일반화된 Let-다형성 타입 유추 알고리즘
(A Generalized Let-Polymorphic Type Inference Algorithm)

이 욱 세 † 이 광 근 ††
(Oukseh Lee) (Kwangkeun Yi)

요 약 본 연구에서는 일반화된 let-다형성(let-polymorphic) 타입 유추 알고리즘을 제시하고, 이로부터 얻어지는 모든 예 알고리즘들은 Hindley/Milner 타입 체계를 안전하고 완전하게 (sound and complete) 구현하고 있음을 증명하며, 일반화된 알고리즘의 두 예 알고리즘들간에 어느 한쪽이 항상 오류를 더 신속히 감지하게 되는 조건을 제시한다.

일반화된 알고리즘으로부터 이론적으로 검증된 두 타입 유추 알고리즘, 즉, 상향성 표준 타입 유추 알고리즘 W와 하향성 알고리즘 M 뿐만 아니라 두 알고리즘의 혼성 알고리즘(hybrid algorithm)을 만들 수 있다. 만들어진 예 알고리즘들의 안전성, 완전성, 상대적인 신속한 오류 감지 성질들은 본 논문의 증명에 의해 자동적으로 유추된다. 본 논문에서 제시하는 알고리즘으로부터 만들어 낼 수 있는 예 알고리즘에는 SML/NJ와 Objective Caml 컴파일러에서 채택한 알고리즘을 포함하고 있다.

Abstract We present a generalized let-polymorphic type inference algorithm, prove that any of its instances is sound and complete with respect to the Hindley/Milner let-polymorphic type system, and present a condition that one instance algorithm always detects type errors earlier than the other.

By instantiating the generalized algorithm with different parameters, we can obtain not only the two opposite algorithms (the bottom-up standard algorithm W and the top-down folklore algorithm M) but also other various hybrid algorithms that avoid their extremities in type-checking (W fails too late, while M fails too early). Such hybrid algorithms' soundness, completeness, and their relative earliness in detecting type-errors can be obtained automatically. The set of hybrid algorithms that come from the generalized algorithm is a superset of those used in the two most popular ML compilers, SML/NJ and OCaml.

1. 서 론

Hindley/Milner let-다형성 타입 체계(let-polymorphic type system)[1]의 서로 다른 타입 유추 알고리즘, 즉, 상향성 표준 타입 유추 알고리즘 *W*[1,2,3]와 하향성 타입 유추 알고리즘 *M*[4,5]은 타입 유추에 있어 극단적이다. *W* 알고리즘은 프로그램의 문맥에 영향 받지 않고(context-insensitive) 타입을 유추하므로 타입 오류를 늦게 감지한다. 반면에 *M* 알고리즘은 최대한 문맥에 따라(context-sensitive) 타입을 유추하므로 신속히 오류를 감지한다. *W* 알고리즘은 함수 적용(application)시 함수와 인자의 타입을 독립적으로 유추하고 나서 서로 충돌이 일어나는지 검사한다. 그렇기 때문에, 타입 오류가 있는 프로그램에 대해 실제 타입 오류가 있는 지점을 성공적으로 유추한 뒤 나중에 오류를 감지하게 되는 경우가 많다. 반면에, *M* 알고리즘은 프로그램의 문맥을 보고 최대한의 정보를 실은 타입 조건(type constraint) 또는 기대 타입(expected type)을 만들어서 유추한다. 만들어진 기대 타입을 만족하지 않는 지점에서 멈추므로 때로 타입 오류가 있는 지점보다 일찍 오류를 감지하게 된다. 예를 들어, 함수 적용 "1 2"를 *W*로 유추하면 1, 2를 각각 유추하고 나서 1에다 2를 적용할 수 없으므로 실패하고, *M*으로 유추하면 1을 유추할 때 함수 타입을 기대하였는데 그렇지 않아 실패하게 된다.

실제 컴파일러 시스템에서는 극단적이지 않은 타입

† 비 회 원 : 한국과학기술원 전자전산학과
cookcu@ropas.kaist.ac.kr
†† 종신회원 : 한국과학기술원 전자전산학과 교수
kwang@cs.kaist.ac.kr
논문접수 : 2000년 6월 29일
심사완료 : 2000년 10월 21일

유추 알고리즘이 필요한데, 이러한 혼성 알고리즘을 고안하는 체계적인 방법이 없다. 이미 SML/NJ[6]와 Objective Caml(이하 OCaml)[7]에서는 혼성 알고리즘(hybrid algorithm)을 사용하고 있는데, 이러한 알고리즘들의 안전성(soundness)과 완전성(completeness)도 증명되지 않았고, 또한 두 알고리즘이 어떻게 다른지도 알려지지 않았다. M 알고리즘의 수행 속도 개선을 위해 제시한 H 알고리즘[8,9]의 경우에도 일일이 안전성과 완전성을 증명해야 했고 또한 M보다는 늦게 W보다는 신속히 오류를 감지하는 성질을 증명해야 했다.

이러한 혼성 알고리즘을 체계적으로 고안하고 증명하기 위해서 (1) 어떻게 두 알고리즘을 섞어서 혼성 알고리즘을 고안하는지, (2) 고안된 알고리즘이 여전히 안전하고 완전한지, (3) 그리고 가능하다면 상대적으로 타입 오류를 신속히 (또는 늦게) 감지하는지를 찾아낼 수 있는 체계가 필요하다.

본 연구에서는 일반화된 타입 유추 알고리즘을 제시하고, 이로부터 만들어 낼 수 있는 예 알고리즘이 항상 안전하고 완전함을 증명하고, 두 예 알고리즘 사이에 어느 한쪽이 항상 더 신속히 오류를 감지하는 조건을 제시한다. 제시된 일반화된 타입 유추 알고리즘으로부터 W, M뿐만 아니라 여러 혼성 알고리즘을 만들어 낼 수 있으며, 이들의 안전성과 완전성은 자동적으로 증명된다. 또한, 간단한 조건만 증명함으로써 상대적으로 신속히 오류를 감지하는 성질도 자동적으로 이끌어진다. 일반화된 알고리즘으로부터 만들 수 있는 혼성 알고리즘에는 현재 널리 사용되고 있는 SML/NJ와 OCaml의 타입 유추 알고리즘과 H도 포함하고 있다.

1.1 수식

사용된 수식은 다음과 같다. 특별한 표시가 없으면 α, β, γ는 타입 변수(type variable)를, τ, ρ, θ는 타입을, σ는 타입 스킴(type scheme)을 Γ는 타입 환경(type environment)을, P, Q, R, S는 타입 치환 함수(substitution)를 의미한다. 벡터 $\vec{\alpha}$는 $\{\alpha_1,\cdots,\alpha_n\}$을, $\forall\vec{\alpha}.\tau$는 $\forall\alpha_1.\cdots\forall\alpha_n.\tau$를, $R\vec{\alpha}$는 $\{R\alpha_1,\cdots,R\alpha_n\}$를 줄여 표기한 것이다.

두 타입 스킴이 같다는 것은 구속 변수(bound variable)의 이름을 바꾸어 같아질 수 있음을 말한다. 자유 타입 변수(free type variable)는 구속되지 않은 타입 변수들로, 타입 스킴 $\forall\vec{\alpha}.\tau$의 자유 타입 변수 집합 $ftv(\forall\vec{\alpha}.\tau)$는 $ftv(\tau)\backslash\vec{\alpha}$로 정의된다. 여기서 $ftv(\tau)$는 타입 에 나타난 모든 자유 타입 변수들의 집합을 말한다.

타입 환경은 각 변수의 타입 스킴을 저장하고 있는 것으로 $\{x_1\mapsto\sigma_1,\cdots x_n\mapsto\sigma_n\}$과 같이 표기한다. $\Gamma+x:\sigma$는 타입 환경 Γ에서 변수 x에 대한 이전의 타입 스킴을 버리고 새로운 σ로 바꾼 타입 환경을 의미하고, $\{y\mapsto\sigma' \mid x\neq y,\ y\mapsto\sigma'\in\Gamma\}\cup\{x\mapsto\sigma\}$로 정의된다. 타입 환경 Γ의 자유 타입 변수 집합 $ftv(\Gamma)$는 $\cup_{x\in dom(\Gamma)} ftv(\Gamma(x))$ 로 정의된다.

타입 치환 함수(substitution)는 타입 변수를 타입으로 치환하는 것으로, $\{\tau_1/\alpha_1,\cdots,\tau_n/\alpha_n\}$로 쓰거나 줄여서 $\{\vec{\tau}/\vec{\alpha}\}$로 쓴다. 타입 치환 함수 S에 대해, $supp(S)$는 지원 변수(support)의 집합으로 $\{\alpha \mid S\alpha\neq\alpha\}$로 정의하고, $itv(S)$는 연루된 타입 변수의 집합으로 $\{\alpha \mid \beta\in supp(S),\alpha\in\{\beta\}\cup ftv(S\beta)\}$로 정의된다. 타입 τ에 대해 $S\tau$는 τ에 있는 S의 지원 변수 α_i를 $S(\alpha_i)$로 치환한 타입을 말하고, 타입 스킴 σ에 대해 $S\sigma$는 자유 타입 변수만 치환하여 얻는 것을 말한다. 즉, σ가 $\forall\vec{\alpha}.\tau$일 때 $\vec{\beta}\cap(itv(S)\cup ftv(\sigma))=\emptyset$를 만족하는 $\vec{\beta}$에 대해 $S\sigma=\forall\vec{\beta}.S\{\vec{\beta}/\vec{\alpha}\}\tau$로 정의된다. 타입 환경 Γ에 대해 $S\Gamma$는 $\{x\mapsto S\sigma \mid x\mapsto\sigma\in\Gamma\}$로 정의된다. 두 타입 치환 함수 S와 R를 연속으로 적용한 RS는 $\{R(S\alpha)/\alpha \mid \alpha\in supp(S)\}\cup\{R\alpha/\alpha \mid \alpha\in supp(R)\setminus supp(S)\}$로 정의된다. 두 타입 치환 함수 S, R이 같다는 것은 모든 $\alpha\in supp(S)\cup supp(R)$에 대해 $S\alpha=R\alpha$가 성립할 때를 말한다. 타입 치환 함수 P와 타입 변수 집합 V에 대해 $P\downarrow_V$는 P와 같되 V를 지원하지 않는 타입 치환 함수를 말하는 것으로 $\{\tau/\alpha\in P \mid \alpha\notin V\}$로 정의된다.

타입 τ, 타입 스킴 $\forall\vec{\alpha}.\tau'$에 대해 $S\tau'=\tau$이고 $supp(S)\subseteq\vec{\alpha}$를 만족하는 S가 존재하면, τ가 $\forall\vec{\alpha}.\tau'$의 예(instance)라 하고 $\forall\vec{\alpha}.\tau'>\tau$로 표기한다. 타입스킴 $\sigma=\forall\vec{\alpha}.\tau$, $\sigma'=\forall\vec{\beta}.\tau'$에 대해, $supp(S)\subseteq\vec{\alpha}$와 $\tau'=S\tau$를 만족하는 타입 치환 함수 S가 존재하고 $ftv(\sigma)\subseteq ftv(\sigma')$이면, σ'를 σ의 일반예(generic instance)라 하고 $\sigma>\sigma'$로 표기한다. 또한 타입 환경 Γ, Γ'에 대해 $dom(\Gamma)=dom(\Gamma')$이고, 모든 $x\in dom(\Gamma)$에 대해 $\Gamma(x)>\Gamma'(x)$일 때 $\Gamma>\Gamma'$라 표기한다. $Clos_\Gamma(\tau)$는 [2]의 $\mathrm{Gen}(\Gamma,\tau)$와 같은 것으로 $\vec{\alpha}=ftv(\tau)\setminus ftv(\Gamma)$ 일 때 $\forall\vec{\alpha}.\tau$로 정의된다.

타입 유추 알고리즘에는 다음 성질을 만족하는 동일화(unification) 알고리즘 U가 사용된다.

Expr	e	::=	$()$	상수
		|	x	변수
		|	$\lambda x.e$	함수
		|	$e\ e$	함수 적용
		|	$\texttt{let}\ x = e\ \texttt{in}\ e$	let 구문
		|	$\texttt{fix}\ f\ \lambda x.e$	재귀 함수
Type	τ	::=	ι	상수 타입
		|	α	타입 변수
		|	$\tau \to \tau$	함수 타입
TypeScheme	σ	::=	$\tau \mid \forall\vec{\alpha}.\sigma$	타입 스킴
TypeEnv	Γ	$\in$	$Var \xrightarrow{fin} TypeScheme$	타입 환경

$$\text{(CON)}\quad \Gamma \vdash () : \iota$$

$$\text{(VAR)}\quad \frac{\Gamma(x) \succ \tau}{\Gamma \vdash x : \tau}$$

$$\text{(FN)}\quad \frac{\Gamma + x:\tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda x.e : \tau_1 \to \tau_2}$$

$$\text{(APP)}\quad \frac{\Gamma \vdash e_1 : \tau_1 \to \tau_2 \quad \Gamma \vdash e_2 : \tau_1}{\Gamma \vdash e_1\ e_2 : \tau_2}$$

$$\text{(LET)}\quad \frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma + x: Clos_\Gamma(\tau_1) \vdash e_2 : \tau_2}{\Gamma \vdash \texttt{let}\ x = e_1\ \texttt{in}\ e_2 : \tau_2}$$

$$\text{(FIX)}\quad \frac{\Gamma + f:\tau \vdash \lambda x.e : \tau}{\Gamma \vdash \texttt{fix}\ f\ \lambda x.e : \tau}$$

그림 1 언어와 타입 유추 규칙

정리 1 *(동일화 알고리즘 U)[10]. 두 개의 타입을 입력으로 받아, 다음과 같은 타입 치환 함수를 주는 U 알고리즘은 존재한다.*

• $U(\tau,\tau')$가 S를 결과로 성공하면, S는 τ와 τ'를 동일화한다. 즉, $S\tau = S\tau'$ 이다.

• τ와 τ'를 동일화하는 타입 치환 함수(unifier) S'가 존재하면, $U(\tau,\tau')$는 성공하고, 결과 S에 대하여 $S' = RS$ 인 타입 치환 함수 R이 존재한다. 이러한 S를 가장 일반적인 동일화 함수(most general unifier)라 한다.

게다가 $U(\tau,\tau')$는 τ와 τ'의 자유 타입 변수에만 연루되어 있다.

2. 일반화된 타입 유추 알고리즘 G

2.1 개요

언어와 Hindley/Milner 타입 체계는 그림 1에 있고 타입 유추 알고리즘 W와 M은 그림 2에 있다.

일반화된 알고리즘은 하향성이고 문맥에 따라 유추하는 M 알고리즘을 기반으로 한다. 핵심은 M 알고리즘에서 두 가지 요소를 변경시킬 수 있도록 하는 것이다. 하나는 타입 조건에 실을 정보의 양이고 또 하나는 동일화 알고리즘의 위치이다. M 알고리즘은 타입 조건에 가능한 많은 정보를 실어 보내고 각 상수, 변수, 함수에서 동일화한다. W 알고리즘은 타입 조건에 정보를 실어 보내지 않으며 함수 적용과 재귀 함수 정의에서만 동일화한다. 이 두 가지 요소를 변경시킴으로써 여러 전략의 타입 유추 알고리즘을 만들 수 있다. 반면에 W 알고리즘을 기반으로 하면 타입 조건을 입력으로 받지 않기 때문에 변경시킬 여지가 없다.

예제 1 다음 프로그램의 타입을 유추하여 보자.

(IsOne 2) : bool

여기서 IsOne은 인자가 1인지 검사하는 함수로 타입이 int→bool 이다. 타입 조건을 점점 느슨하게 해 가면서 여러 방법으로 타입 유추해 보자.

- IsOne을 최대 조건인 $\beta \to$ bool 로 타입 유추한다. 이 유추가 성공하고 나면 IsOne의 인자 타입은 int 임을 알게 된다. 그러면, 2를 int 조건으로 유추한다. (M)
- 보다 느슨한 조건인 $\beta_1 \to \beta_2$ 로 IsOne을 타입 유추한다. 여기서 β_1, β_2는 새 타입 변수로 IsOne을 함수 조건으로 유추한다는 의미이다. 즉, 인자 타입이나 결과 타입은 상관없이 IsOne이 함수이기만 기대한다는 의미이다. 이 유추가 성공하고 나서 IsOne의 결과 타입이 bool인지 검사한다. 그리고 나서 IsOne의 인자 타입 int 조건으로 2를 유추한다. (H)
- 이번에는 IsOne을 조건 없이 유추한다. 성공한 뒤 IsOne의 타입이 $\beta \to$ bool 인지 검사한다. 그리고 나서 IsOne의 인자 타입 int 조건으로 2를 유추한다. (OCaml)
- IsOne을 조건 없이 유추한다. 성공한 뒤 IsOne의 타입이 함수인지 검사한다. 그리고, 2를 int 조건으로 유추한다. 그리고, IsOne의 결과 타입이 bool 인지 검사한다.
- 마찬가지로 IsOne을 조건 없이 유추하고 IsOne의 타입이 함수인지 검사한다. 그러나, 2를 조건 없이 유추한다. 그리고 나서, IsOne의 인자 타입이 2의 타입인지, 그리고, IsOne의 결과 타입이 bool 인지 검사한다.
- IsOne을 조건 없이 유추한다. 그러나, IsOne의 타입을 검사하지 않는다. 2 또한 조건 없이 유추한다. 모든 유추가 성공하고 나서 IsOne의 인자 타입이 2의 타입인지 그리고, IsOne의 결과 타입이 bool인지 검사한다. (W) □

$\mathcal{W}$: $TypEnv \times Expr \to Subst \times Type$

$\mathcal{W}(\Gamma, ()) = (id, \iota)$ ($\mathcal{W}$.1)
$\mathcal{W}(\Gamma, x) = (id, \{\vec{\beta}/\vec{\alpha}\}\tau)$ where $\Gamma(x) = \forall\vec{\alpha}.\tau$, new $\vec{\beta}$ ($\mathcal{W}$.2)
$\mathcal{W}(\Gamma, \lambda x.e) =$
  let $(S_1, \tau_1) = \mathcal{W}(\Gamma + x{:}\,\beta, e)$, new β ($\mathcal{W}$.3)
  in $(S_1, S_1\beta \to \tau_1)$
$\mathcal{W}(\Gamma, e_1\ e_2) =$
  let $(S_1, \tau_1) = \mathcal{W}(\Gamma, e_1)$ ($\mathcal{W}$.4)
    $(S_2, \tau_2) = \mathcal{W}(S_1\Gamma, e_2)$ ($\mathcal{W}$.5)
    $S_3 = \mathcal{U}(S_2\tau_1, \tau_2 \to \beta)$, new β ($\mathcal{W}$.6)
  in $(S_3S_2S_1, S_3\beta)$
$\mathcal{W}(\Gamma,$ `let` x `=` e_1 `in` $e_2) =$
  let $(S_1, \tau_1) = \mathcal{W}(\Gamma, e_1)$ ($\mathcal{W}$.7)
    $(S_2, \tau_2) = \mathcal{W}(S_1\Gamma + x{:}\,Clos_{S_1\Gamma}(\tau_1), e_2)$ ($\mathcal{W}$.8)
  in (S_2S_1, τ_2)
$\mathcal{W}(\Gamma,$ `fix` $f\ \lambda x.e) =$
  let $(S_1, \tau_1) = \mathcal{W}(\Gamma + f{:}\,\beta, \lambda x.e)$, new β ($\mathcal{W}$.9)
    $S_2 = \mathcal{U}(S_1\beta, \tau_1)$ ($\mathcal{W}$.10)
  in $(S_2S_1, S_2\tau_1)$

$\mathcal{M}$: $TypEnv \times Expr \times Type \to Subst$

$\mathcal{M}(\Gamma, (), \rho) = \mathcal{U}(\rho, \iota)$ ($\mathcal{M}$.1)
$\mathcal{M}(\Gamma, x, \rho) = \mathcal{U}(\rho, \{\vec{\beta}/\vec{\alpha}\}\tau)$ ($\mathcal{M}$.2)
  where $\Gamma(x) = \forall\vec{\alpha}.\tau$, new $\vec{\beta}$
$\mathcal{M}(\Gamma, \lambda x.e, \rho) =$
  let $S_1 = \mathcal{U}(\rho, \beta_1 \to \beta_2)$, new β_1, β_2 ($\mathcal{M}$.3)
    $S_2 = \mathcal{M}(S_1\Gamma + x{:}\,S_1\beta_1, e, S_1\beta_2)$ ($\mathcal{M}$.4)
  in S_2S_1
$\mathcal{M}(\Gamma, e_1\ e_2, \rho) =$
  let $S_1 = \mathcal{M}(\Gamma, e_1, \beta \to \rho)$, new β ($\mathcal{M}$.5)
    $S_2 = \mathcal{M}(S_1\Gamma, e_2, S_1\beta)$ ($\mathcal{M}$.6)
  in S_2S_1
$\mathcal{M}(\Gamma,$ `let` x `=` e_1 `in` $e_2, \rho) =$
  let $S_1 = \mathcal{M}(\Gamma, e_1, \beta)$, new β ($\mathcal{M}$.7)
    $S_2 = \mathcal{M}(S_1\Gamma + x{:}\,Clos_{S_1\Gamma}(S_1\beta), e_2, S_1\rho)$ ($\mathcal{M}$.8)
  in S_2S_1
$\mathcal{M}(\Gamma,$ `fix` $f\ \lambda x.e, \rho) = \mathcal{M}(\Gamma + f{:}\,\rho, \lambda x.e, \rho)$ ($\mathcal{M}$.9)

그림 2 타입 유추 알고리즘 W와 M. 모든 새 타입 변수들은 서로 구별된다. 각 재귀 호출 $W(\Gamma, e)$에서 발생한 새 타입 변수의 집합 V는 $V \cap ftv(\Gamma) = \emptyset$ 을 만족한다. 마찬가지로 각 재귀 호출 $M(\Gamma, e, \rho)$에서 발생한 새 타입 변수의 집합 V는 $V \cap (ftv(\Gamma) \cup ftv(\rho)) = \emptyset$ 을 만족한다.

앞의 예제에서 모든 타입 유추 방법들은 동일한 형태를 가지고 있다. 하위 프로그램의 타입을 유추하기 전 타입조건을 느슨하게 만들고 그 유추가 성공하고 나면 느슨한 타입 조건을 보상해 준다. 즉, 느슨한 타입 조건으로 유추한 결과를 원래의 최대 조건으로 검사해 준다.

제시할 일반화된 타입 유추 알고리즘은 타입 조건을 느슨하게 만들 수 있도록 하고, 느슨하게 만들어진 타입 조건으로 유추된 결과를 나중에 보상하도록 하였다. 각

$\mathcal{G}$: $TypEnv \times Expr \times Type \to Subst$

$\mathcal{G}(\Gamma, (), \rho) = \mathcal{U}(\rho, \iota)$ ($\mathcal{G}$.1)
$\mathcal{G}(\Gamma, x, \rho) = \mathcal{U}(\rho, \{\vec{\beta}/\vec{\alpha}\}\tau)$, new $\vec{\beta}$, $\Gamma(x) = \forall\vec{\alpha}.\tau$ ($\mathcal{G}$.2)
$\mathcal{G}(\Gamma, \lambda x.e, \rho) =$
  let $S_1 = \mathcal{U}(\beta_1 \to \beta_2, \theta)$, new β_1, new β_2, (1) $\theta \geq \rho$ ($\mathcal{G}$.3)
    $S_2 = \mathcal{G}(S_1\Gamma + x{:}\,S_1\beta_1, e, S_1\beta_2)$ ($\mathcal{G}$.4)
    $S_3 = \mathcal{U}(S_2S_1\theta, S_2S_1\rho)$ ($\mathcal{G}$.5)
  in $S_3S_2S_1$
$\mathcal{G}(\Gamma, e_1\ e_2, \rho) =$
  let $S_1 = \mathcal{G}(\Gamma, e_1, \theta_1)$, new β, (2) $\theta_1 \geq \beta \to \rho$ ($\mathcal{G}$.6)
    $S_2 = \mathcal{U}(S_1\theta_1, \theta_2)$, (3) $\theta_2 \geq S_1(\beta \to \rho)$ ($\mathcal{G}$.7)
    $S_3 = \mathcal{G}(S_2S_1\Gamma, e_2, \theta_3)$, (4) $\theta_3 \geq S_2S_1\beta$ ($\mathcal{G}$.8)
    $S_4 = \mathcal{U}(S_3S_2S_1\theta_1, S_3S_2S_1(\beta \to \rho))$ ($\mathcal{G}$.9)
    $S_5 = \mathcal{U}(S_4S_3\theta_3, S_4S_3S_2S_1\beta)$ ($\mathcal{G}$.10)
  in $S_5S_4S_3S_2S_1$
$\mathcal{G}(\Gamma,$ `let` x=e_1 `in` $e_2, \rho) =$
  let $S_1 = \mathcal{G}(\Gamma, e_1, \beta)$, new β ($\mathcal{G}$.11)
    $S_2 = \mathcal{G}(S_1\Gamma + x{:}\,Clos_{S_1\Gamma}(S_1\beta), e_2, \theta)$, (5) $\theta \geq S_1\rho$ ($\mathcal{G}$.12)
    $S_3 = \mathcal{U}(S_2\theta, S_2S_1\rho)$ ($\mathcal{G}$.13)
  in $S_3S_2S_1$
$\mathcal{G}(\Gamma,$ `fix` $f\ \lambda x.e, \rho) =$
  let $S_1 = \mathcal{G}(\Gamma + f{:}\,\theta_1, \lambda x.e, \theta_2)$, (6) $\theta_1 \wedge \theta_2 \geq \rho$ ($\mathcal{G}$.14)
    $S_2 = \mathcal{U}(S_1\theta_1, S_1\theta_2, S_1\rho)$ ($\mathcal{G}$.15)
  in S_2S_1

그림 3 일반화된 타입 유추 알고리즘 G. 각각의 $\theta \geq \rho$의 $ftv(\theta) \setminus ftv(\rho)$에 속하는 모든 타입 변수들은 새 타입 변수이다. 모든 타입 변수들은 서로 구별된다. 각 재귀 호출 $G(\Gamma, e, \rho)$에서 발생한 새 타입 변수의 집합 V는 $V \cap (ftv(\Gamma) \cup ftv(\rho)) = \emptyset$ 을 만족한다.

타입 유추 과정에서 하위 프로그램을 유추하기 전에 타입 조건을 느슨하게 만들 수 있다. 하위 프로그램의 유추가 끝난 후 느슨한 타입 조건을 동일화를 통하여 보상해 준다. 모든 하위 프로그램의 유추가 끝난 후에는 이전의 보상이 완전하지 않을 수도 있기 때문에 현재 프로그램의 유추가 끝나기 직전에 완전히 보상해 준다. 예를 들어 함수 적용문 $e_1\ e_2$를 ρ 조건으로 유추할 때, e_1을 $\beta \to \rho$보다 느슨한 조건으로 유추할 수 있다. 예를 들면 아무 조건 없이 유추할 수 있다. e_1의 유추가 끝난 뒤 보상해 주는데 완전히 보상해 줄 필요는 없다. 예를 들면 결과가 함수인지만 검사한다. e_2를 유추할 때도 역시 e_1의 인자 타입보다 느슨한 조건으로 유추한다. 이 모든 과정이 끝난 후 느슨한 조건들을 모두 동일화를 통해 보상해 준다. 즉, e_2의 타입이 e_1의 인자 타입이 되는지 검사하고, e_1 타입의 결과 타입이 ρ인지 검사해 준다.

2.2 알고리즘 정의

일반화된 타입 유추 알고리즘 G는 그림 3에 있다. M의 경우와 같이 프로그램, 타입 환경, 타입 조건을

입력으로 받아 타입 치환 함수를 결과로 준다. G의 결과로 얻어진 타입 치환 함수를 입력으로 준 타입 조건에 적용하여 얻어지는 타입이 입력 프로그램의 타입이 된다.

G에서 (1)에서 (6)까지 $\theta \geq \rho$ 형식으로 표기한 곳이 타입 조건을 느슨하게 만드는 과정이다. 각 과정에서 가능한 최대 타입 조건 ρ로부터 느슨한 θ 조건을 만들어 하위 프로그램에 넘겨준다. ρ로부터 얻어지는 느슨한 타입 조건 θ란 θ에 있는 타입 변수를 치환하여 ρ를 얻을 수 있는 타입 조건을 말한다.

정의 1 ($\theta \geq \rho$). *두 타입 θ, ρ에 대해 $Q\theta = \rho$와 $supp(Q) \subseteq ftv(\theta) \setminus ftv(\rho)$를 만족하는 타입 치환 함수 Q가 존재하면, θ가 ρ보다 느슨한 타입이라고 하고 $\theta \geq \rho$라고 표기한다. 또한, 타입 θ_1, θ_2, ρ에 대하여 $Q\theta_1 = Q\theta_2 = \rho$이고 $supp(Q) \subseteq (ftv(\theta_1) \cup ftv(\theta_2)) \setminus ftv(\rho)$를 만족하는 타입 치환 함수 Q가 있을 때 $\theta_1 \wedge \theta_2 \geq \rho$라 표기한다.*

변수의 경우(G.2), 변수 타입 $\Gamma(x)$는 주어진 타입 조건 ρ를 만족하여야 한다: $U(\rho, \{\vec{\beta}/\vec{\alpha}\}\tau)$. 상수의 경우 (G.1)도 비슷하다.

함수 $\lambda x.e$의 경우를 살펴보자. 주어진 타입 조건이 ρ일 때, 먼저 함수의 내용(function body) e를 유추할 타입 조건을 정해야 한다. 주어진 타입 조건 ρ와 같거나 느슨한 타입을 고른 후, 결정된 타입의 인자 타입과 결과 타입을 동일화를 통하여 얻는다.

$$S_1 = U(\beta_1 \to \beta_2, \theta), \text{ new } \beta_1, \beta_2 \quad (1)\ \theta \geq \rho \quad \text{(G.3)}$$

얻어낸 타입의 결과 타입을 함수 내용의 타입 조건으로 넘겨준다.

$$S_2 = G(S_1\Gamma + x : S_1\beta_1, e, S_1\beta_2) \quad \text{(G.4)}$$

예를 들어, θ를 새 타입 변수로 결정하면, (G.3)의 동일화는 효과가 없어 결국 함수 내용의 타입은 아무 조건 없이 유추된다. 만약, θ를 ρ로 결정하면 (G.3)는 주어진 조건 ρ가 함수인지 검사한 후 ρ의 결과 타입을 조건으로 하여 함수 내용을 유추한다. e에 대한 유추가 끝난 뒤 느슨하게 만든 타입 조건을 보상해 주어야 한다. 유추의 마지막 과정은 느슨한 타입 조건으로 유추한 타입이 원래의 타입 조건을 만족하는지 검사하는 것이다.

$$S_3 = U(S_2S_1\theta, S_2S_1\rho) \quad \text{(G.5)}$$

함수 적용(application) $e_1\ e_2$의 경우를 살펴보자. 주어진 타입 조건이 ρ일 때, 우선 함수 부분 e_1을 유추할 타입 조건을 결정해야 한다. 가장 정보가 많은 타입 조건 $\beta \to \rho$보다 느슨한 타입이면 된다. 여기서, β는 새 타입 변수이다.

$$S_1 = G(\Gamma, e, \theta_1), \text{ new } \beta, \ (2)\ \theta_1 \geq \beta \to \rho \quad \text{(G.6)}$$

함수 부분의 타입이 성공적으로 유추되고 나서 인자 부분의 타입을 유추하기 전에 타입 조건을 느슨하게 한 것을 보상해준다. 그러나, 완전히 보상해 줄 필요는 없다. 이 보상은 함수 부분의 유추된 타입에 기대하는 정도를 조정할 수 있도록 하여 이루어진다. 최대 기대 타입인 $\beta \to \rho$로 검사할 수도 있고 아무 것도 검사하지 않을 수도 있다. 이러한 조정 가능한 보상도 역시 $S_1(\beta \to \rho)$보다 느슨한 타입 θ_2를 골라서 e_1의 타입과 동일화하여 이루어진다.

$$S_2 = U(S_1\theta_1, \theta_2), \quad (3)\ \theta_2 \geq S_1(\beta \to \rho) \quad \text{(G.7)}$$

다음 순서로 인자 부분 e_2의 타입을 유추하기 위한 타입 조건을 결정한다. e_1의 인자타입인 β보다 느슨한 타입 θ_3을 타입 조건으로 e_2를 유추한다.

$$S_3 = G(S_2S_1\Gamma, e_2, \theta_3), \quad (4)\ \theta_3 \geq S_2S_1\beta \quad \text{(G.8)}$$

마지막으로 모든 느슨함을 두 번의 동일화를 통하여 보상해 준다. 하나는 느슨하게 만든 타입 조건 θ_1을 원래의 최대 타입 조건 $\beta \to \rho$로 검사하여 준다.

$$S_4 = U(S_3S_2S_1\theta_1, S_3S_2S_1(\beta \to \rho)) \quad \text{(G.9)}$$

또 하나는 느슨하게 만든 타입 조건 θ_3을 원래의 타입 조건, 즉, e_1의 인자 타입으로 검사한다.

$$S_5 = U(S_4S_3\theta_3, S_4S_3S_2S_1\beta) \quad \text{(G.10)}$$

θ_2는 이미 θ_1과 (G.7)에서 동일화했고 (G.9)에서 θ_1을 완전히 보상하였으므로 보상해 줄 필요가 없다.

let $x = e_1$ in e_2의 경우를 살펴보자. e_1에 줄 수 있는 타입 조건은 없으므로 조건 없이 e_1의 타입을 유추한다.

$$S_1 = G(\Gamma, e_1, \beta), \text{ new } \beta \quad \text{(G.11)}$$

다음으로 e_2를 유추하기 위한 타입 조건을 결정한다. 주어진 타입 조건보다 느슨한 타입이면 된다.

$$S_2 = G(S_1\Gamma + x : Clos_{S_1\Gamma}(S_1\beta), e_2, \theta), \ (5)\ \theta \geq S_1\rho \quad \text{(G.12)}$$

마지막으로 느슨한 타입 조건 θ를 원래의 타입 조건으로 보상해 준다.

$$S_3 = U(S_2\theta, S_2S_1\rho) \quad \text{(G.13)}$$

재귀 함수(recursive function) fix $f\ \lambda x.e$의 경우를 살펴보자. 먼저 $\lambda x.e$를 유추하기 위한 타입 조건과 f의 타입을 결정해야 한다. 두 타입 모두 주어진 타입 조건 ρ보다 느슨한 타입이면 된다.

$S_1 = G(\Gamma + f: \theta_1, \lambda x.e, \theta_2)$, (6) $\theta_1 \wedge \theta_2 \geq \rho$ (G.14)
유추에 성공하고 나서 느슨한 타입들을 원래의 타입 조건으로 보상해 준다.

$$S_2 = U(S_1\theta_1, S_1\theta_2, S_1\rho) \quad \text{(G.15)}$$

2.3 예 알고리즘

G에서 느슨한 타입 조건을 결정함으로써 다양한 타입 유추 알고리즘을 얻을 수 있다. 예 알고리즘에는 표준 알고리즘 W, 하향성 알고리즘 M, 혼성 알고리즘 H, 그리고 SML/NJ와 OCaml에서 사용된 혼성 타입 유추 알고리즘을 모두 포함한다.

- 모든 θ를 새 타입 변수로 결정하면 W 알고리즘이 된다.
- 모든 θ를 느슨하게 만들지 않으면, 즉, 최대 타입 조건으로 결정하면 M 알고리즘이 된다.
- (G.6)의 (2)에서 타입 조건을 새 함수 타입 ($\beta_1 \rightarrow \beta_2$, 단, β_1, β_2는 새 타입 변수)으로 하고 나머지는 느슨하게 하지 않으면 H 알고리즘이 된다.
- (G.6)의 (2)에서 타입 조건을 새 타입 변수로 하고 나머지는 느슨하게 하지 않으면 OCaml의 타입 유추 알고리즘이 된다.
- SML/NJ의 타입 유추 알고리즘은 다음과 같다. (G.3)의 (1)에서 함수가 재귀 함수일 때는 최대 타입 조건으로 결정하고, 아니면 새 타입 변수로 결정한다. (G.14)의 (6)에서 θ_1, θ_2를 같은 새 타입 변수로 결정한다. 나머지는 새 타입 변수로 결정한다.

위의 다섯 예 알고리즘의 타입 조건 θ를 그림 4에 요약하였다.

	(1) θ	(2) θ_1	(3) θ_2	(4) θ_3	(5) θ	(6) θ_1, θ_2
$\mathcal{W}$	β_1	β_1	β_2	β_3	β_1	β_1, β_2
SML/NJ's	β_1 or ρ	β_1	β_2	β_3	β_1	β_1, β_1
OCaml's	ρ	β_1	$S_1(\beta \rightarrow \rho)$	$S_2S_1\beta$	$S_1\rho$	ρ, ρ
$\mathcal{H}$	ρ	$\beta \rightarrow \beta_2$	$S_1(\beta \rightarrow \rho)$	$S_2S_1\beta$	$S_1\rho$	ρ, ρ
$\mathcal{M}$	ρ	$\beta \rightarrow \rho$	$S_1(\beta \rightarrow \rho)$	$S_2S_1\beta$	$S_1\rho$	ρ, ρ

그림 4 G의 다섯 예 알고리즘. β_i는 모두 새 타입 변수이다.

3. 안전성과 완전성

G의 모든 예 알고리즘은 Hindley/Milner 타입 체계에 대해 안전하고 완전하다.

정리 2(안전성). *프로그램 e, 타입 환경 Γ, 타입 ρ에 대해, $G(\Gamma, e, \rho)$가 성공하여 S를 결과로 주면 $S\Gamma \vdash e: S\rho$가 성립한다.*
증명. 부록 A에 있음.

안전성은 프로그램 e를 타입 조건 ρ로 G에 의해 타입 유추하였을 때 성공하여 결과 S를 주면, 타입 유추 규칙을 통하여 e가 타입 $S\rho$를 가진다는 것을 말한다.

정리 3(완전성). *프로그램 e, 타입 환경 Γ에 대해 $P\Gamma \vdash e: P\rho$를 만족하는 타입 치환 함수 P가 존재하면, $G(\Gamma, e, \rho)$는 성공하여 S를 결과로 주며, $P+_V = (RS)+_V$를 만족하는 타입 치환 함수 R이 존재한다. 여기서, V는 $G(\Gamma, e, \rho)$에서 발생한 새 타입 변수의 집합이다.*
증명. 부록 B에 있음.

완전성은 프로그램 e가 타입 조건 ρ를 만족하는 타입 τ를 가지면(즉, $\exists P.P\rho = \tau$), G 알고리즘이 성공하고 결과로 S를 주는데 $S\rho$가 τ보다 일반적이라는 의미이다(즉, $\exists R.RS\rho = \tau$).

4. 덜 느슨한 알고리즘이 신속히 오류를 감지한다

타입 조건에 실리는 정보의 양이 알고리즘이 얼마나 신속히 오류를 감지하는지를 결정한다. 보다 느슨한 타입 조건으로는 하위 프로그램을 성공적으로 타입 유추할 가능성이 높아지기 때문에 오류를 감지하는 시점이 연기된다.

G의 두 예 알고리즘의 어느 한쪽이 항상 느슨한 타입 조건으로 결정할 때 보다 느슨한 알고리즘이라고 한다. '항상'의 의미는 각각의 느슨한 타입 조건을 결정하는 과정에서, 최대 기대 타입이 더 느슨하다면 결정된 타입 조건 또한 더 느슨하다는 뜻이다. 즉, A의 타입 조건을 결정하는 $\theta_i \geq \rho_i$와 이에 상대되는 A'의 $\theta_i' \geq \rho_i'$에 대해 ρ_i'가 ρ_i보다 느슨하다면 θ_i' 또한 θ_i보다 느슨하게 할 때 A'를 보다 느슨한 알고리즘이라고 한다.

정의 2($A \sqsubset A'$). *A, A'이 G의 예 알고리즘이라고 하자. $A(\Gamma, e, \rho)$ 도중의 $\theta_i \geq \rho_i$와 이에 상대되는 $A'(\Gamma, e, \rho)$ 도중의 $\theta_i' \geq \rho_i'$ 사이에, 만약 $\rho_i = R\rho_i'$인 R이 존재할 때, $\theta_i = (R+_{subp(P)} \cup P)\theta_i'$이고*

supp$(P) \subseteq ftv(\theta_i') \setminus ftv(\rho_i')$ *를 만족하는 P가 존재하면, A'이 A보다 느슨한 알고리즘이라 하고 $A \sqsubseteq A'$라고 표기한다.*

그러면, 앞에 제시한 다섯 예 알고리즘들 사이에 느슨한 정도로 순서가 생긴다. W가 가장 느슨하고 M이 가장 느슨하지 않다.

보조정리 1. *$M \sqsubseteq H \sqsubseteq$ OCaml 알고리즘 $\sqsubseteq$ SML/NJ 알고리즘 $\sqsubseteq W$.*
증명. 부록 D에 있음.

타입 오류를 감지하는 시점은 호출 문자열(call string)[4]에 의해 엄밀히 살펴 볼 수 있다. $G(\Gamma, e, \rho)$의 호출 문자열은, $[\![G(\Gamma, e, \rho)]\!]$로 표기하며, 공백 문자열에서 시작해서, 매번 $G(\Gamma_1, e_1, \rho_1)$를 호출할 때는 $(\Gamma_1, e_1, \rho_1)^d$를 붙이고, 귀환할 때는 $(\Gamma_1, e_1, \rho_1)^u$를 붙여 만든 문자열을 의미한다. 위첨자 d와 u는 각각 스택 포인터가 아래쪽으로(downward) 또는 위쪽으로(upward) 이동하는 것을 의미한다. G의 모든 예 알고리즘의 호출 문자열은 유한함을 주목하라. 왜냐하면, G의 모든 예 알고리즘은 각 하위 프로그램을 단 한 번만 호출하기 때문이다. G의 모든 예 알고리즘의 호출/귀환 순서는 같다.

두 G의 예 알고리즘 A, A'에 대해 A가 A'보다 덜 느슨하면 타입 오류를 더 신속히 감지한다.

정리 4(상대적인 오류 감지의 신속성). *G의 두 예 알고리즘 A, A'에 대해 $A \sqsubseteq A'$이면, 타입 환경 Γ_0, 프로그램 e_0, 타입 ρ_0에 대해 $[\![A(\Gamma_0, e_0, \rho_0)]\!]$에 $(\Gamma, e, \rho)^{d/u}$가 있으면 $[\![A'(\Gamma_0, e_0, \rho_0)]\!]$에 $(\Gamma', e, \rho')^{d/u}$가 있고, $R\Gamma' > \Gamma$이고 $R\rho' = \rho$를 만족하는 타입 치환 함수 R이 존재한다.*
증명. 부록 E에 있음.

하위 프로그램을 호출하고 귀환하는 순서는 같으므로 위의 정리는 즉, A'가 A보다 느슨하면 A의 호출 문자열이 항상 같거나 짧다는 것을 의미한다. 호출 문자열이 짧다는 것은 더 일찍 멈춘다는 것을 의미하므로, 즉, 오류를 신속히 감지한다는 것이다.

보조정리 1과 정리 4에 의해, 제시된 다섯 예 알고리즘들 간에 오류를 신속히 감지하는 순서가 증명된다.

따름정리 1. *타입 환경 Γ, 프로그램 e, 타입 ρ에 대해*

$$\begin{aligned} |[\![M(\Gamma, e, \rho)]\!]| &\le |[\![H(\Gamma, e, \rho)]\!]| \\ &\le |[\![\text{OCaml's}(\Gamma, e, \rho)]\!]| \\ &\le |[\![\text{SML/NJ's}(\Gamma, e, \rho)]\!]| \\ &\le |[\![W(\Gamma, e, \rho)]\!]| \end{aligned}$$

가 성립한다. 여기서 $|s|$는 호출 문자열 s의 길이이다.

5. 결 론

두 알고리즘 W와 M은 Hindley/Milner let-다형성 타입 체계의 극단적 타입 유추 알고리즘이다. 사실상 표준인 W 알고리즘은 문맥에 상관없이 타입을 검사하므로 타입 오류를 늦게 감지한다. 반면에 하향성 M 알고리즘은 가능한 문맥을 모두 참고하여 타입을 검사하므로 타입 오류를 신속히 감지한다. 실제 컴파일러 시스템에서는 이러한 극단적인 행태를 피한 혼성 알고리즘이 필요하게 되는데 이를 위한 체계적인 방법론이 아직 연구되지 않았다.

본 연구에서는 타입 유추시 문맥을 참고하는 정도를 조절할 수 있도록 하여 일반화된 타입 유추 알고리즘을 제시하였다. W, M 뿐만 아니라 다양한 혼성 알고리즘을 제시된 알고리즘의 예로 만들어 낼 수 있고, 예 알고리즘은 모두 타입 체계를 안전하고 완전하게 구현함을 증명하였다. 또한 예 알고리즘이 다른 예 알고리즘보다 상대적으로 신속히 오류를 감지하는 조건을 제시하였다. 결국 예 알고리즘의 안전성, 완전성, 상대적인 오류 감지의 신속성이 자동적으로 이끌어 낼 수 있게 되었다. G 알고리즘의 예 알고리즘은 현재 널리 쓰이는 SML/NJ와 OCaml 컴파일러의 혼성 타입 유추 알고리즘까지 포함한다.

보다 일반화된 알고리즘을 만드는 방법은 타입 조건만 느슨하게 하는 것뿐만 아니라 타입 환경이나 타입 치환 함수까지 느슨하게 하는 방법이 있을 수 있다. McAdam의 알고리즘[11]은 G의 예 알고리즘이 아닌데, 하위 프로그램들의 타입 유추 중 결과로 발생하는 타입 치환 함수를 누적하지 않고 타입 유추를 하다가 모든 하위 프로그램들의 유추가 끝나야만 비로소 타입 치환 함수들이 일관성이 있는지 검사한다. G 알고리즘에서 그랬듯이 타입을 느슨하게 하는 것뿐만 아니라 누적되는 타입 치환 함수를 조절하게 하면 McAdam의 타입 유추 알고리즘까지 포함하는 일반화된 유추 알고리즘을 만들 수 있다. 이렇게 느슨한 타입 치환 함수는 G가 그랬듯이 반드시 나중에 보상을 해 주어야만 한다.

보다 일반적인 방법[12,13,14,15,16]으로 타입 유추 알고리즘을 두 단계를 나누어서 생각하는 방법이 있다. (1) 동일화 조건들을 이끌어 내는 단계와 (2) 그 조건들을 푸는 단계를 따로 생각하는 방법이다. 그 관점에서 보면, G 알고리즘은 동일화 조건을 푸는 순서를 정할 수 있는 조절 방법을 제시한 것이다. 타입 조건을 느슨하게 함으로써 동일화 조건을 푸는 것을 연기시키고 연기된 과정은 결국 나중에 보상 과정에서 풀도록 한 것이다.

부 록

A. 정리 2의 증명

정리 2(안전성). *프로그램 e, 타입 환경 Γ, 타입 ρ에 대해, $G(\Gamma, e, \rho)$가 성공하여 S를 결과로 주면 $S\Gamma \vdash e : S\rho$가 성립한다.*

다음의 보조정리들을 사용한다.

보조정리 2[2]. *$\Gamma \vdash e : \tau$ 이면 $S\Gamma \vdash e : S\tau$ 이다.*

보조정리 3[2]. *$\sigma > \sigma'$ 이면 $S\sigma > S\sigma'$ 이다.*

보조정리 4[1]. *타입 τ, 타입 환경 Γ, 타입 치환 함수 S에 대해, $\vec{\alpha} = ftv(\tau) \setminus ftv(\Gamma)$, $\vec{\beta}$는 새 타입 변수, $S' = S\{\vec{\beta}/\vec{\alpha}\}$라 하면, $SClos_\Gamma(\tau) = Clos_{S'\Gamma}(S'\tau)$ 이다.*

정리 2의 증명. e의 구조에 대한 귀납법으로 증명한다.

• **()의 경우**: $S\rho = S\iota = \iota$. 그러므로 (CON) 규칙에 의해 $S\Gamma \vdash () : S\rho$ 이 성립한다.

• **x의 경우**: 보조정리 3에 의해 $S\rho = S\{\vec{\beta}/\vec{\alpha}\}\tau < SI(x)$가 성립한다. 그러므로 (VAR) 규칙에 의해 $S\Gamma \vdash x : S\rho$가 성립한다.

• **$\lambda x.e$의 경우**: 귀납 가정에 의해 (G.4) 의해 $S_2S_1\Gamma + \lambda x.e : S_2S_1(\beta_1 \to \beta_2)$가 성립한다. 보조정리 2에 의해 양변에 S_3를 적용하면

$$S_3S_2S_1\Gamma \vdash \lambda x.e : S_3S_2S_1(\beta_1 \to \beta_2)$$

가 성립한다. (G.3)에 의해 $S_1(\beta_1 \to \beta_2) = S_1\theta$ 이고 (G.5)에 의해 $S_3S_2S_1\theta = S_3S_2S_1\rho$ 이므로, $S_3S_2S_1\Gamma \vdash \lambda x.e : S_3S_2S_1\rho$가 성립한다.

• **$e_1\ e_2$의 경우**: 귀납 가정에 의해 (G.6)은 $S_1\Gamma \vdash e_1 : S_1\theta_1$을 의미한다. 보조정리 2에 의해 $S_5S_4S_3S_2$을 양변에 취하면 $S_5S_4S_3S_2S_1\Gamma \vdash e_1 : S_5S_4S_3S_2S_1\theta_1$가 성립한다. (G.9)에 의해 $S_4S_3S_2S_1\theta_1 = S_4S_3S_2S_1(\beta \to \rho)$이고, (G.10)에 의해 $S_5S_4S_3S_2S_1\beta = S_5S_4S_3\theta_3$ 이므로,

$$S_5S_4S_3S_2S_1\Gamma \vdash e_1 : S_5S_4S_3(\theta_3 \to S_2S_1\rho) \quad (7)$$

가 성립한다. 귀납 가정에 의해 (G.8)은 $S_3S_2S_1\Gamma \vdash e_2 : S_3\theta_3$ 을 의미한다. 보조정리 2에 의해 양변에 S_5S_4를 적용하면

$$S_5S_4S_3S_2S_1\Gamma \vdash e_2 : S_5S_4S_3\theta_3 \quad (8)$$

가 된다. 그러므로 (7)과 (8)에 (APP)를 적용하면

$$S_5S_4S_3S_2S_1\Gamma \vdash e_1\ e_2 : S_5S_4S_3S_2S_1\rho$$

가 성립한다.

• **let $x = e_1$ in e_2의 경우**: $\vec{\alpha} = ftv(S_1\beta) \setminus ftv(S_1\Gamma)$이고 $\vec{\beta}$는 (G.11)에서 발생한 새 타입 변수들일 때, $S_2' = S_2\{\vec{\beta}/\vec{\alpha}\}$라 하자. 귀납 가정에 의해 (G.11)은 $S_1\Gamma \vdash e_1 : S_1\beta$이 성립한다. 보조정리 2에 의해 양변에 S_2'를 적용하면

$$S_2'S_1\Gamma \vdash e_1 : S_2'S_1\beta \quad (9)$$

가 성립한다. 귀납 가정에 의해 (G.12)는

$$S_2S_1\Gamma + x : S_2Clos_{S_1\Gamma}(S_1\beta) \vdash e_2 : S_2\theta \quad (10)$$

를 의미한다. S_2'와 S_2는 $S_1\Gamma$의 자유 타입 변수가 아닌 경우에만 다르므로 $S_2S_1\Gamma = S_2'S_1\Gamma$이고, 보조정리 4에 의해 $S_2Clos_{S_1\Gamma}(S_1\beta) = Clos_{S_2'S_1\Gamma}(S_2'S_1\beta)$가 성립한다. 그러므로, (10)은

$$S_2'S_1\Gamma + x : Clos_{S_2'S_1\Gamma}(S_2'S_1\beta) \vdash e_2 : S_2\theta \quad (11)$$

가 된다. (9)와 (11)에 (LET) 규칙을 적용하면 $S_2'S_1\Gamma \vdash$ let $x = e_1$ in $e_2 : S_2\theta$가 성립한다. 즉, $S_2S_1\Gamma \vdash$ let $x = e_1$ in $e_2 : S_2\theta$가 성립한다. 보조정리 2에 의해 양변에 S_3를 적용하면, $S_3S_2S_1\Gamma \vdash$ let $x = e_1$ in $e_2 : S_3S_2\theta$가 성립한다. (G.13)은 $S_3S_2\theta = S_3S_2S_1\rho$을 의미하므로

$$S_3S_2S_1\Gamma \vdash \text{let } x = e_1 \text{ in } e_2 : S_3S_2S_1\rho$$

가 성립한다.

• **fix $f\ \lambda x.e$의 경우**: 귀납 가정에 의해 (G.14)는 $S_1\Gamma + f : S_1\theta_1 \vdash \lambda x.e : S_1\theta_2$을 의미한다. 보조정리 2에 의해 양변에 S_2를 적용하면

$$S_2S_1\Gamma + f : S_2S_1\theta_1 \vdash \lambda x.e : S_2S_1\theta_2$$

가 성립한다. (G.15)에 의해 $S_2S_1\theta_1 = S_2S_1\theta_2 = S_2S_1\rho$ 이므로

$$S_2S_1\Gamma + f : S_2S_1\rho \vdash \lambda x.e : S_2S_1\rho$$

가 성립한다. 여기에 (FIX) 규칙을 적용하면 $S_2S_1\Gamma \vdash$ fix $f\ \lambda x.e : S_2S_1\rho$가 성립한다. □

B. 정리 3의 증명

정리 3(완전성). *프로그램 e, 타입 환경 Γ에 대해 $P\Gamma \vdash e : P\rho$를 만족하는 타입 치환 함수 P가 존재하면, $G(\Gamma, e, \rho)$는 성공하여 S를 결과로 주며,*

$P\dagger_V=(RS)\dagger_V$ *를 만족하는 타입 치환 함수* R *이 존재한다. 여기서,* V *는* $G(\Gamma,e,\rho)$ *에서 발생한 새 타입 변수의 집합이다.*

증명은 다음 보조정리들을 사용한다.

보조정리 5[4]. $SClos_\Gamma(\tau)>Clos_{S\Gamma}(S\tau)$.

보조정리 6[2]. $\Gamma>\Gamma'$ *인 타입 환경* Γ, Γ' *에 대해* $\Gamma'\vdash e:\tau$ *이면* $\Gamma\vdash e:\tau$ *이다.*

보조정리 7[1]. *타입 치환 함수* R, S, *타입* τ *에 대해, 다음이 성립한다.*

- $itv(RS)\subseteq itv(R)\cup itv(S)$
- $ftv(S\tau)\subseteq ftv(\tau)\cup itv(S)$

보조정리 8. $S=G(\Gamma,e,\rho)$ *이면* $itv(S)\subseteq ftv(\Gamma)\cup ftv(\rho)\cup V$ *이다. 단,* V *는* $G(\Gamma,e,\rho)$ *에서 발생한 새 타입 변수의 집합이다.*

증명. 부록 C에 있음. □

보조정리 9[4]. $itv(S)\cap V=\emptyset$ *이면* $(RS)\dagger_V=R\dagger_V S$ *이다.*

정리 3의 증명. e의 구조에 대한 귀납법으로 증명한다. 새 타입 변수에 대해 엄밀히 증명하기 위해 다음을 가정한다. G 알고리즘에서 발생한 모든 새 타입 변수는 다르다. 각각의 호출 $G(\Gamma,e,\rho)$ 에서 발생한 새 타입 변수들의 집합 V는 $V\cap(ftv(\Gamma)\cup ftv(\rho))=\emptyset$ 를 만족한다. $\theta\geq\rho$ 를 사용할 때마다 발생하는 $Q\theta=\rho$를 만족하는 타입 치환 함수 Q의 지원 변수 $supp(Q)=ftv(\theta)\setminus ftv(\rho)$ 는 모두 새 타입 변수이다.

- **() 와 x의 경우**: M의 증명과 같다. [4,5]
- **$\lambda x.e$의 경우**: 주어진 가정이 $P\Gamma\vdash\lambda x.e:\tau_1\to\tau_2$ 라 하자. 여기서, $\tau_1\to\tau_2=P\rho$ 이다. 그리고, $V=\{\beta_1,\beta_2\}\cup supp(Q)\cup V_1$ 라 하자. 여기서, β_1, β_2 는 (G.3)의 새 타입 변수이고, Q는 (G.3)의 $\theta\geq\rho$ 에 대한 타입 치환 함수이며, V_1 은 (G.4)의 $G(S_1\Gamma+x:S_1\beta_1,e,S_1\beta_2)$ 에서 발생한 새 타입 변수의 집합이다.

(G.3)의 $U(\beta_1\to\beta_2,\theta)$ 가 성공함을 증명하여야 한다. $P'=(PQ)\dagger_{\{\beta_1,\beta_2\}}\cup\{\tau_1/\beta_1,\tau_2/\beta_2\}$ 라 하자. 그러면, P' 은 $\beta_1\to\beta_2$ 와 θ를 동일화한다. 왜냐하면,

$$\begin{array}{lll} P'\theta &= PQ\theta & \beta_1,\beta_2\notin ftv(\theta)\text{이므로}\\ &= P\rho & Q\text{의 정의에 의해}\\ &= \tau_1\to\tau_2 & \text{가정에 의해}\\ &= P'(\beta_1\to\beta_2) & P'\text{의 정의에 의해.}\end{array}$$

이기 때문이다. 그러면 정리 1에 의해 (G.3)의 동일화는 성공하여 결과 S_1을 주며

$$R_1S_1=P' \tag{12}$$

을 만족하는 타입 치환 함수 R_1이 존재한다.

(FN) 규칙에 의해 주어진 가정은

$$P\Gamma+x:\tau_1\vdash e:\tau_2 \tag{13}$$

를 의미한다. (G.4)의 $G(S_1\Gamma+x:S_1\beta_1,e,S_1\beta_2)$과 (13)에 귀납 가정을 적용하기 위해 $\tau_2=P_1(S_1\beta_2)$와 $P\Gamma+x:\tau_1=P_1(S_1\Gamma+x:S_1\beta_1)$를 만족하는 P가 존재함을 보여야 한다. 그러한 P_1은 (12)의 R_1이다. 왜냐하면,

$$\begin{array}{lll} R_1(S_1\beta_2) &= P'\beta_2 & (12)\text{에 의해}\\ &= \tau_2 & P'\text{의 정의에 따라}\end{array}$$

$$\begin{array}{lll} & R_1(S_1\Gamma+x:S_1\beta_1) & \\ =& P'(\Gamma+x:\beta_1) & (12)\text{에 의해}\\ =& PQ\Gamma+x:\tau_1 & \beta_1,\beta_2\notin ftv(\Gamma)\text{이므로}\\ =& P\Gamma+x:\tau_1 & supp(Q)\cap ftv(\Gamma)=\emptyset\text{이므로}\end{array}$$

이기 때문이다. 그러면 귀납 가정에 의해 (G.4)의 $G(S_1\Gamma+x:S_1\beta_1,e,S_1\beta_2)$는 S_2를 결과로 성공하며 다음을 만족하는 R_2가 존재한다.

$$(R_2S_2)\dagger_{V_1}=R_1\dagger_{V_1} \tag{14}$$

그러면,

$$\begin{array}{ll} & itv(S_1)\\ \subseteq & \{\beta_1,\beta_2\}\cup ftv(\theta) \qquad \text{정리 1에 의해}\\ \subseteq & \{\beta_1,\beta_2\}\cup ftv(\rho)\cup supp(Q) \quad supp(Q)=ftv(\theta)\setminus ftv(\rho)\\ & \qquad\qquad\qquad\qquad\qquad\qquad \text{이므로}\end{array}$$

가 성립하므로 G의 정의에 의해

$$V_1\cap itv(S_1)=\emptyset \tag{15}$$

가 성립한다. 그러므로,

$$\begin{array}{lll} & (R_2S_2S_1)\dagger_{V_1} & \\ =& (R_2S_2)\dagger_{V_1}S_1 & \text{보조정리 9와 (15)에 의해}\\ =& R_1\dagger_{V_1}S_1 & (14)\text{에 의해}\\ =& (R_1S_1)\dagger_{V_1} & \text{보조정리 9와 (15)에 의해}\\ =& P'\dagger_{V_1} & (12)\text{에 의해}\end{array} \tag{16}$$

가 성립한다.

이제 (G.5)의 동일화 $U(S_2S_1\theta,S_2S_1\rho)$가 성공함을 증명할 차례이다.

$$\begin{array}{lll} R_2(S_2S_1\theta) &= P'\theta & (16)\text{과 } ftv(\theta)\cap V_1=\emptyset\text{이므로}\\ &= PQ\theta & \beta_1,\beta_2\notin ftv(\theta)\\ &= P\rho & Q\text{의 정의에 따라}\\ &= PQ\rho & ftv(\rho)\cap supp(Q)=\emptyset\text{이므로}\\ &= P'\rho & \beta_1,\beta_2\notin ftv(\rho)\\ &= R_2(S_2S_1\rho) & (16)\text{과 } ftv(\theta)\cap V_1=\emptyset\text{이므로}\end{array}$$

이므로, R_2는 $S_2S_1\theta$와 $S_2S_1\rho$을 동일화한다. 그러므로, (G.5)의 동일화는 S_3를 결과로 성공하며

$$R_3S_3=R_2 \tag{17}$$

를 만족하는 타입 치환 함수 R_3이 존재한다.

결국, $G(\Gamma,\lambda x.e,\rho)$는 $S_3S_2S_1$를 결과로 성공하며 $(R_3S_3S_2S_1)\dagger_V=P\dagger_V$이다. 왜냐하면,

$(R_3S_3S_2S_1)\dagger_V$
$= (R_2S_2S_1)\dagger_V$ (17)에 의해
$= P'\dagger_V$ (16)에 의해
$= P\dagger_V$ $supp(Q) \cup \{\beta_1, \beta_2\} \subseteq V$이므로

가 성립하기 때문이다.

• $e_1\ e_2$ **의 경우**: 주어진 가정이 $P\Gamma \vdash e_1\ e_2 : P\rho$ 이고

$V = \{\beta\} \cup supp(Q_1) \cup supp(Q_2) \cup supp(Q_3) \cup V_1 \cup V_2$

라 하자. 여기서 β는 (G.6)의 새 타입 변수이고, Q_1, Q_2, Q_3는 각각 (G.6)의 $\theta_1 \geq \beta \to \rho$, (G.7)의 $\theta_2 \geq S_1(\beta \to \rho)$, (G.8)의 $\theta_3 \geq S_2S_1\beta$의 타입 치환 함수이고, V_1, V_2는 (G.6)의 $G(\Gamma, e_1, \theta_1)$와 (G.8)의 $G(S_2S_1\Gamma, e_2, \theta_3)$에서 발생한 새 타입 변수들의 집합이다. (APP) 규칙을 적용하면 어떤 타입 τ에 대해

$$P\Gamma \vdash e_1 : \tau \to P\rho \tag{18}$$

$$P\Gamma \vdash e_2 : \tau \tag{19}$$

가 성립한다.

먼저 귀납 가정에 의해 (G.6)의 $G(\Gamma, e_1, \theta_1)$이 성공함을 증명한다. $P' = P\dagger_{(\beta)} \cup \{\tau/\beta\}$ 라 하자. 그러면

$P'Q_1\theta_1 = P'(\beta \to \rho)$ Q_1의 정의에 따라
$= \tau \to P\rho$ $\beta \notin ftv(\rho)$

이고, $ftv(\Gamma) \cap (supp(Q_1) \cup \beta) = \emptyset$ 이므로 $P'Q_1\Gamma = P\Gamma$가 성립한다. 그러므로, (G.6)의 $G(\Gamma, e_1, \theta_1)$과 (18)에 귀납 가정을 적용하면

$$(R_1S_1)\dagger_{V_1} = (P'Q_1)V_1 \tag{20}$$

가 성립한다.

그러면 R_1Q_2가 (G.7)의 $S_1\theta_1$와 θ_2을 동일화한다. 왜냐하면,

$ftv(S_1\theta_1) \cap supp(Q_2)$
$\subseteq (itv(S_1) \cup ftv(\theta_1)) \cap supp(Q_2)$ 보조정리 7에 의해
$\subseteq (ftv(\Gamma) \cup V_1 \cup ftv(\theta_1)) \cap supp(Q_2)$ 보조정리 8에 의해
$= \emptyset$ (21)

이므로

$R_1Q_2(S_1\theta_1)$
$= R_1S_1\theta_1$ (21)에 의해
$= P'Q_1\theta_1$ $ftv(\theta_1) \cap V_1 = \emptyset$이므로 (20)에 의해
$= P'(\beta \to \rho)$ Q_1의 정의에 따라
$= P'Q_1(\beta \to \rho)$ $ftv(\beta \to \rho) \cap supp(Q_1) = \emptyset$이므로
$= R_1S_1(\beta \to \rho)$ $ftv(\beta \to \rho) \cap V_1 = \emptyset$이므로 (20)에 의해
$= R_1Q_2(\theta_2)$ Q_2의 정의에 따라

가 성립하기 때문이다. 그러므로, 정리 1에 의해, (G.7)의 동일화는 S_2를 결과로 성공하며 $R_2S_2 = R_1Q_2$를 만족하는 R_2가 존재한다. 그러면,

$(R_2S_2S_1)\dagger_{supp(Q_2) \cup V_1}$
$= (R_1Q_2S_1)\dagger_{supp(Q_2) \cup V_1}$
$= (R_1S_1)\dagger_{supp(Q_2) \cup V_1}$ $supp(Q_2) \cap itv(S_1) = \emptyset$ 이므로 보조정리 8에 의해
$= (P'Q_1)\dagger_{supp(Q_2) \cup V_1}$ (20)에 의해 (22)

가 성립한다.

(G.8)의 $G(S_2S_1\Gamma, e_2, \theta_3)$와 (19)에 귀납 가정을 적용하기 위하여, $P_1(S_2S_1\Gamma) = P\Gamma$와 $P_1\theta_3 = \tau$를 만족하는 P_1이 존재함을 보여야 한다. R_2Q_3가 그러한 P_1이다. 이에 대한 증명은 다음과 같다. 우선,

$ftv(S_2S_1\Gamma)$
$\subseteq itv(S_2) \cup itv(S_1) \cup ftv(\Gamma)$ 보조정리 7에 의해
$\subseteq ftv(\theta_2) \cup ftv(\theta_1) \cup V_1 \cup ftv(\Gamma)$ 정리 1과 보조정리 8에 의해

이므로

$$supp(Q_3) \cap ftv(S_2S_1\Gamma) = \emptyset \tag{23}$$

가 성립한다. 그러면,

$R_2Q_3(S_2S_1\Gamma)$
$= R_2S_2S_1\Gamma$ (23)에 의해
$= P'Q_1\Gamma$ $ftv(\Gamma) \cap (supp(Q_2) \cup V_1) = \emptyset$ 이므로 (22)에 의해
$= P\Gamma$ $ftv(\Gamma) \cap (\{\beta\} \cup supp(Q_1)) = \emptyset$이므로

$R_2Q_3(\theta_3)$
$= R_2S_2S_1\beta$ Q_3의 정의에 따라
$= P'Q_1\beta$ $\beta \notin supp(Q_2) \cup V_1$이므로 (22)에 의해
$= P'\beta$ $\beta \notin supp(Q_1)$
$= \tau$ P'의 정의에 따라

가 성립한다. 그러면 귀납 가정에 의해 (G.8)은 S_3를 결과로 성공하며

$$(R_3S_3)\dagger_{V_3} = (R_2Q_3)\dagger_{V_2} \tag{24}$$

를 만족하는 R_3가 존재한다. 게다가

$$(R_3S_3)\dagger_{V_2 \cup supp(Q_3)} = R_2\dagger_{V_2 \cup supp(Q_3)} \tag{25}$$

를 만족한다.

그러면 R_3가 (G.9)의 $S_3S_2S_1\theta_1$과 $S_3S_2S_1(\beta \to \rho)$를 동일화한다.

$R_3S_3S_2S_1\theta_1$
$= R_2S_2S_1\theta_1$ $ftv(\theta_1) \cap (V_2 \cup supp(Q_3)) = \emptyset$이므로 (25)에 의해
$= P'Q_1\theta_1$ $ftv(\theta_1) \cap (V_1 \cup supp(Q_2)) = \emptyset$이므로 (22)에 의해
$= P'(\beta \to \rho)$ Q_1의 정의에 따라
$= P'Q_1(\beta \to \rho)$ $ftv(\beta \to \rho) \cap supp(Q_1) = \emptyset$이므로
$= R_2S_2S_1(\beta \to \rho)$ $ftv(\beta \to \rho) \cap (V_1 \cup supp(Q_2)) = \emptyset$이므로 (22)에 의해
$= R_3S_3S_2S_1(\beta \to \rho)$ $ftv(\beta \to \rho) \cap (V_2 \cup supp(Q_3)) = \emptyset$이므로 (25)에 의해.

그러므로, 정리 1에 의해, (G.9)의 동일화는 S_4를 결과로 성공하며

$$R_4S_4 = R_3 \tag{26}$$

를 만족하는 R_4가 존재한다.

마지막으로 R_4는 (G.10)의 $S_4S_3\theta_3$와 $S_4S_3S_2S_1\beta$를 동일화한다:

$R_4(S_4S_3\theta_3)$
$= R_3S_3\theta_3$ (26)에 의해
$= R_2Q_3\theta_3$ $ftv(\theta_3) \cap V_2 = \emptyset$이므로 (24)에 의해
$= R_2S_2S_1\beta$ Q_3의 정의에 따라
$= R_4(S_4S_3S_2S_1\beta)$ $\beta \notin V_2 \cup supp(Q_3)$이므로 (25), (26)에 의해.

그러므로, (G.10)의 동일화는 S_5를 결과로 성공하고

$$R_5S_5 = R_4 \tag{27}$$

를 만족하는 R_5가 존재한다. 결론적으로 $G(\Gamma, e_1\ e_2, \rho)$는 $S_5S_4S_3S_2S_1$를 결과로 성공한다.

이제 남은 것은 $(R_5S_5S_4S_3S_2S_1)\dagger_V = P\dagger_V$를 증명하는 것이다. 보조정리 7, 8과 정리 1에 의해 $itv(S_2S_1) \subseteq ftv(\Gamma) \cup ftv(\theta_1) \cup ftv(\theta_2) \cup V_1$가 성립하므로 알고리즘의 정의에 의해

$$itv(S_2S_1) \cap (V_2 \cup supp(Q_3)) = \emptyset \tag{28}$$

가 성립한다. 그러므로,

$(R_5S_5S_4S_3S_2S_1)\dagger_V$	
$= (R_4S_4S_3S_2S_1)\dagger_V$	(27)에 의해
$= (R_3S_3S_2S_1)\dagger_V$	(26)에 의해
$= ((R_3S_3)\dagger_{V_2 \cup supp(Q_3)}S_2S_1)\dagger_V$	보조정리 9와 (28)에 의해
$= (R_2\dagger_{V_2 \cup supp(Q_3)}S_2S_1)\dagger_V$	(25)에 의해
$= (R_2S_2S_1)\dagger_V$	보조정리 9와 (28)에 의해
$= (P'Q_1)\dagger_V$	(22)에 의해
$= P\dagger_V$	$\{\beta\} \cup supp(Q_1) \subseteq V$이므로

가 성립한다.

• let $x = e_1$ in e_2의 경우: 주어진 가정이 $P\Gamma \vdash$ let $x = e_1$ in $e_2 : P\rho$이고, $V = \{\beta\} \cup supp(Q) \cup V_1 \cup V_2$라 하자. 여기서 β는 (G.11)의 새 타입 변수이고 Q는 (G.12)의 $\theta \geq S_1\rho$의 타입 치환 함수이고, V_1, V_2는 각각 (G.11)의 $G(\Gamma, e_1, \beta)$와 (G.12)의 $G(S_1\Gamma + x : Clos_{S_1\Gamma}(S_1\beta), e_2, \theta)$에서 발생한 새 타입 변수의 집합이다. (LET) 규칙에 의해 어떤 타입 τ에 대해

$$P\Gamma \vdash e_1 : \tau \tag{29}$$

$$P\Gamma + x : Clos_{P\Gamma}(\tau) \vdash e_2 : P\rho \tag{30}$$

가 성립한다. $P' = P\dagger_{\{\beta\}} \cup \{\tau/\beta\}$라 하자. 그러면, $\beta \notin ftv(\Gamma)$이므로 $P'\beta = \tau$이고 $P'\Gamma = P\Gamma$이다. 그러므로 (G.11)의 $G(\Gamma, e_1, \beta)$와 (29)에 귀납 가정을 적용하면

$$(R_1S_1)\dagger_{V_1} = P'\dagger_{V_1} \tag{31}$$

를 만족하는 R_1이 존재한다. 그러면

$R_1Q(S_1\Gamma)$	
$= R_1S_1\Gamma$	$supp(Q) \cap ftv(S_1\Gamma) = \emptyset$이므로 보조정리 7,8에 의해
$= P'\Gamma$	$ftv(\Gamma) \cap V_1 = \emptyset$이므로 (31)에 의해
$= P\Gamma$	$\beta \notin ftv(\Gamma)$이므로

$R_1Q(Clos_{S_1\Gamma}(S_1\beta))$	
$> Clos_{R_1QS_1\Gamma}(R_1QS_1\beta)$	보조정리 5에 의해
$= Clos_{P\Gamma}(R_1S_1\beta)$	$supp(Q) \cap ftv(S_1\beta) = \emptyset$이므로 보조정리 7,8에 의해
$= Clos_{P\Gamma}(P'\beta)$	$\beta \notin V_1$이므로 (31)에 의해
$= Clos_{P\Gamma}(\tau)$	P'의 정의에 따라

가 성립한다. 즉, $R_1Q(S_1\Gamma + x : Clos_{S_1\Gamma}(S_1\beta)) > P\Gamma + x : Clos_{P\Gamma}(\tau)$가 성립한다. 그러면, 보조정리 6과 (30)에 의해

$$R_1Q(S_1\Gamma + x : Clos_{S_1\Gamma}(S_1\beta)) \vdash e_2 : P\rho \tag{32}$$

가 성립한다.

(G.12)의 $G(S_1\Gamma + x : Clos_{S_1\Gamma}(S_1\beta), e_2, \theta)$과 (32)에 귀납 가정을 적용하기 위하여 $R_1Q\theta = P\rho$을 증명하여야 한다:

$R_1Q(\theta)$	$=$	$R_1S_1\rho$ Q의 정의에 의해
	$=$	$P'\rho$ $ftv(\rho) \cap V_1 = \emptyset$이므로 (31)에 의해
	$=$	$P\rho$ $\beta \notin ftv(\rho)$이므로.

그러므로, 귀납 가정에 의하여 (G.12)의 $G(S_1\Gamma + x : Clos_{S_1\Gamma}(S_1\beta), e_2, \theta)$는 S_2를 결과로 성공하고

$$(R_2S_2)\dagger_{V_2} = (R_1Q)\dagger_{V_2} \tag{33}$$

를 만족하는 R_2가 존재한다. 게다가 R_2는 다음을 만족한다.

$$(R_2S_2)\dagger_{supp(Q) \cup V_2} = R_1\dagger_{supp(Q) \cup V_2} \tag{34}$$

그러면, R_2는 (G.13)의 $S_2\theta$와 $S_2S_1\rho$를 동일화한다:

$R_2(S_2\theta)$	$=$	$R_1Q\theta$ $ftv(\theta) \cap V_2 = \emptyset$이므로 (33)에 의해
	$=$	$R_1S_1\rho$ Q의 정의에 따라
	$=$	$R_2(S_2S_1\rho)$ 보조정리 7,8에 의해 $ftv(S_1\rho) \cap (supp(Q) \cup V_2) = \emptyset$ 이므로 (34)에 의해.

그러므로 (G.13)의 동일화는 S_3를 결과로 성공하며

$$R_3S_3 = R_2 \tag{35}$$

를 만족하는 R_3가 존재한다. 결론적으로 $G(\Gamma,$ let $x = e_1$ in $e_2, \rho)$는 $S_3S_2S_1$를 결과로 성공한다.

마지막으로 $(R_3S_3S_2S_1)\dagger_V = P\dagger_V$만 증명하면 된다. 보조정리 8에 의해 $itv(S_1) \subseteq ftv(\Gamma) \cup \beta \cup V_1$이므로 Q의 정의에 따라

$$itv(S_1) \cap (supp(Q) \cup V_2) = \emptyset \tag{36}$$

가 성립한다. 그러므로,

$(R_3S_3S_2S_1)\dagger_V$	
$= (R_2S_2S_1)\dagger_V$	(35)에 의해
$= ((R_2S_2)\dagger_{supp(Q) \cup V_2}S_1)\dagger_V$	보조정리 9와 (36)에 의해
$= (R_1\dagger_{supp(Q) \cup V_2}S_1)\dagger_V$	(34)에 의해
$= (R_1S_1)\dagger_V$	보조정리 9와 (36)에 의해
$= P'\dagger_V$	(31)에 의해
$= P\dagger_V$	$\beta \in V$이므로

가 성립한다.

• fix $f\ \lambda x.e$의 경우: 주어진 가정이 $P\Gamma \vdash$ fix $f\ \lambda x.e : P\rho$이고 $V = supp(Q) \cup V'$이라 하자. 여기서 Q는 (G.14)의 $\theta_1 \wedge \theta_2 \geq \rho$의 타입 치환 함수이고, V'은 (G.14)의 $G(\Gamma + f : \theta_1, \lambda x.e, \theta_2)$에서 발생한 새 타입 변수의 집합이다. (FIX) 규칙에 의해 $P\Gamma + f : P\rho \vdash \lambda x.e : P\rho$

가 성립한다. $supp(Q) \cap ftv(\Gamma) = \emptyset$ 이므로

$$PQ\Gamma + f: PQ\theta_1 \vdash \lambda x.e: PQ\theta_2$$

가 성립한다. (G.14)의 $G(\Gamma + f: \theta_1, \lambda x.e, \theta_2)$ 에 귀납 가정을 적용하면 S_1 을 결과로 성공하며

$$(R_1 S_1)\dagger_{V'} = (PQ)\dagger_{V'} \quad (37)$$

를 만족하는 R_1 이 존재한다. 그러면, R_1 이 (G.15)의 $S_1\theta_1$, $S_1\theta_2$, $S_1\rho$ 를 동일화한다:

$R_1(S_1\rho)$
$= PQ\rho$ — $ftv(\rho) \cap V' = \emptyset$ 이므로 (37)에 의해
$= P\rho$ — $ftv(\rho) \cap supp(Q) = \emptyset$ 이므로
$= PQ(\theta_1$ 또는 $\theta_2)$ — Q의 정의에 따라
$= R_1(S_1\theta_1$ 또는 $S_1\theta_2)$ — $(ftv(\theta_1) \cup ftv(\theta_2)) \cap V' = \emptyset$ 이므로 (37)에 의해.

그러므로 (G.15)의 동일화는 S_2 를 결과로 성공하고

$$R_2 S_2 = R_1 \quad (38)$$

를 만족하는 R_1 이 존재한다. 결론적으로 $G(\Gamma,$ fix $f\ \lambda x.e, \rho)$ 는 S_2S_1 을 결과로 성공하고 $(R_2S_2S_1)\dagger_V = P\dagger_V$ 를 만족한다. 왜냐하면,

$(R_2S_2S_1)\dagger_V = (R_1S_1)\dagger_V$ (38)에 의해
$= (PQ)\dagger_V$ (37)에 의해
$= P\dagger_V$ $supp(Q) \subseteq V$ 이므로

이기 때문이다. □

C. 보조정리 8의 증명

e 의 구조에 대한 귀납법으로 증명한다.

• **() 의 경우**: 정리 1에 의해 $itv(U(\rho, \iota)) \subseteq ftv(\rho) \cup ftv(\iota) = ftv(\rho)$가 성립한다.

• **x 의 경우**:

$itv(U(\rho, \{\vec{\beta}/\vec{\alpha}\}\tau))$
$\subseteq ftv(\rho) \cup ftv(\{\vec{\beta}/\vec{\alpha}\}\tau)$ 정리 1에 의해
$\subseteq ftv(\rho) \cup (ftv(\tau) \setminus \vec{\alpha}) \cup \vec{\beta}$
$= ftv(\rho) \cup ftv(\forall \vec{\alpha}.\tau) \cup \vec{\beta}$
$= ftv(\rho) \cup ftv(\Gamma(x)) \cup \vec{\beta}$ $\Gamma(x) = \forall \vec{\alpha}.\tau$ 이므로
$\subseteq ftv(\rho) \cup ftv(\Gamma) \cup \vec{\beta}$.

$\vec{\beta}$ 는 $G(\Gamma, x, \rho)$ 에서 발생한 새 타입 변수이다.

• **$\lambda x.e$ 의 경우**: Q 를 (G.3)의 $\theta \geq \rho$ 의 타입 치환 함수라 하자. 가정에 의해 $supp(Q)$ 는 모두 새 변수이다.

$itv(S_1) \subseteq ftv(\theta) \cup ftv(\beta_1 \to \beta_2)$
정리 1에 의해
$\subseteq ftv(\rho) \cup supp(Q) \cup \{\beta_1, \beta_2\}$
$supp(Q) = ftv(\theta) \setminus ftv(\rho)$이므로,

$itv(S_2) \subseteq ftv(S_1\Gamma) \cup ftv(S_1\beta_1) \cup ftv(S_1\beta_2) \cup V_1$
귀납 가정에 의해
$\subseteq itv(S_1) \cup ftv(\Gamma) \cup \{\beta_1, \beta_2\} \cup V_1$
보조정리 7에 의해

가 성립한다. 여기서, V_1 은 (G.4)의 $G(S_1\Gamma + x: S_1\beta_1, e, S_1\beta_2)$ 에서 발생한 새 타입 변수이다. 그리고,

$itv(S_3)$
$\subseteq ftv(S_2S_1\theta) \cup ftv(S_2S_1\rho)$ 정리 1에 의해
$\subseteq itv(S_2) \cup itv(S_1) \cup ftv(\theta) \cup ftv(\rho)$ 보조정리 7에 의해
$\subseteq itv(S_2) \cup itv(S_1) \cup supp(Q) \cup ftv(\rho)$

가 성립한다. 그러므로, $itv(S_3S_2S_1) \subseteq ftv(\Gamma) \cup ftv(\rho) \cup (supp(Q) \cup \{\beta_1, \beta_2\} \cup V_1)$ 가 성립한다. 여기서, $supp(Q) \cup \{\beta_1, \beta_2\} \cup V_1$은 $G(\Gamma, \lambda x.e, \rho)$에서 발생한 새 타입 변수이다.

다른 경우도 비슷하게 증명된다. □

D. 보조정리 1의 증명

보조정리 1. *$M \sqsubseteq H \sqsubseteq$ OCaml 알고리즘 $\sqsubseteq$ SML/NJ 알고리즘 $\sqsubseteq W$.*

증명. 바로 인접한 예 알고리즘들간에 증명하면 된다.

• **$M \sqsubseteq H$ 의 증명**: 두 알고리즘은 (G.6)의 (2)에서만 다르다. M은 $\beta \to \rho \geq \beta \to \rho$ 이고 H는 $\beta' \to \beta_2' \geq \beta' \to \rho'$이다. $R(\beta' \to \rho') = \beta \to \rho$ 를 만족하는 R이 존재한다고 가정하자. 그러면, $(R\dagger_{\{\beta_2'\}} \cup \{\rho/\beta_2'\})(\beta' \to \beta_2') = R\beta' \to \rho = \beta \to \rho$ 가 성립한다.

• **$H \sqsubseteq$ OCaml 알고리즘의 증명**: 두 알고리즘은 (G.6)의 (2)에서만 다르다. H는 $\beta \to \beta_2 \geq \beta \to \rho$ 이고, OCaml의 알고리즘은 $\beta_1' \geq \beta' \to \rho'$ 이다. 어떤 타입 치환 함수 R에 대해서도 $(R\dagger_{\{\beta_1'\}} \cup \{\beta \to \beta_2/\beta_1'\})\beta_1' = \beta \to \beta_2$이 성립한다.

• **OCaml 알고리즘 $\sqsubseteq$ SML/NJ 알고리즘의 증명**:

- (G.3)의 (1)의 경우: OCaml 알고리즘은 $\rho \geq \rho$ 이고 SML/NJ 알고리즘은 $\beta_1' \geq \rho'$ 또는 $\rho' \geq \rho'$ 이다. $R\rho' = \rho$ 를 만족하는 타입 치환 함수 R이 존재한다고 가정하자. 그러면, $(R\dagger_{\{\beta_1'\}} \cup \{\rho/\beta_1'\})\beta_1' = \rho$ 또는 $(R \cup \{\})\rho' = \rho$ 가 성립한다.

- (G.6)의 (2)의 경우: OCaml 알고리즘은 $\beta_1 \geq \rho$ 이고 SML/NJ 알고리즘은 $\beta_1' \geq \rho'$ 이다. 어떤 R이든 $(R\dagger_{\{\beta_1'\}} \cup \{\beta_1/\beta_1'\})\beta_1' = \beta_1$ 이 성립한다.

- (G.7)의 (3)의 경우: OCaml 알고리즘은 $S_1(\beta \to \rho) \geq S_1(\beta \to \rho)$ 이고 SML/NJ 알고리즘은 $\beta_2' \geq S_1'(\beta' \to \rho')$ 이다. 어떤 R이든 $(R\dagger_{\{\beta_2'\}} \cup \{S_1(\beta \to \rho)/\beta_2'\})\beta_2' = S_1(\beta \to \rho)$ 을 만족한다.

- (G.8)의 (4)의 경우: OCaml 알고리즘은 $S_2S_1\beta \geq S_2S_1\beta$ 이고 SML/NJ 알고리즘은 $\beta_3' \geq S_2'S_1'\beta'$ 이다. 어떤 R이든 $(R\dagger_{\{\beta_3'\}} \cup \{S_2S_1\beta/\beta_3'\})\beta_3' = S_2S_1\beta$ 를 만족한다.

- (G.12)의 (5)의 경우: OCaml 알고리즘은

$S_1\rho \geq S_1\rho$ 이고 SML/NJ 알고리즘은 $\beta_1' \geq S_1'\rho'$ 이다. 어떤 R 이든 $(R+_{\{\beta_1'\}} \cup \{S_1\rho/\beta_1'\})\beta_1' = S_1\rho$ 를 만족한다.

- (G.14)의 (6)의 경우: OCaml 알고리즘은 $\rho \wedge \rho \geq \rho$ 이고 SML/NJ 알고리즘은 $\beta_1' \wedge \beta_1' \geq \rho'$ 이다. 어떤 R 이든 $(R+_{\{\beta_1'\}} \cup \{\rho/\beta_1'\})\beta_1' = \rho$ 를 만족한다.

• SML/NJ 알고리즘 $\sqsubseteq W$의 증명:

- (G.3)의 (1)의 경우: SML/NJ 알고리즘은 $\beta_1 \geq \rho$ 또는 $\rho \geq \rho$ 이고, W는 $\beta_1' \geq \rho'$ 이다. 어떤 R 이든 $(R+_{\{\beta_1'\}} \cup \{\beta_1/\beta_1'\})\beta_1' = \beta_1$ 또는 $(R+_{\{\beta_1'\}} \cup \{\rho/\beta_1'\})\beta_1' = \rho$ 를 만족한다.

- (G.14)의 (6)의 경우: SML/NJ 알고리즘은 $\beta_1 \wedge \beta_1 \geq \rho$ 이고 W는 $\beta_1' \wedge \beta_2' \geq \rho'$ 이다. 어떤 R 이든 $(R+_{\{\beta_1',\beta_2'\}} \cup \{\beta_1/\beta_1', \beta_1/\beta_2'\})(\beta_1' \text{ 또는 } \beta_2') = \beta_1$을 만족한다.

- 나머지 경우: SML/NJ 알고리즘은 어떤 타입 τ에 대해 $\beta_i \geq \tau$ 이고 W는 어떤 타입 τ' 에 대해 $\beta_i' \geq \tau'$ 이다. 어떤 R 이든 $(R+_{\{\beta_i'\}} \cup \{\beta_i/\beta_i'\})\beta_i' = \beta_i$을 만족한다. □

E. 정리 4의 증명

정리 4(상대적인 오류 감지의 신속성). *G의 두 예 알고리즘 A, A'에 대해 $A \sqsubseteq A'$ 이면, 타입 환경 Γ_0, 프로그램 e_0, 타입 ρ_0에 대해 $[\![A(\Gamma_0, e_0, \rho_0)]\!]$에 $(\Gamma, e, \rho)^{d/u}$가 있으면 $[\![A'(\Gamma_0, e_0, \rho_0)]\!]$에 $(\Gamma', e, \rho')^{d/u}$가 있고, $R\Gamma' > \Gamma$이고 $R\rho' = \rho$를 만족하는 타입 치환 함수 R이 존재한다.*

증명은 다음의 보조정리들을 사용한다.

보조정리 10[4]. *$\Gamma > \Gamma'$ 이면 $Clos_\Gamma(\tau) > Clos_{\Gamma'}(\tau)$ 이다.*

보조정리 11. *G의 두 예 알고리즘 A, A', 타입 환경 Γ, Γ', 타입 ρ, ρ'에 대해 $R\Gamma' > \Gamma$와 $R\rho' = \rho$를 만족하는 R이 존재한다고 하자. $A(\Gamma, e, \rho)$가 S를 결과로 성공한다면, $A'(\Gamma', e, \rho')$ 또한 S'을 결과로 성공하며 $(R'S')+_V = (SR)+_V$를 만족하는 R'이 존재한다. 여기서, V는 $A(\Gamma, e, \rho)$가 사용한 새 타입 변수들의 집합이다.*

증명. $A(\Gamma, e, \rho)$가 S를 결과로 성공했으므로 안전성에 의해 $S\Gamma \vdash e : S\rho$가 성립한다. 보조정리 3에 의해 $SR\Gamma' > S\Gamma$이고 $S\rho = SR\rho'$이므로, 보조정리 6에 의해 $SR\Gamma' \vdash e : SR\rho'$가 성립한다. 그러면 완전성에 의해 $A'(\Gamma', e, \rho')$은 S'를 결과로 성공하며 $(R'S')+_V = (SR)+_V$를 만족하는 R'이 존재한다. □

정리 4의 증명. 호출 문자열 $[\![A(\Gamma_0, e_0, \rho_0)]\!]$의 접두 문자열(prefix)의 길이에 대해 귀납적으로 증명한다. $A'(\Gamma_0, e_0, \rho_0)$에서 사용된 모든 이름에 위첨자 $'$를 붙인다.

• 접두 문자열의 길이가 1인 경우: 처음 호출인 경우이다. 항등 타입 치환 함수가 조건을 만족한다. 즉, $\{\}\Gamma_0 > \Gamma_0$ 이고 $\{\}\rho_0 = \rho_0$ 이다.

이제 문자열의 길이가 1보다 큰 경우를 증명한다. 먼저 접두 문자열이 귀환(u)으로 끝나는 경우를 보자.

• e로부터 귀환한 경우: $[\![A(\Gamma_0, e_0, \rho_0)]\!]$이 $(\Gamma, e, \rho)^d \cdots (\Gamma, e, \rho)^u$를 포함한 경우이다. 귀납 가정에 의하여 $[\![A'(\Gamma_0, e_0, \rho_0)]\!]$ 또한 $(\Gamma', e, \rho')^d$를 포함하고, $R\rho' = \rho$이고 $R\Gamma' > \Gamma$를 만족하는 R이 존재한다. 보조정리 11에 의해 $A'(\Gamma', e, \rho')$은 성공한다. 즉, $[\![A'(\Gamma_0, e_0, \rho_0)]\!]$이 $(\Gamma', e, \rho')^u$를 포함한다.

이제 접두 문자열이 호출로 끝나는 경우를 증명한다: $(\Gamma_0, e_0, \rho_0) \cdots (\Gamma, e, \rho)^d$.

• $\lambda x.e$의 e를 호출한 경우: $[\![A(\Gamma_0, e_0, \rho_0)]\!]$이

$$(\Gamma, \lambda x.e, \rho)^d (S_1\Gamma + x : S_1\beta_1, e, S_1\beta_2)^d$$

를 포함한 경우이다. 여기서 S_1은 (G.3)의 $U(\beta_1 \to \beta_2, \theta)$의 결과이고, β_1, β_2는 (G.3)의 새 타입 변수들이다. 귀납 가정에 의해 $[\![A'(\Gamma_0, e_0, \rho_0)]\!]$ 또한 $(\Gamma', \lambda x.e, \rho')^d$를 포함하고,

$$R\Gamma' > \Gamma \tag{39}$$

와 $R\rho' = \rho$를 만족하는 R이 존재한다.

$A'(\Gamma', \lambda x.e, \rho')$가 e를 호출하기 위해서는 (G.3)의 동일화가 성공해야 한다. $A \sqsubseteq A'$이므로

$$\theta = (R+_{supp(P)} \cup P)\theta' \tag{40}$$

와 $supp(P) \subseteq ftv(\theta') \setminus ftv(\rho')$을 만족하는 P가 존재한다. G의 정의에 의해

$$supp(P) \cap ftv(\Gamma') = \emptyset \tag{41}$$

임을 주목하라.

$$R_0 = R+_{\{\beta_1',\beta_2'\} \cup supp(P)} \cup P \cup \{\beta_1/\beta_1', \beta_2/\beta_2'\}$$

라 하자. 여기서 β_1', β_2'는 A'에서 사용된 (G.3)의 새 타입 변수이다. 그러면 S_1R_0이 (G.3)의 $\beta_1' \to \beta_2'$과 θ'을 동일화한다:

$$\begin{aligned} S_1R_0(\theta') &= S_1(R+_{supp(P)} \cup P)\theta' && \beta_1', \beta_2' \notin ftv(\theta') \text{이므로} \\ &= S_1\theta && (40)\text{에 의해} \\ &= S_1(\beta_1 \to \beta_2) && (G.3)\text{에 의해} \\ &= S_1R_0(\beta_1' \to \beta_2') && R_0\text{의 정의에 의해} \end{aligned}$$

그러므로, A'의 (G.3)에 동일화는 S_1'을 결과로 성공

한다. $[A'(\Gamma_0, e_0, \rho_0)]$ 는 $(S_1'\Gamma' + x: S_1'\beta_1', e, S_1'\beta_2')^d$ 를 포함한다.

남은 것은 $R'(S_1'\Gamma' + x: S_1'\beta_1') > (S_1\Gamma + x: S_1\beta_1)$ 와 $R'(S_1'\beta_2') = S_1\beta_2$ 를 만족하는 R' 이 존재함을 증명하는 것이다. (G.3)가 S_1' 을 결과로 성공했으므로 정리 1에 의해

$$S_1R_0 = R_1S_1' \tag{42}$$

를 만족하는 R_1 이 존재한다. 그러면 R_1 이 그러한 R' 이다. 왜냐하면,

$$\begin{array}{ll} R_1(S_1'\Gamma'' + x: S_1'\beta_1') & \\ = S_1R_0(\Gamma'' + x: \beta_1') & (42)에\ 의해 \\ = S_1((R\dagger_{supp(P)} \cup P)\Gamma'' + x: R_0\beta_1') & \beta_1', \beta_2' \notin ftv(\Gamma')이므로 \\ = S_1(R\Gamma'' + x: \beta_1) & (41)과\ R_0의\ 정의에\ 의해 \\ > S_1(\Gamma + x: \beta_1) & (39)와\ 보조정리\ 3에\ 의해 \end{array}$$

$$\begin{array}{rll} R_1(S_1'\beta_2') & = & S_1R_0\beta_2' \quad (42)에\ 의해 \\ & = & S_1\beta_2 \quad R_0의\ 정의에\ 따라 \end{array}$$

이기 때문이다.

• $e\ e_2$ **의 e 를 호출한 경우**: $[A(\Gamma_0, e_0, \rho_0)]$ 이

$$(\Gamma, e\ e_2, \rho)^d(\Gamma, e, \theta_1)^d$$

를 포함한 경우이다. 여기서 θ_1 는 (G.8)의 $\beta \to \rho$ 로부터 느슨해진 타입 조건이다. 귀납 가정에 의해 $[A'(\Gamma_0, e_0, \rho_0)]$ 또한 $(\Gamma', e\ e_2, \rho')^d$ 를 포함하고

$$R\Gamma' > \Gamma \tag{43}$$

와 $R\rho' = \rho$ 를 만족하는 R 이 존재한다. 그러므로, G 의 정의에 따라 $[A'(\Gamma_0, e_0, \rho_0)]$ 는 $(\Gamma', e, \theta_1')^d$ 를 포함하고 있다. 여기서, θ_1' 은 (G.8)의 $\beta' \to \rho'$ 로부터 느슨해진 타입 조건이다.

남은 것을 증명하자. $R_0 = R\dagger_{\{\beta'\}} \cup \{\beta/\beta'\}$ 라 하자. 여기서, β, β' 은 각각 (G.6)에 있는 A, A' 의 새 타입 변수이다. $A \sqsubseteq A'$ 이고

$$\begin{array}{rll} R_0(\beta' \to \rho') & = & \beta \to R\rho' \quad \beta' \notin ftv(\rho')이므로 \\ & = & \beta \to \rho \end{array}$$

이므로 정의 2에 따라

$$(R_0\dagger_{supp(P)} \cup P)\theta_1' = \theta_1$$

와 $supp(P) \subseteq ftv(\theta_1') \setminus ftv(\beta' \to \rho')$ 를 만족하는 P 가 존재한다. G 의 정의에 의해 $supp(P) \cap ftv(\Gamma') = \emptyset$ 이 성립하므로,

$$\begin{array}{ll} (R_0\dagger_{supp(P)} \cup P)(\Gamma') & \\ = R\Gamma' & (\{\beta\} \cup supp(P)) \cap ftv(\Gamma') = \emptyset 이므로 \\ > \Gamma & (43)에\ 의해 \end{array}$$

가 성립한다.

• $e_1\ e$ **의 e 를 호출한 경우**: $[A(\Gamma_0, e_0, \rho_0)]$ 이

$$(\Gamma, e_1\ e, \rho)^d(\Gamma, e_1, \theta_1)^d \cdots (\Gamma, e_1, \theta_1)^u(S_2S_1\Gamma, e, \theta_3)^d$$

를 포함한 경우이다. 여기서 θ_1, θ_2, θ_3 는 각각 (G.6), (G.7), (G.8)의 느슨한 타입 조건이고, S_1 은 (G.6)의 $G(\Gamma, e_1, \theta_1)$ 의 결과이고, S_2 는 (G.7)의 $U(S_1\theta_1, \theta_2)$ 의 결과이다. 귀납 가정에 의해 $[A'(\Gamma_0, e_0, \rho_0)]$ 는 $(\Gamma', e_1\ e, \rho')^d$ 를 포함하고

$$R\Gamma' > \Gamma \tag{44}$$

와 $R\rho' = \rho$ 를 만족하는 R 이 있다.

$A'(\Gamma', e_1\ e, \rho')$ 이 e 를 호출하기 위해서는 (G.6)의 e_1 호출이 성공하고 (G.7)의 동일화가 성공해야 한다.

- (G.6)의 $A'(\Gamma', e_1, \theta_1')$ 이 성공한다: $R_0 = R\dagger_{\{\beta'\}} \cup \{\beta/\beta'\}$ 라 하자. 여기서, β, β' 는 각각 A, A' 의 (G.6)에서의 새 타입 변수이다. $A \sqsubseteq A'$ 이고

$$\begin{array}{rll} R_0(\beta' \to \rho') & = & \beta \to R\rho' \quad \beta' \notin ftv(\rho')이므로 \\ & = & \beta \to \rho \end{array} \tag{45}$$

이므로, 정의 2에 의해

$$\theta_1 = (R_0\dagger_{supp(P_1)} \cup P_1)\theta_1' \tag{46}$$

와 $supp(P_1) \subseteq ftv(\theta_1') \setminus ftv(\beta' \to \rho')$ 를 만족하는 타입 치환 함수 P 가 존재한다. G 의 정의에 의해

$$supp(P_1) \cap (ftv(\Gamma') \cup ftv(\beta' \to \rho')) = \emptyset \tag{47}$$

이므로

$$\begin{array}{ll} (R_0\dagger_{supp(P_1)} \cup P_1)\Gamma' & \\ = R\Gamma' & (47)과\ \beta' \notin ftv(\Gamma')이므로 \\ > \Gamma & (44)에\ 의해 \end{array} \tag{48}$$

이다. 결국, $[A(\Gamma_0, e_0, \rho_0)]$ 는 $(\Gamma, e_1, \theta_1)^u$ 를 포함하고 있고 $(R_0\dagger_{supp(P_1)} \cup P_1)\Gamma' > \Gamma$ 이고 $(R_0\dagger_{supp(P_1)} \cup P_1)\theta_1 = \theta_1$ 이다. 그러므로, 보조정리 11에 의해 $A'(\Gamma', e_1, \theta_1')$ 는 S_1' 을 결과로 성공하며,

$$(R_1S_1')\dagger_{V_1} = (S_1(R_0\dagger_{supp(P_1)} \cup P_1))\dagger_{V_1} \tag{49}$$

를 만족하는 R_1 이 존재한다. 여기서 V_1 은 $A'(\Gamma', e_1, \theta_1')$ 에서 발생한 새 타입 변수들의 집합이다.

- (G.7)의 $U(S_1'\theta_1', \theta_2')$ 는 성공한다: $A \sqsubseteq A'$ 이고

$$\begin{array}{ll} R_1(S_1'(\beta' \to \rho')) & \\ = S_1(R_0\dagger_{supp(P_1)} \cup P_1)(\beta' \to \rho') & \\ \quad (49)와\ ftv(\beta' \to \rho') \cap V_1 = \emptyset 이므로 & \\ = S_1R_0(\beta' \to \rho') & (47)에\ 의해 \\ = S_1(\beta \to \rho) & (45)에\ 의해 \end{array} \tag{50}$$

이므로, 정의 2에 의해

$$\theta_2 = (R_1\dagger_{supp(P_2)} \cup P_2)\theta_2' \tag{51}$$

와 $supp(P_2) \subseteq ftv(\theta_2') \setminus ftv(S_1'(\beta' \to \rho'))$ 를 만족하는 P_2 가 존재한다.

$$\begin{array}{ll} ftv(S_1'\theta_1') \cup ftv(S_1'\beta') \cup ftv(S_1'\Gamma') & \\ \subseteq itv(S_1') \cup ftv(\theta_1') \cup \{\beta'\} \cup ftv(\Gamma') & 보조정리\ 7에\ 의해 \\ \subseteq V_1 \cup ftv(\theta_1') \cup \{\beta'\} \cup ftv(\Gamma') & 보조정리\ 8에\ 의해 \end{array}$$

이므로, G의 정의에 의해

$$supp(P_2) \cap (ftv(S_1'\theta_1') \cup ftv(S_1'\beta') \cup ftv(S_1'\Gamma')) = \emptyset \quad (52)$$

가 성립한다. 그러면 $S_2(R_1 \dagger_{supp(P_3)} \cup P_2)$가 (G.7)의 $S_1'\theta_1'$과 θ_2'를 동일화한다:

$$\begin{aligned}
&S_2(R_1 \dagger_{supp(P_2)} \cup P_2)(S_1'\theta_1') \\
&= S_2R_1S_1'\theta_1' && (52)\text{에 의해} \\
&= S_2S_1(R_0 \dagger_{supp(P_1)} \cup P_1)\theta_1' && (49)\text{와 } ftv(\theta_1') \cap V_1 = \emptyset \text{이므로} \\
&= S_2S_1\theta_1 && (46)\text{에 의해} \\
&= S_2\theta_2 && (G.7)\text{에 의해} \\
&= S_2(R_1 \dagger_{supp(P_2)} \cup P_2)(\theta_2') && (51)\text{에 의해}
\end{aligned}$$

그러므로 A'의 (G.7)에서의 동일화는 S_2'를 결과로 성공한다.

결과적으로 $[\![A'(\Gamma_0, e_0, \rho_0)]\!]$는 $(S_2'S_1'\Gamma', e, \theta_3')^d$를 포함한다.

남은 증명은 $R'\theta_3' = \theta_3$와 $R'(S_2'S_1'\Gamma') > S_2S_1\Gamma$를 만족하는 R'을 찾는 것이다. (G.7)이 S_2'을 결과로 성공했으므로 정리 1에 의해

$$R_2S_2' = S_2(R_1 \dagger_{supp(P_2)} \cup P_2) \quad (53)$$

를 만족하는 R_1이 존재한다. $A \sqsubseteq A'$이고

$$\begin{aligned}
R_2(S_2'S_1'\beta') &= S_2(R_1 \dagger_{supp(P_2)} \cup P_2)S_1'\beta' && (53)\text{에 의해} \\
&= S_2R_1S_1'\beta' && (52)\text{에 의해} \\
&= S_2S_1\beta && (50)\text{에 의해}
\end{aligned}$$

이므로, 정의 2에 의해

$$\theta_3 = (R_2 \dagger_{supp(P_3)} \cup P_3)\theta_3'$$

와 $supp(P_3) \subseteq ftv(\theta_3') \setminus ftv(S_2'S_1'\beta')$를 만족하는 P_3가 존재한다. 보조정리 7, 8, 정리 1에 의해,

$$\begin{aligned}
&ftv(S_2'S_1'\Gamma') \\
&\subseteq ftv(\theta_1) \cup ftv(\theta_2) \cup V_1 \cup ftv(\Gamma') \\
&\subseteq supp(P_1) \cup ftv(\beta \to \rho) \cup supp(P_2) \cup V_1 \cup ftv(\Gamma')
\end{aligned}$$

이므로 G의 정의에 의해

$$supp(P_3) \cap ftv(S_2'S_1'\Gamma') = \emptyset \quad (54)$$

가 성립한다. 그러므로 $(R_2 \dagger_{supp(P_3)} \cup P_3)$이 그러한 R'이 된다:

$$\begin{aligned}
&(R_2 \dagger_{supp(P_3)} \cup P_3)(S_2'S_1'\Gamma') \\
&= R_2S_2'S_1'\Gamma' && (54)\text{에 의해} \\
&= S_2(R_1 \dagger_{supp(P_2)} \cup P_2)S_1'\Gamma' && (53)\text{에 의해} \\
&= S_2R_1S_1'\Gamma' && (52)\text{에 의해} \\
&= S_2S_1(R_0 \dagger_{supp(P_1)} \cup P_1)\Gamma' && (49)\text{와 } ftv(\Gamma') \cap V_1 = \emptyset \text{이므로} \\
&> S_2S_1\Gamma && (48)\text{과 보조정리 3에 의해.}
\end{aligned}$$

• let $x = e$ in e_2 **의 e를 호출한 경우:** $[\![A(\Gamma_0, e_0, \rho_0)]\!]$이

$$(\Gamma, \text{let } x = e \text{ in } e_2, \rho)^d (\Gamma, e, \beta)^d$$

를 포함한 경우이다. 여기서 β는 (G.11)의 새 타입 변수이다. 귀납 가정에 의해 $[\![A'(\Gamma_0, e_0, \rho_0)]\!]$ 또한 $(\Gamma', \text{let } x = e \text{ in } e_2, \rho')^d$를 포함하고 $R\Gamma' > \Gamma$와 $R\rho' = \rho$를 만족하는 R이 존재한다. G의 정의에 따라 $[\![A'(\Gamma_0, e_0, \rho_0)]\!]$는 $(\Gamma', e_1, \beta')^d$를 포함하고 있다. 여기서 β'는 A'의 (G.11)에서의 새 타입 변수이다. $R_0 = R \dagger_{\{\beta'\}} \cup \{\beta/\beta'\}$라 하면 $R_0\Gamma' = R\Gamma' > \Gamma$와 $R_0\beta' = \beta$를 만족한다.

• let $x = e_1$ in e **의 e를 호출한 경우:** $[\![A(\Gamma_0, e_0, \rho_0)]\!]$이

$$\begin{aligned}
&(\Gamma, \text{let } x = e_1 \text{ in } e, \rho)^d (\Gamma, e, \beta)^d \cdots \\
&(\Gamma, e, \beta)^u (S_1\Gamma + x : Clos_{S_1\Gamma}(S_1\beta), e, \theta)^d
\end{aligned}$$

포함한 경우이다. 여기서 β는 (G.11)의 새 타입 변수이고, θ는 (G.12)의 느슨한 타입 조건이고, S_1는 (G.11)의 $G(\Gamma, e_1, \beta)$의 결과이다. 귀납 가정에 의해 $[\![A'(\Gamma_0, e_0, \rho_0)]\!]$는 $(\Gamma', \text{let } x = e_1 \text{ in } e, \rho')^d$를 포함하고

$$R\Gamma' > \Gamma \quad (55)$$

와 $R\rho' = \rho$를 만족하는 R이 존재한다. $R_0 = R \dagger_{\{\beta'\}} \cup \{\beta/\beta'\}$라 하자. 여기서 β'은 A'의 (G.11)에서의 새 타입 변수이다. 그러면,

$$\begin{aligned}
R_0\Gamma' &= R\Gamma' \quad \beta' \notin ftv(\Gamma') \\
&> \Gamma \quad (55)\text{에 의해}
\end{aligned} \quad (56)$$

와 $R_0\beta' = \beta$가 성립한다. 그러면, 보조정리 11에 의해 (G.11)의 $A'(\Gamma', e_1, \beta')$는 S_1'를 결과로 성공한다. 그러므로 $[\![A'(\Gamma_0, e_0, \rho_0)]\!]$는 $(S_1'\Gamma' + x : Clos_{S_1'\Gamma'}(S_1'\beta'), e_2, \theta')^d$를 포함한다.

남은 것은 $R'\theta' = \theta$와 $R'(S_1'\Gamma' + x : Clos_{S_1'\Gamma'}(S_1'\beta')) > S_1\Gamma + x : Clos_{S_1\Gamma}(S_1\beta)$를 만족하는 R'이 존재함을 증명하는 것이다. (G.11)이 S_1'을 결과로 성공하므로 보조정리 11에 의해

$$(R_1S_1') \dagger_{V_1} = (S_1R_0) \dagger_{V_1} \quad (57)$$

를 만족하는 R_1이 존재한다. 여기서, V_1은 $A'(\Gamma', e_1, \beta')$에서 발생한 새 타입 변수의 집합이다. $A \sqsubseteq A'$이고

$$\begin{aligned}
R_1(S_1'\rho') &= S_1R_0\rho' && (57)\text{과 } ftv(\rho') \cap V_1 = \emptyset \text{이므로} \\
&= S_1R\rho' && \beta' \notin ftv(\rho') \\
&= S_1\rho
\end{aligned}$$

이므로, 정의 2에 의해

$$(R_1 \dagger_{supp(P)} \cup P)\theta' = \theta$$

와 $supp(P) \subseteq ftv(\theta') \setminus ftv(S_1'\rho')$를 만족하는 P가 존재한다.

$$\begin{aligned}
&ftv(S_1'\Gamma') \cup ftv(S_1'\beta') \\
&\subseteq itv(S_1') \cup ftv(\Gamma') \cup \{\beta'\} && \text{보조정리 7에 의해} \\
&\subseteq V_1 \cup ftv(\Gamma') \cup \{\beta'\} && \text{보조정리 8에 의해}
\end{aligned}$$

이므로, G의 정의에 의해

$$supp(P) \cap (ftv(S_1'\Gamma') \cup ftv(S_1'\beta')) = \emptyset \quad (58)$$

가 성립한다. 그러므로, $(R_1 \dagger_{supp(P)} \cup P)$가 그러한 R'이다. 왜냐하면,

$(R_1 \dagger_{supp(P)} \cup P)(S_1'\Gamma')$
$= R_1(S_1'\Gamma')$ (58)에 의해
$= S_1R_0\Gamma'$ (57)과 $V_1 \cap ftv(\Gamma') = \emptyset$
$> S_1\Gamma'$ (56)과 보조정리 3에 의해 (59)

$(R_1 \dagger_{supp(P)} \cup P)(Clos_{S_1'}\Gamma' S_1'\beta')$
$= R_1 Clos_{S_1'\Gamma'}(S_1'\beta')$ (58)에 의해
$> Clos_{R_1S_1'\Gamma'}(R_1S_1'\beta')$ 보조정리 5에 의해
$> Clos_{S_1\Gamma}(R_1S_1'\beta')$ (59)과 보조정리 10에 의해
$= Clos_{S_1\Gamma}(S_1R_0\beta')$ (57)와 $\beta' \notin V_1$이므로
$= Clos_{S_1\Gamma}(S_1\beta)$ R_0의 정의에 따라

이기 때문이다.

• fix $f\ \lambda x.e$ 의 e를 호출한 경우: $[A(\Gamma_0, e_0, \rho_0)]$ 이

$$(\Gamma, \text{fix } f\ \lambda x.e, \rho)^d (\Gamma + f: \theta_1, \lambda x.e, \theta_2)^d$$

를 포함하는 경우이다. 여기서, θ_1, θ_2는 (G.14)에서의 느슨한 타입 조건이다. 귀납 가정에 의해 $[A'(\Gamma_0, e_0, \rho_0)]$는 $(\Gamma', \text{fix } f\ \lambda x.e, \rho')^d$를 포함하고

$$R\Gamma'' > \Gamma \quad (60)$$

와 $R\rho' = \rho$를 만족하는 R이 존재한다. G의 정의에 따라 $[A'(\Gamma_0, e_0, \rho_0)]$ 또한 $(\Gamma' + f: \theta_1', \lambda x.e, \theta_2')^d$를 포함한다.

남은 것을 증명하자. $A \sqsubseteq A'$이고 $R\rho' = \rho$이므로, 정의 2에 의해

$$\theta_1 = (R \dagger_{supp(P)} \cup P)\theta_1' \quad (61)$$

와 $\theta_2 = (R \dagger_{supp(P)} \cup P)\theta_2'$를 만족하는 P가 존재한다. 그러므로,

$(R \dagger_{supp(P)} \cup P)(\Gamma' + f: \theta_1')$
$= R\Gamma'' + f: \theta_1$ (61)과 $ftv(\Gamma') \cap supp(P) = \emptyset$이므로
$> \Gamma + f: \theta_1$ (60)에 의해

가 성립한다. □

참 고 문 헌

[1] Robin Milner. A theory of type polymorphism in programming. *Journal of Computer and System Sciences*, 17:348-375, 1978.

[2] Luis Damas and Robin Milner. Principal type-scheme for functional programs. In *Proceedings of the 9th Annual ACM Symposium on Principles of Programming Languages*, pages 207-212, New York, 1982. ACM Press.

[3] Luis Manuel Martins Damas. *Type assignment in programming Languages*. Ph. D. Thesis, University of Edinburgh, 1984.

[4] Oukseh Lee and Kwangkeun Yi. Proofs about a folklore let-polymorphic type inference algorithm. *ACM Transactions on Programming Languages and Systems*, 20(4):707-723, July 1998.

[5] 이욱세, 이광근. 하향식 타입 유추 알고리즘 M의 속성 증명. *정보과학회 논문지*, 25(8):1281-1292, August 1998.

[6] The Standard ML of New Jersey, release 110.0.6. Bell Labs, Lucent Technologies, November 1999. http://cm.bell-labs.com/cm/cs/what/smlnj.

[7] Xavier Leroy, Didier Rémy, Jérôme Vouillon, and Damien Doligez. The objective caml system release 2.04. Institut National de Recherche en Informatique et en Automatique, November 1999. http://caml.inria.fr.

[8] 주상현, 이욱세, 이광근. 복합형 타입 유추 알고리즘 M의 병목을 해소하기 위한 알고리즘 H. *정보과학회 논문지*, 2000. (심사완료)

[9] 주상현. *M의 병목을 해소하는 타입유추 알고리즘*. 석사논문, 한국과학기술원, 2000.

[10] J. A. Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM*, 12(1):23-41, January 1965.

[11] Bruce J. McAdam. On the unification of substitutions in type inference. In Kevin Hammond, Anthony J. T. Davie, and Chris Clack, editors, *Proceedings of The International Workshop on Implementation of Fuctional Languages*, volume 1595 of *Lecture Notes in Computer Science*, pages 139-154. Springer-Verlag, September 1998.

[12] Fritz Henglein. Type inference with polymorphic recursion. *ACM Transactions on Programming Languages and Systems*, 15(2):253-289, April 1993.

[13] Kenta Cho and Kazunori Ueda. Diagnosing non-well-moded concurrent logic programs. In *Joint International Conference on Logic Programming*, pages 215-229. MIT Press, 1996.

[14] Alexander Aiken and Edward L. Wimmers. Type inclusion constraints and type inference. In *Proceedings of Functional Programming Languages and Computer Architecture*, pages 31-41, 1993.

[15] Satish R. Thatte. Type inference with partial types. *Theoretical Computer Science*, 124(1):127-148, February 1994.

[16] Didier Rémy. Extending ML type system with a sorted equational theory. Research Report 1766, Institut National de Recherche en Informatique et Automatisme, Rocquencourt, BP 105, 78 153 Le Chesnay Cedex, France, 1992.

이 욱 세

1995년 한국과학기술원 전산학과 학사(B.C.). 1997년 한국과학기술원 전산학과 석사(M.S.). 1997년 ~ 현재 한국과학기술원 전자전산학과 박사과정 재학중. 관심분야는 ML, 타입 이론, 프로그램 분석.

이 광 근

1987년 B.S. 계산통계학, 서울대학교 자연과학대학. 1990년 M.S. Computer Science, University of Illinois at Urbana-Chanpaign. 1993년 Ph.D. Computer Science, University of Illinois at Urbana-Champaign. 1993년 ~ 1995년 Member of Technical Staff, Software principles Research Dept., At & T Bell laboratories. 1995년 ~ 현재 한국과학기술원 전산학과 조교수.