc(c,b)$. $arc(c,d)$. $arc(d,f)$.

추상화 깊이가 2일 때, $lfp(T^\alpha_P)$는 다음과 같으며 여기서 $path(X,[X\backslash P])$의 인스턴스들은 노드 f에 연결된 노드들을 나타낸다.

$$lfp(T^A_P) = \left\{ \begin{array}{l} arc(a,b), arc(a,c), arc(b,e), arc(c,b), arc(c,d), arc(d,f), final(f), \\ path(f,[f]), path(d,[d|\_]) \ path(c,[c|\_]), path(a,[a|\_]) \end{array} \right\}$$

다음의 소정리와 정리을 통해서 추상 해석의 안전성(safeness)을 보인다

소정리 2 ([1])

모든 $I^\downarrow \in 2^{B^\downarrow_P}$에 대해서 $T_P(\gamma_\alpha(I^\downarrow)) \subseteq \gamma_\alpha(T^\alpha_P(I^\downarrow))$

정리 2 ([1])

임의의 프로그램 p에 대해서 $\gamma_\alpha(lfp(T^\alpha_P)) \subseteq lfp(T_P)$.

증명. 소정리 1, 소정리 2 와 명제 1 로 부터 증명된다.□

추상화의 의미는 프로그램를 위한 추상 모델(abstract model)이라는 개념으로 설명될 수 있다. $I^\downarrow \in 2^{B^\downarrow_P}$가 P를 위한 추상 모델이면 P를 위한 모델인 실제 해석 I가 존재하고 그 역도 성립한다. [1]에서 보인 것처럼 프로그램 P의 안전한(safe) 상향식 추상 해석은 P를 위한 추상 모델을 제공한다.

정리 3 (strong soundness ([1]))

P가 논리 프로그램이고 $G : B_1, \cdot, B_n$는 질의라고 하자. ϑ가 P에 대한 G의 refutation에서 계산된 대치이면

$$\{\vartheta_{|G} \in \gamma_\Phi(mgu^A(<B_1, \cdots, B_n>, <B'_1, \cdot, B'_n>)_{|G} \mid <B'_1, \cdot, B'_n> \ll_G lfp(T^\alpha_P))\}$$

이다.

3.2 기본 틀의 확장

이 절에서는 앞의 기본 틀을 절를 고려함으로써 확장한다. C를 $Clause/\sim$ 즉 모든 가능한 절이라고 하면 추상 절의 집합은 다음과 같이 정의된다.

정의 5

$C^\downarrow = \{\alpha_c(c) \mid c \in C\}$ 이며 여기서 임의의 절 $c : A \leftarrow B_1, \cdot, B_n$에 대해서 $\alpha_c(A \leftarrow B_1, \cdot, B_n) = (\alpha'_\alpha(A) \leftarrow \alpha'_\alpha(B_1), \cdot, \alpha'_\alpha(B_n)]\sim$ 이고 $\alpha'_\alpha(B_i)$는 B_i를 해당하는 깊이 k 추상 애톰으로 사상한다.

우리는 앞의 기본적인 상향식 추상 해석을 다음의 추상 해석으로 확장한다.

$$((2^{B_P} \times 2^C, \subseteq),\ U_P,\ (2^{B^\downarrow_P} \times 2^{C^\downarrow}, \subseteq), U^A_P \alpha, \gamma).$$

확장된 추상 해석에서는 [14]의 방법론을 따라서 T_P을 애톰뿐만 아니라 절도 포함하도록 확장하여 실제 연산자(concrete operator) U_P를 정의하였다

정의 6

$U_P: 2^{B_P} \times 2^C \to 2^{B_I} \times 2^C$ 는 다음과 같이 정의된다

$U_P(I,J) = (T_P(I), J')$ 여기서

$$J' = \left\{ c\vartheta \;\middle|\; \begin{array}{l} c = A \leftarrow B_1, \ \ldots, B_n \in P \\ <B'_1, \ldots, B'_n> \ll_c I \\ \vartheta = mgu(<B_1, \ldots, B_n>, <B'_1, \ldots, B'_n>) \end{array} \right\}$$

프로그램 P의 실제 의미는 $lfp(U_P)$이고 그 첫번째 부분은 $lfp_a(U_P)$, 두 번째 부분은 $lfp_c(U_P)$로 표시한다. 추상화 함수 $\alpha : 2^{B_P} \times 2^C \to 2^{B_P^\sharp} \times 2^{C^\sharp}$는 $\alpha(I,J) = (\alpha_a(I), \alpha_c(J))$로 정의된다. 상응하는 실제화 함수 $\gamma : 2^{B_P^\sharp} \times 2^{C_P^\sharp} \to 2^{B_P} \times 2^{C_P}$는 $\gamma(I^\sharp, J^\sharp) = (\gamma_a(I^\sharp), \gamma_c(J^\sharp))$이며 여기서 $\gamma_c(J^\sharp) = \{c \mid c' \in J^\sharp, \alpha_c(c) = c'\}$이다.

추상 연산자 U_P^A는 T_P^A를 애톰뿐만 아니라 절도 포함하도록 확장함으로써 정의된다

정의 7

$U_P^A : 2^{B_P^\sharp} \times 2^{C^\sharp} \to 2^{B_P^\sharp} \times 2^{C^\sharp}$는 다음과 같이 정의된다.

$U_P^A(I^\sharp, J^\sharp) = (T_P^A(I^\sharp), J^\sharp)$

여기서

$$J^{A'} = \left\{ \alpha_c(c\vartheta^A) \;\middle|\; \begin{array}{l} c = A \leftarrow B_1, \ldots, B_n \in P, \\ <B_1^A, \ldots, B_n^A> \ll_c I^A, \\ \vartheta^A = mgu^A(<B_1, \ldots, B_n>, <B_1^A, \ldots, B_n^A>) \end{array} \right\}$$

추상 의미는 $lfp(U_P^A)$로써 정의되며 $lfp(U_P^A)$는 다음의 정리에서 그 계산이 유한 시간에 끝난다는 것이 증명된다. $lfp(U_P^A)$의 첫번째 부분은 $lfp_a(U_P^A)$로 두번째 부분은 $lfp_c(U_P^A)$로 표시된다. $lfp_a(U_P^A)$는 $lfp(U_P^A)$와 같으며 그 프로그램을 위한 추상 의미를 제공하며, $lfp_c(U_P^A)$는 몸체 애톰이 $lfp(U_P^A)$에 있는 모든 추상 절를 포함한다. $lfp_c(U_P^A)$는 실제 수행에서 각 절의 성공 패턴에 대한 근사값을 제공한다.

정리 4

P를 논리 프로그램이라 하자 어떤 유한한 n에 대해서 $lfp(U_P^A) = (U_P^A)^n(\phi)$

증명. U_P^A는 단조(증가)함으로 Knaster-Tarski 정리 [17]에 의해서 $lfp(U_P^A)$가 존재하며 $(2^{B_P^\sharp} \times 2^{C^\sharp}, \subseteq)$가 유한함으로 어떤 유한한 n에 대해서 $lfp(U_P^A) = (U_P^A)^n(\phi)$.□

예제 3

예제 2의 프로그램에 대한 U_P^A의 최소 고정점은 다음과 같다. 추상화 깊이가 2라고 가정하자.

$$lfp_a(U_P^A) = \left\{ \begin{array}{l} \mathrm{arc}(a,b)\ \mathrm{arc}(a\ c), \mathrm{arc}(b\ e), \mathrm{arc}(c\ b), \mathrm{arc}(c\ d), \mathrm{arc}(d\ f), \mathit{final}(f) \\ path(f,[f])\ path(d,[d|\_]), path(c,[c|\_]), path(a,[a|\_]) \end{array} \right\}$$

$$lfp_c(U_P^A) = \left\{ \begin{array}{l} \mathrm{arc}(a,b), \mathrm{arc}(a,c), \mathrm{arc}(b,e), \mathrm{arc}(c,b), \mathrm{arc}(c\ d), \mathrm{arc}(d\ f), \mathit{final}(f) \\ path(f,[f]) \leftarrow \mathit{final}(f), path(d,[d|\_]) \leftarrow \mathrm{arc}(d,f), path(f,[f]), \\ path(c,[c|\_]) \leftarrow \mathrm{arc}(c,d), path(d,[d|\_]), path(a,[a|\_]) \leftarrow \mathrm{arc}(a,c)\ path(c,[c|\_]) \end{array} \right\}$$

$lfp_a(U_P^A)$에 있는 $path$의 인스턴스들은 노드 f에 연결된 노드를 나타내며 그 수는 그래프의 노드 수를 넘지 못한다.

다음의 소정리와 정리는 확장된 상향식 추상 해석의 안전성을 증명한다.

소정리 3

모든 $I^\sharp \in 2^{B_P^\sharp}$, $J^\sharp \in 2^{C^\sharp}$에 대해서 $\gamma(U_P^A(I^\sharp, J^\sharp)) \subseteq U_P(\gamma(I^\sharp, J^\sharp))$.

증명. 증명은 [4]에 있음.□

정리 5

임의의 프로그램 P에 대해서 $\gamma(lfp(U_P^A)) \supseteq lfp(U_P)$이다

증명. $((2^{B_P} \times 2^C, \subseteq), \alpha, (2^{B_P^\sharp} \times 2^{C^\sharp}, \subseteq), \gamma)$이 Galois insertion 이라는 것은 쉽게 증명할 수 있다. 따라서 이 사실과 소정리 3 및 명제 1으로부터 증명된다.□

4. 개선된 병렬 수행 모델

이 절에서는 먼저 Prolog와 같은 논리 프로그램의 선형 수행(linear execution)을 앞에서 소개된 확장된 추상 해석의 결과를 이용해서 수행 성능이 향상될 수 있음을 보인다. 다음의 정리는 새로운 수행 모델에 관한 기초를 제공한다.

정리 6

$\delta = H \leftarrow B_1, \cdots, B_n$는 규칙이고 B'_i는 B_i의 인스턴스라고 하자. $lfp_c(U_P^A[\delta][i])$는 $lfp_c(U_P^A)$ 내에 있는 δ의 B_i의 모든 추상 인스턴스를 나타낸다고 하자.

만약 B'_i가 $lfp_c(U_P^A[\delta][i])$내의 어떤 추상 애톰과도 유니파이 되지 않으면 모든 $\sigma \in Sub$에 대해서 $B'_i\sigma \in lfp_c(U_P^A[\delta][i])$이다.

증명. B'_i는 $lfp_c(U^{\alpha}_P[\delta][i])$내의 어떤 추상 애톰과도 유니파이 되지 않으므로 B'_i는 $lfp_c(U^{\alpha}_P[\delta][i])$내의 어떤 추상 애톰과도 공통적인 인스턴스를 갖지 않는다. γ_α의 정의에 의하여, $\gamma_\alpha(lfp_c(U^{\alpha}_P[\delta][i]))$는 $lfp_c(U^{\alpha}_P[\delta][i])$내의 추상 애톰의 모든 인스턴스를 포함한다. 따라서, B'_i의 어떤 인스턴스도 $\gamma_\alpha(lfp_c(U^{\alpha}_P[\delta][i]))$내에 포함되지 않는다. 또한 정리 6에 의해서 $\gamma_\alpha(lfp_c(U^{\alpha}_P[\delta][i])) \supseteq lfp_c(U^{\alpha}_P[\delta][i])$이라는 것을 쉽게 증명할 수 있다. 그러므로, B'_i의 어떠한 인스턴스도 $lfp_c(U_P[\delta][i])$에
포함되지 않는다.□

위의 정리의 가정을 고려하여 새로운 수행 원칙을 살펴보면 다음과 같다.

- B'_i는 만약 B'_i가 $lfp_c(U^{\alpha}_P[\delta][i])$내의 어떠한 추상 애톰과도 유니파이 되지 않으면 호출될 필요가 없다. 이는 호출된 섭골이 실패할 것이 확실하기 때문이다
- B'_i가 호출되고 답 σ로 성공했을 때 만약 $B'_i\sigma$가 $lfp_c(U^{\alpha}_P[\delta][i])$내의 어떠한 추상 애톰과도 유니파이 되지 않으면 그 답은 제거된다 이는 구해진 답이 σ의 어떠한 성공 패턴에도 참가할 수 없음이 확실하기 때문이다.

예제 4
예제 2의 프로그램를 생각해 보자.주어진 질의 ←*path*(*a*,*P*)에 대한 새로운 수행 과정을 살펴보면 다음과 같다 질의 *path*(*a*,*P*)의 수행에서, δ_1에서의 호출 arc(*a*,*N*)은 arc(*a*,*b*)와 arc(*a*,*c*)로 성공한다. 그러나 arc(*a*,*b*)는 $lfp_c(U^{\alpha}_P[\delta][i])$의 어떠한 추상 애톰과도 유니파이 되지 않으므로 무시되고 따라서 단지 *path*(*c*,*P*)만이 호출된다. 호출 *path*(*c*,*P*)의 수행에서도 새로운 수행에서는 호출 *arc*(*c*,*N*)가 *arc*(*c*,*b*)와 *arc*(*c*,*d*)와 성공한다 그러나 위의 원칙에 근거하여 *arc*(*c*,*b*)는 제거하므로 *path*(*d*,*P*)만이 호출된다. 질의 *path*(*a*,*P*)부터 *path*(*f*,*P*)가 호출될 때까지 새로운 수행은 *arc*(*X*,*N*)에 대해서 5개의 호출을 하고 *path*(*X*,*N*)에 대해서 3개의 호출을 한다. 한편 개선되기 전의 원래의 수행에서는 *arc*(*X*,*N*)에 대해서 9개의 호출을 하고 *path*(*X*,*N*)에 대해서 7개의 호출을 한다.

확장된 상향식 추상 해석은 [3, 5, 9, 10, 16]와 같은 하향식 (병렬) 수행 모델을 개선하는데 쉽게 적용될 수 있다. 여기에서는 확장된 상향식 추상 해석을 이용하여 논리 프로그램의 내재된 AND 병렬성과 OR 병렬성을 이용한 AND/OR 프로세스 모델[5]를 개선한다. AND/OR 프로세스 모델에서는, 생성된 AND/OR 프로세스 트리가 bipartute 그래프를 형성한다: 한 절를 위한 AND 프로세스는 각 몸체 애톰을 위해서 자식 OR 프로세스를 생성하고, 한 애톰을 위한 OR 프로세스는 후보 절에 대해서 자식 AND 프로세스들을 동시에 생성한다.

부모 프로세스와 자식 프로세스는 *success*, *fail*, *redo*와 *cancel*등의 4종류의 메시지를 교환한다. *Success*와 *fail* 메시지는 자식 프로세스가 답을 찾는데 성공했는지 실패했는지를 알리기 위해서 자식 프로세스에서 부모 프로세스에게 보내진다.

Redo 와 *cancel* 메시지는 부모 프로세스가 자식 프로세스에게 다른 답을 찾도록 요청하거나 끝낼것을 요청하기 위해서 부모 프로세스에서 자식 프로세스에게 보내진다.

AND 프로세스는 한 절 내의 공유 변수를 처리하기 위해서 절 내의 변수를 공유하는 몸체 애톰들 사이의 생산자/소비자 관계를 나타내는 데이터 종속 그래프을 유지한다[3, 5]. 여기에서는 쉬운 이해를 위해서 컴파일 시간에 계산된 정적 데이터 종속그래프를 가정한다[3]. 제안된 방법은 동적 데이터 종속그래프에 대해서도 적용 가능하다.

AND 프로세스는 전진 수행(forward execution)과 후진 수행(backward execution)의 두 단계로 작동한다. 전진 수행은 일종의 그래프 축약 과정으로 success 메시지가 자식 OR 프로세스로부터 왔을 때 데이터 종속그래프에서 모든 생산자의 답이 구해진 몸체 애톰에 대해서 OR 프로세스를 생성한다 자식 OR 프로세스가 실패했을 때는, 후진 수행이 시작되며 백트랙(backtrack) 애톰를 찾고 해당 OR 프로세스에 redo 메시지를 보낸다 [5, 13, 19].

새로운 AND 프로세스에서 자식 OR 프로세스를 생성할 때는 정리 6을 기초로 한 다음의 원칙을 이용한다.

- B'_i을 위한 OR 프로세스는 만약 B'_i가 $lfp_c(U^{\alpha}_P[\delta][i])$내의 어떠한 추상 애톰과도 유니파이 되지 않으면 생성될 필요가 없다.
- B'_i를 위한 OR 프로세스가 생성되고 성공 메시지가 답 σ를 구해서 왔을 때 만약 $B'_i\sigma$가 $lfp_c(U^{\alpha}_P[\delta][i])$내의 어떠한 추상 애톰과도 유니파이 되지 않으면 그 답은 제거된다.

AND 프로세스가 생성될 때 수행된다. 두 번째 부분은 자식 OR 프로세스로부터의 성공 메시지를 처리하기 위해서 수행된다. 아래의 알고리즘에서 $pred(B_k)$는 규칙 δ을 위한 데이터 종속 그래프 DDG_δ에서 B_k의 모든 선행자(predecessor) 집합을 의미한다.

규칙 δ $H \leftarrow B_1$, , B_i을 위한 AND 프로세스에서 새로운 전진 수행 알고리즘

DDG_δ : δ을 위한 데이터 종속 그래프

Solved, *Blocked*, *Pending* . *AND*프로세스의 상태 변수 (각 상태에 있는 애톰들의 집합을 나타냄)

$\theta_{current}$: *rule* δ을 위한 *AND* 프로세스의 현재 해답(*solution*)

To process a start message

1. Initialization: *Solved* $= H$, *Pending* $= \phi$, *Blocked* $= B_1$, B_i, and
   $\theta_{current} = \theta_{i,j}$ where $\theta_{i,j}$ is the mgu of the atom of the parent OR process and H
2. For every atom B such that $pred(B) \subseteq$ *Solved*,
   if $B' = B\,\theta_{current}$ is unifiable with an abstract atom in $lfp_c(U_P^{\alpha}[\delta][i])$, then create OR process for the atom and move B from *Bolcked* to *Pending*,
   else call *backward execution algorithm.*

To process a success message with solution σ *from OR process for* B_i :

1. If $B' = B\,\theta_{current}\sigma$ is not unifiable with any of the abstract atoms in $lfp_c(U_P^{\alpha}[\delta][i])$ then discard it and ask one more solution (redo) to the OR process.
2. Otherwise $\theta_{current} = \theta_{current}\sigma$
   2.a Move the solved atom from *Pending* to *Solved*
   2.b If all the body atoms are in *Solved*, then send a success message to the parent process, else continue.
   2.c For each body atom $\{B_k \mid B_k$ *Blocked*, *pred* $(B_k) \subseteq$ *Solved*$\}$
   if $B'_k = B_k \in \theta_{current}$ is unifiable with an abstract atom in $lfp_c(U_P^{\alpha}[\delta][k])$, then create an OR process and move B_k from *Blocked* to *Pending*,
   else call backward execution algorithm.

위의 알고리즘에서, 추상 해석을 통해서 OR 프로세스가 실패할 것이 확실한 경우에는 OR 프로세스를 생성하는 대신에 후진 수행 알고리즘을 호출한다.
여기서는 [19]에 있는 후진 수행 알고리즘을 가정한다

5. 토 론

상향식 실제 수행에서 모든 성공 패턴을 생성하는 것은 많은 시간이 요구된다. 본 논문에서의 상향식 추상 해석에서는 이 문제를 깊이 k 추상화를 이용하여 부분적으로 해결하였다. 예제 2에서 어떤 그래프 $G = (V,E)$가 프로그램에서 사실로써 나타났다고 가정하고 상향식 추상 해석의 복잡도를 계산하면 다음과 같다. 상향식 추상 해석에서, 추상화 깊이가 2일때 그래프 G가 사이클을 갖든 안갖든 $lfp_\alpha(U_P^{\alpha})$에 나타날 수 있는 δ_1의 인스턴스의 갯수는 $|E|$를 넘지못하고 $lfp_c(U_P^{\alpha})$에 나타날 수 있는 δ_1의 인스턴스의 갯수는 $|E|$를 넘지못한다 여기서 $|S|$는 집합 S의 원소의 갯수를 나타낸다. 반면에 실제 상향식 수행에서는 그래프 G가 사이클을 안갖는 경우 $lfp(T_P)$에 나타날 수 있는 $path(X,[X \mid P])$의 인스턴스의 갯수는 $|V|$를 넘지못하며 그래프 G가 사이클을 갖는 경우에는 무한대가 될 수 있다.

위의 분석에서 보여진 것처럼 작은 깊이의 추상화를 이용하면 상향식 추상 해석의 복잡도가 크지 않으며 앞의 예제에서 보여진 것처럼 작은 깊이의 추상화로도 불필요한 OR 프로세스를 제거하는데 유효하다는 점은 주목되어야한다. 또한, 제안된 방법은 다음의 예제에서처럼 어떤 프로그램에 대해서는 무한 수행을 방지할 수 있다.

예제 5

예제 2에 나타난 rule과 질의 $\leftarrow path(a, X)$ 및 사이클을 포함한 다음의 그래프를 생각해 보자.

arc(a,b). arc(a,c) arc(b,e) arc(c,b). arc(c,d). arc (d,f). arc(e,b).

원래의 수행에서 δ의 호출 $path(b,P)$는 사이클 때문에 무한 호출을 하게된다. 그러나 개선된 수행에서는 추상 해석에 의한 계산 결과에 포함되지 않으므로 $path(b,P)$는 호출되지 않는다. 따라서 무한 호출도 발생하지 않는다.

상향식 추상 해석은 질의 무관성과 관련된 또 하나의 장점을 갖는다. 즉 프로그램의 분석 결과는 그 프로그램에 대한 어떠한 질의에도 공통적으로 사용될 수 있으므로 질의가 바뀌어도 추가의 분석이 필요하지 않다. 일반적으로 하나의 프로그램에 대해서 여러가지의 질의가 있을 수 있으므로 이러한 특성은 컴파일 시간 최적화를 위한 큰 장점이 될 수 있다.

6. 결 론

본 논문에서는 기존의 상향식 추상 해석를 확장하여 모든 절의 성공 패턴에 대한 근사값을 제공하도록 하고 이를 이용하여 논리 프로그램의 AND/OR 병렬 수행 모델을 개선하였다.새로운 AND/OR 병렬 수행 모델은 실패할 것이 확실한 불필요한 OR 프로세스를 생성하지 않음으로써 전체적인 수행 효율을 높였다

현재 제안된 상향식 추상 해석은 깊이 k 추상화에 기초하고 있는데 이는 그라운드 의존성, 변수 공유 추론, 타입 추론 등의 다른 추상 도메인으로 쉽게 확장가능하며 이를 이용하면 새로운 형태의 병렬 수행 모델의 개선도 가능할 것이다.

참 고 문 헌

[1] Barbuti, R., Giacobazzi R., and Levi, G A general framework for semantics-based bottom-up abstract interpretation of logic programs. *ACM Transactions on Programming Languages and Systems, 15*, No. 1, pp. 133-181, 1993.

[2] Chang, J H., and Despain, A. "Semi-intelligent Backtracking of Prolog Based on Static Data Dependency Analysis", *Proc of the 1985 Symp on Logic Programming*, pp. 10-21, July 1985.

[3] Chang, J.-H., Despain, A. M., and DeGroot, D. AND-parallelism of logic programs based on static data dependency analysis. In *Proceedings of the 30th IEEE Computer Society International Conference*, pp. 218-226, 1985.

[4] Chang, B.-M., Choe, K.-M., Han, T. and Giacobazzi, R., "An Efficient Execution of Logic Programs Using Bottom-up Abstract Interpretation", Technical Report, CS-TR-93-78, Dept. of Computer Science, KAIST, April 1993.

[5] Conery, J., "The AND/OR Process Model for Parallel Interpretation of Logic Programs", *Ph D dissertation, Technical Report 204, U C.Irvine*, June 1983.

[6] Conery, J., and Kibler, D., "AND Parallelism and Nondeterminism in Logic Programs", *New Generation Computiong, Vol 3*, pp. 43-70, Springer-Verlag, 1985.

[7] Cousot, P., and Cousot., R. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixedpoints, In *Proceedings of the 4th ACM Symposium on Principles of Programming Languages*, ACM, New York, 1977, pp. 238-252.

[8] DeGroot, D, "Restricted And-Parallelism", *ICOT International Conference on Fifth Generation Computer Systems*, pp. 471-478, 1984 Computer Conference, Vol 16, No. 9, pp. 821-844, September 1986.

[9] DeGroot, D. Restricted AND-parallelism, In *Proceedings of the International Conference on Fifth Generation Computer Systems*, ICOT, Tokyo, 1984, pp. 471-478.

[10] Dietrich, S.W, and Warren, D.S. Extension tables: memo relations in logic programming, In *Proceedings of the 4th IEEE Symposium on Logic Programming*, IEEE Comp. Soc Press, Washington, 1987, pp 264-273.

[11] Falaschi, M., Levi, G., Palamidessi, C., and Martelli, M. Declarative modeling of the operational behavior of logic languages. *Theoretical Computer Science 69 (1989)*, 289-318.

[12] Kale, L.V., "The REDUCE-OR Process Model for Parallel Evaluation of Logic Programs," *Proc of the Fourth International Conference on Logic Programming*, The MIT Press, pp. 616-632, Melbourne, Australia, May 1987.

[13] Kim, D.-H., Choe, K.-M., and Han, T. Refined mark(s)-set-based backtrack literal selection for AND parallelism in logic programs, *Parallel Processing Letters 2* (1992), 71-79.

[14] Marriott, K., and S o ndergaard, H. Bottom-up abstract interpretation of logic programs. In *Proceedings of the 5th International Conference on Logic Programming*, The MIT Press, Cambridge, Mass., 1988, pp. 733-748.

[15] Sato, T., and Tamaki, H Enumeration of success patterns in logic programs. *Theoretical Computer Science 34 (1984)*, 227-240.

[16] Tamaki, H., and Sato, T. OLD resolution with tabulation. In *Proceedings of the 3rd International Conference on Logic Programming*, Lecture Notes in Computer Science. Vol. 239. Springer-Verlag, Berlin, 1986, pp. 84-98.

[17] Tarski, A. A lattice theoretical fixed point theorem and its applications. *Pacific Journal of Mathematics 5*, (1955), pp. 289-321.

[18] van Emden, M.~H. and Kowalski, R.~A The semantics of predicate logic as a programming language. *Journal of the ACM 23*, 4 (1976), 733

-742.

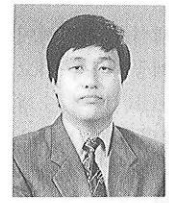
[19] Woo N., and Choe, K.-M. Selecting the backtrack literal in the AND/OR process model. In *Proceedings of the 3th* IEEE International Symposium on Logic Programming, IEEE Comp. Soc. Press, Washington, 1986, pp. 200-210.

창 병 모

1988년 2월 서울대학교 컴퓨터공학과 졸업(학사). 1990년 2월 한국과학기술원 전산학과에서 공학 석사학위 취득. 1994년 2월 한국과학기술원 전산학과에서 공학 박사학위 취득. 1994년 3월부터 한국전자통신연구소 post-doc 연구원으로 근무. 관심분야는 컴파일러, 병렬 언어 및 병렬 시스템, 논리프로그래밍, 연역 데이타베이스 등임.

최 광 무

1976년 서울대학교 전자공학과 졸업(학사). 1978년 한국과학기술원 전산학과 졸업(석사). 1984년 한국과학기술원 전산학과 졸업(박사). 현재 한국과학기술원 전산학과 부교수. 관심분야는 프로그래밍 언어론, 논리 프로그램의 병렬 수행 및 컴파일러 구성임.



한 태 숙

1976년 서울대학교 전자공학과 졸업(학사). 1978년 한국과학기술원 전산학과 졸업(석사). 1990년 Univ. of North Carolina at Chapel Hill 졸업(박사). 현재 한국과학기술원 전산학과 조교수. 관심분야는 프로그래밍 언어론, 함수형 언어임.