

# SNU 4541.310 Programming Language

## Simple Type System

Prof. Kwangkeun Yi

## 단순 타입 시스템 Simple Type System

어디보자: 타입 추론

추론 규칙의 안전성 증명

추론 규칙의 구현

## 어디보자: 타입 추론

- ▶  $e$ 의 생김새마다 오직 하나의 규칙. 따라서,

$$\Gamma \vdash \lambda x.e : \tau \rightarrow \tau' \Leftrightarrow \Gamma + x : \tau \vdash e : \tau'$$

$$\Gamma \vdash e e' : \tau \Leftrightarrow \Gamma \vdash e : \tau' \rightarrow \tau \wedge \Gamma \vdash e' : \tau'$$

⋮

- ▶ 주어진  $e$ 에 대해서,  $\Gamma \vdash e : \tau$ 인  $\tau$ 가 여럿가능

$$\frac{\{x : \iota\} \vdash x : \iota}{\vdash \lambda x.x : \iota \rightarrow \iota} \quad \frac{\{x : \iota \rightarrow \iota\} \vdash x : \iota \rightarrow \iota}{\vdash \lambda x.x : (\iota \rightarrow \iota) \rightarrow (\iota \rightarrow \iota)}$$

$$\frac{\frac{\vdots}{\vdash \lambda x.x + 1 : \iota \rightarrow \iota} \quad \frac{\vdots}{\vdash (\lambda y.y) 2 : \iota}}{\vdash (\lambda x.x + 1) ((\lambda y.y) 2) : \iota}$$

$$\frac{\frac{\vdots}{\vdash \lambda x.x + 1 : \iota \rightarrow \iota} \quad \frac{\frac{\vdots}{\vdash \lambda y.y : \iota \rightarrow \iota} \quad \frac{\text{!}}{\vdash \lambda z.z : \iota}}{\vdash (\lambda y.y) (\lambda z.z) : \iota}}{\vdash (\lambda x.x + 1) ((\lambda y.y) (\lambda z.z)) : \iota}$$

$$\frac{\frac{\vdots}{\{f : \tau \rightarrow \tau'\} \vdash f : \tau \rightarrow \tau'} \quad \frac{\vdots}{\{f : \tau \rightarrow \tau'\} \vdash f : \tau} \quad \tau = \tau \rightarrow \tau'}{\frac{\{f : \tau \rightarrow \tau'\} \vdash f f : \tau'}{\vdash \lambda f.f f : (\tau \rightarrow \tau') \rightarrow \tau'}}$$

# 추론 규칙의 안전성 증명: 추론되는 대로 실행된다

## 증명방법 I

- ▶ Progress: 값이 나올 때 까지 문제없이 진행한다.  
 $\vdash e : \tau$  이고  $e$ 가 값이 아니면 반드시  $e \rightarrow e'$ .
- ▶ Preservation: 진행은 타입을 보존한다.  
 $\vdash e : \tau$  이고  $e \rightarrow e'$  이면  $\vdash e' : \tau$ .

## 치환 $\{v/x\}e$

프로그램 실행의 핵심:  $(\lambda x.e) v \rightarrow \{v/x\}e$

$$\{v/x\}n = n$$

$$\{v/x\}x = v$$

$$\{v/x\}y = y \quad \text{if } y \neq x$$

$$\{v/x\}(e_1 + e_2) = (\{v/x\}e_1) + (\{v/x\}e_2)$$

$$\{v/x\}(e_1 e_2) = (\{v/x\}e_1) (\{v/x\}e_2)$$

$$\{v/x\}(\lambda y.e) = \lambda y.\{v/x\}e \quad \text{if } y \notin \{x\} \cup FV(v)$$

**사실:** 묶여있는 변수만 다른  $\lambda x.e$ 와  $\lambda x'.e'$ 는 같은 것; 서로 항상 대신할 수 있다.

- ▶ 따라서,  $\{v/x\}(\lambda y.e)$ 가 항상 정의 될 수 있는  $\lambda y.e$ 라고 (즉,  $y \notin \{x\} \cup FV(v)$ 라고) 간주 해도 무방.

# 타입이 있으면 문제없이 진행

## Theorem (Progress)

$\vdash e : \tau$  이고  $e$ 가 값이 아니면 반드시 진행  $e \rightarrow e'$  한다.

**Proof.**  $\vdash e : \tau$ 의 증명에 대한 귀납법으로. <sup>1</sup>

$e = e_1 e_2$ 인 경우:  $\vdash e_1 e_2 : \tau$ 이므로 타입추론 규칙에 의해  $\vdash e_1 : \tau' \rightarrow \tau$ 이고  $\vdash e_2 : \tau'$  이다. 따라서, 귀납 가정에 의해서,

다른 경우도 마찬가지로 증명.



# 타입이 있으면 문제없이 진행

## Theorem (Progress)

$\vdash e : \tau$  이고  $e$ 가 값이 아니면 반드시 진행  $e \rightarrow e'$  한다.

**Proof.**  $\vdash e : \tau$ 의 증명에 대한 귀납법으로. <sup>1</sup>

$e = e_1 e_2$ 인 경우:  $\vdash e_1 e_2 : \tau$ 이므로 타입추론 규칙에 의해  $\vdash e_1 : \tau' \rightarrow \tau$ 이고  $\vdash e_2 : \tau'$ 이다. 따라서, 귀납 가정에 의해서,

- ▶  $e_1$ 이 값이 아니면 진행  $e_1 \rightarrow e'_1$  하고, 이는 곧 프로그램 실행  $\rightarrow$ 의 정의에 의해  $e_1 e_2 \rightarrow e'_1 e_2$ 과 같다.

다른 경우도 마찬가지로 증명.

□

# 타입이 있으면 문제없이 진행

## Theorem (Progress)

$\vdash e : \tau$  이고  $e$ 가 값이 아니면 반드시 진행  $e \rightarrow e'$  한다.

**Proof.**  $\vdash e : \tau$ 의 증명에 대한 귀납법으로. <sup>1</sup>

$e = e_1 e_2$ 인 경우:  $\vdash e_1 e_2 : \tau$ 이므로 타입추론 규칙에 의해  $\vdash e_1 : \tau' \rightarrow \tau$ 이고  $\vdash e_2 : \tau'$ 이다. 따라서, 귀납 가정에 의해서,

- ▶  $e_1$ 이 값이 아니면 진행  $e_1 \rightarrow e'_1$  하고, 이는 곧 프로그램 실행  $\rightarrow$ 의 정의에 의해  $e_1 e_2 \rightarrow e'_1 e_2$ 과 같다.
- ▶ 마찬가지로,  $e_1$ 이 값이고  $e_2$ 가 값이 아니라면 진행  $e_2 \rightarrow e'_2$  하고, 이는 곧 프로그램 실행  $\rightarrow$ 의 정의에 의해  $e_1 e_2 \rightarrow e_1 e'_2$ 과 같다.

다른 경우도 마찬가지로 증명.

□

# 타입이 있으면 문제없이 진행

## Theorem (Progress)

$\vdash e : \tau$  이고  $e$ 가 값이 아니면 반드시 진행  $e \rightarrow e'$  한다.

**Proof.**  $\vdash e : \tau$ 의 증명에 대한 귀납법으로. <sup>1</sup>

$e = e_1 e_2$ 인 경우:  $\vdash e_1 e_2 : \tau$ 이므로 타입추론 규칙에 의해  $\vdash e_1 : \tau' \rightarrow \tau$ 이고  $\vdash e_2 : \tau'$ 이다. 따라서, 귀납 가정에 의해서,

- ▶  $e_1$ 이 값이 아니면 진행  $e_1 \rightarrow e'_1$  하고, 이는 곧 프로그램 실행  $\rightarrow$ 의 정의에 의해  $e_1 e_2 \rightarrow e'_1 e_2$ 과 같다.
- ▶ 마찬가지로,  $e_1$ 이 값이고  $e_2$ 가 값이 아니라면 진행  $e_2 \rightarrow e'_2$  하고, 이는 곧 프로그램 실행  $\rightarrow$ 의 정의에 의해  $e_1 e_2 \rightarrow e_1 e'_2$ 과 같다.
- ▶  $e_1$ 과  $e_2$ 가 모두 값이라면,  $\vdash e_1 : \tau' \rightarrow \tau$ 일 수 있는 값  $e_1$ 은 오직  $\lambda x.e'$  경우 뿐이다. 따라서 프로그램 실행  $\rightarrow$ 의 정의에 의해 반드시 진행  $e_1 e_2 = \lambda x.e' e_2 \rightarrow \{e_2/x\}e'$ 한다.

다른 경우도 마찬가지로 증명.

□

# 실행은 타입을 보존

## Theorem (Preservation)

$\vdash e : \tau$  이고  $e \rightarrow e'$  이면  $\vdash e' : \tau$ .

**Proof.**  $\vdash e : \tau$ 의 증명에 대한 귀납법으로 진행한다. <sup>2</sup>

$e = e_1 e_2$ 인 경우:  $\vdash e_1 e_2 : \tau$ 이므로 타입추론 규칙에 의해  $\vdash e_1 : \tau' \rightarrow \tau$ 이고  $\vdash e_2 : \tau'$  이다.  $e_1 e_2 \rightarrow e'$ 이라면 세가지 경우밖에 없다:



# 실행은 타입을 보존

## Theorem (Preservation)

$\vdash e : \tau$  이고  $e \rightarrow e'$  이면  $\vdash e' : \tau$ .

**Proof.**  $\vdash e : \tau$ 의 증명에 대한 귀납법으로 진행한다. <sup>2</sup>

$e = e_1 e_2$ 인 경우:  $\vdash e_1 e_2 : \tau$ 이므로 타입추론 규칙에 의해  $\vdash e_1 : \tau' \rightarrow \tau$ 이고  $\vdash e_2 : \tau'$ 이다.  $e_1 e_2 \rightarrow e'$ 이라면 세가지 경우밖에 없다:

- ▶  $e_1 \rightarrow e'_1$ 이라서  $e_1 e_2 \rightarrow e'_1 e_2$ 인 경우. 귀납 가정에 의해  $\vdash e'_1 : \tau' \rightarrow \tau$ .  $\vdash e_2 : \tau'$ 이므로, 타입추론 규칙에 의해  $\vdash e'_1 e_2 : \tau$ .



# 실행은 타입을 보존

## Theorem (Preservation)

$\vdash e : \tau$  이고  $e \rightarrow e'$  이면  $\vdash e' : \tau$ .

**Proof.**  $\vdash e : \tau$ 의 증명에 대한 귀납법으로 진행한다. <sup>2</sup>

$e = e_1 e_2$ 인 경우:  $\vdash e_1 e_2 : \tau$ 이므로 타입추론 규칙에 의해  $\vdash e_1 : \tau' \rightarrow \tau$ 이고  $\vdash e_2 : \tau'$ 이다.  $e_1 e_2 \rightarrow e'$ 이라면 세가지 경우밖에 없다:

- ▶  $e_1 \rightarrow e'_1$ 이라서  $e_1 e_2 \rightarrow e'_1 e_2$ 인 경우. 귀납 가정에 의해  $\vdash e'_1 : \tau' \rightarrow \tau$ .  $\vdash e_2 : \tau'$ 이므로, 타입추론 규칙에 의해  $\vdash e'_1 e_2 : \tau$ .
- ▶  $e_1$ 은 값이고  $e_2 \rightarrow e'_2$ 이라서  $e_1 e_2 \rightarrow e_1 e'_2$ 인 경우. 위의 경우와 유사.



# 실행은 타입을 보존

## Theorem (Preservation)

$\vdash e : \tau$  이고  $e \rightarrow e'$  이면  $\vdash e' : \tau$ .

**Proof.**  $\vdash e : \tau$ 의 증명에 대한 귀납법으로 진행한다. <sup>2</sup>

$e = e_1 e_2$ 인 경우:  $\vdash e_1 e_2 : \tau$ 이므로 타입추론 규칙에 의해  $\vdash e_1 : \tau' \rightarrow \tau$ 이고  $\vdash e_2 : \tau'$ 이다.  $e_1 e_2 \rightarrow e'$ 이라면 세가지 경우밖에 없다:

- ▶  $e_1 \rightarrow e'_1$ 이라서  $e_1 e_2 \rightarrow e'_1 e_2$ 인 경우. 귀납 가정에 의해  $\vdash e'_1 : \tau' \rightarrow \tau$ .  $\vdash e_2 : \tau'$ 이므로, 타입추론 규칙에 의해  $\vdash e'_1 e_2 : \tau$ .
- ▶  $e_1$ 은 값이고  $e_2 \rightarrow e'_2$ 이라서  $e_1 e_2 \rightarrow e_1 e'_2$ 인 경우. 위의 경우와 유사.
- ▶  $e_1$ 과  $e_2$ 가 모두 값인 경우.  $\vdash e_1 : \tau' \rightarrow \tau$ 인 값  $e_1$ 은 타입추론 규칙에 의해  $\lambda x.e'$  밖에는 없다. 즉,  $e_1 e_2 = (\lambda x.e') v$  이고,  $(\lambda x.e') v \rightarrow \{v/x\}e'$ 이다.  $\vdash \lambda x.e' : \tau' \rightarrow \tau$ 이라면 타입추론 규칙에 의해  $x : \tau' \vdash e' : \tau$ 이다.  $\vdash v : \tau'$ 이므로, "Preservation under Substitution Lemma"에 의해  $\vdash \{v/x\}e' : \tau$ 이다.



# 치환은 타입을 보존

## Theorem (Preservation under Substitution)

$\Gamma \vdash v : \tau'$  이고  $\Gamma + x : \tau' \vdash e : \tau$  이면  $\Gamma \vdash \{v/x\}e : \tau$ .

**Proof.**  $\Gamma + x : \tau' \vdash e : \tau$ 의 증명에 대한 귀납법으로 증명한다.

$e = \lambda y.e'$  인 경우: 항상  $y \notin \{x\} \cup FVv$ 인  $\lambda y.e'$ 로 간주할 수 있으므로  $\{v/x\}\lambda y.e' = \lambda y.\{v/x\}e'$ . 따라서, 보일 것은  $\Gamma \vdash \lambda y.\{v/x\}e' : \tau \stackrel{\text{let}}{=} \tau_1 \rightarrow \tau_2$ .

가정  $\Gamma + x : \tau' \vdash \lambda y.e' : \tau_1 \rightarrow \tau_2$  으로부터 타입추론 규칙에 의해

$\Gamma + x : \tau' + y : \tau_1 \vdash e' : \tau_2$  이고,  $\Gamma \vdash v : \tau'$ 와  $y \notin FV(v)$  으로부터

$\Gamma + y : \tau_1 \vdash v : \tau'^3$  이므로, 귀납 가정에 의해  $\Gamma + y : \tau_1 \vdash \{v/x\}e' : \tau_2$ . 즉,

타입추론 규칙에 의해  $\Gamma \vdash \lambda y.\{v/x\}e' : \tau_1 \rightarrow \tau_2$ .

다른 경우는 더욱 단순한 귀납. □

## 증명 방법 II

다음 두개만으로도 충분:

- ▶  $e \rightarrow error$ 를 정의
- ▶ Preservation:  
 $\vdash e : \tau$  이고  $e \rightarrow e'$  이면  $\vdash e' : \tau$ .

왜?

- ▶ 목표: 값이 아닌 식  $e$ 가 타입이 있으면,  $e$ 가 문제없이 진행하며 끝난다면 그 타입의 값이어야.

## 증명 방법 II

다음 두개만으로도 충분:

- ▶  $e \rightarrow error$ 를 정의
- ▶ Preservation:  
 $\vdash e : \tau$  이고  $e \rightarrow e'$  이면  $\vdash e' : \tau$ .

왜?

- ▶ 목표: 값이 아닌 식  $e$ 가 타입이 있으면,  $e$ 가 문제없이 진행하며 끝난다면 그 타입의 값이어야.
- ▶ 값이 아닌 식  $e$ 가 타입이 있다고 하자. 잘 진행하는가?  
그렇다,  $e \rightarrow error$ 로 진행할 수 없다. 그렇게 진행한다면  
모순: "Preservation Lemma"에 의해서  $error$ 가 타입이 있어야  
하는데  $error$ 의 타입을 결정하는 규칙은 없다.

## 증명 방법 II

다음 두개만으로도 충분:

- ▶  $e \rightarrow error$ 를 정의
- ▶ Preservation:  
 $\vdash e : \tau$  이고  $e \rightarrow e'$  이면  $\vdash e' : \tau$ .

왜?

- ▶ 목표: 값이 아닌 식  $e$ 가 타입이 있으면,  $e$ 가 문제없이 진행하며 끝난다면 그 타입의 값이어야.
- ▶ 값이 아닌 식  $e$ 가 타입이 있다고 하자. 잘 진행하는가?  
그렇다,  $e \rightarrow error$ 로 진행할 수 없다. 그렇게 진행한다면  
모순: “Preservation Lemma”에 의해서  $error$ 가 타입이 있어야  
하는데  $error$ 의 타입을 결정하는 규칙은 없다.
- ▶ 그리고 진행이 타입을 항상 보존하므로, 값으로 진행이 끝나게  
되면 그 값도 같은 타입.

# 추론 규칙의 구현

타입에 대한 연립방정식 세우고 풀기

▶ 타입 연립 방정식  $u$

$$\begin{array}{ll} u \rightarrow \tau \doteq \tau & \text{타입 방정식} \\ | \quad u \wedge u & \text{연립} \\ \tau \rightarrow \alpha & \text{타입 변수} \\ | \quad \iota \mid \tau \rightarrow \tau \end{array}$$

## 타입 연립 방정식 세우기

$$V(\Gamma, e, \tau) = u$$

이랬으면 좋은:

$$S \models V(\Gamma, e, \tau) \Leftrightarrow S\Gamma \vdash e : S\tau.$$

여기서:

“ $S \models u$ ” = “ $S$ 는 방정식  $u$ 의 해(*model*)”  
 $S \in Subst = TyVar \xrightarrow{fin} Type$  이고  $Type$ 은  $\tau$ 들의 집합

이며

$$\frac{S\tau_1 = S\tau_2}{S \models \tau_1 \doteq \tau_2} \qquad \frac{S \models u_1 \quad S \models u_2}{S \models u_1 \wedge u_2}$$

이며

$$S\alpha = \begin{cases} \tau & \text{if } \alpha \mapsto \tau \in S \\ \alpha & \text{if } \alpha \notin \text{Dom } S \end{cases}$$

$$S\iota = \iota$$

$$S(\tau \rightarrow \tau') = (S\tau) \rightarrow (S\tau')$$

$$S\Gamma = \{x : S\tau \mid x : \tau \in \Gamma\}$$

## 타입 연립 방정식 세우기 $V(\Gamma, e, \tau)$

$$V(\Gamma, n, \tau) = \tau \doteq \iota$$

$$V(\Gamma, x, \tau) = \tau \doteq \tau' \quad \text{if } x : \tau' \in \Gamma$$

$$V(\Gamma, e_1 + e_2, \tau) = \tau \doteq \iota \wedge V(\Gamma, e_1, \iota) \wedge V(\Gamma, e_2, \iota)$$

$$V(\Gamma, \lambda x. e, \tau) = \tau \doteq \alpha_1 \rightarrow \alpha_2 \wedge V(\Gamma + x : \alpha_1, e, \alpha_2) \quad \text{new } \alpha_1, \alpha_2$$

$$V(\Gamma, e_1 e_2, \tau) = V(\Gamma, e_1, \alpha \rightarrow \tau) \wedge V(\Gamma, e_2, \alpha) \quad \text{new } \alpha$$

# $V(\Gamma, e, \tau)$ 는 옳은가?

즉,

$$S \models V(\Gamma, e, \tau) \quad \Leftrightarrow \quad S\Gamma \vdash e : S\tau$$

인가?

**Proof.**  $e$ 의 구조에 대한 귀납법으로.

$\lambda x.e$ 인 경우:  $S \models V(\Gamma, \lambda x.e, \tau)$  은

$$\begin{aligned} &= S \models \tau \doteq \alpha_1 \rightarrow \alpha_2 \wedge V(\Gamma + x : \alpha_1, e, \alpha_2) \quad \text{new } \alpha_1, \alpha_2 \\ &\Leftrightarrow S \models \tau \doteq \alpha_1 \rightarrow \alpha_2 \\ &\quad \wedge S \models V(\Gamma + x : \alpha_1, e, \alpha_2) \\ &\Leftrightarrow S\tau = S\alpha_1 \rightarrow S\alpha_2 \\ &\quad \wedge S\Gamma + x : S\alpha_1 \vdash e : S\alpha_2 \quad (\text{귀납가정}) \\ &\Leftrightarrow S\tau = S\alpha_1 \rightarrow S\alpha_2 \\ &\quad \wedge S\Gamma \vdash \lambda x.e : S\alpha_1 \rightarrow S\alpha_2 \\ &\Leftrightarrow S\Gamma \vdash \lambda x.e : S\tau. \end{aligned}$$

다른 경우도 비슷하게.

# 연립 방정식의 해 구하기

동일화 알고리즘(*unification algorithm*):

*"A Machine-Oriented Logic Based on the Resolution Principle"*, J.A.Robinson, Journal of ACM, Vol.12, No.1, pp.23-41, 1965.

- ▶ 타입 방정식들( $\tau \doteq \tau'$ )과 해 공간(*Type*)은 위 논문의 동일화 (*unification*) 알고리즘으로 풀 수 있는 클래스
- ▶ 알고리즘  $\mathcal{U}$ 는  $T \models u$ 인  $T$  중에서 가장 일반적인 해(*most general unifier*)를 구해준다.
  - ▶  $\mathcal{U}(u) \stackrel{\text{let}}{=} S \models u$  이고
  - ▶  $T \models u \Rightarrow T = RS$  인  $R$ 이 있다.

# 알고리즘 $\mathcal{U}(V(\Gamma, e, \alpha))$

$$\mathcal{U}(u) : Subst = TyVar \xrightarrow{\text{fin}} Type$$

이고

$$\begin{aligned}\mathcal{U}(\tau \doteq \tau') &= \text{unify}(\tau, \tau') \\ \mathcal{U}(u \wedge u') &= \text{let } S = \mathcal{U}(u) \\ &\quad S' = \mathcal{U}(S u') \\ &\quad \text{in } S' S\end{aligned}$$

이며, 동일화(*unification*) 알고리즘  $\text{unify}(\tau, \tau') : Subst$  는

$$\begin{aligned}\text{unify}(\tau, \tau) &= \emptyset \\ \text{unify}(\alpha, \tau) \text{ or } \text{unify}(\tau, \alpha) &= \begin{cases} \{\alpha \mapsto \tau\} & \text{if } \alpha \notin \tau \\ \text{fail} & \text{o.w.} \end{cases} \\ \text{unify}(\tau_1 \rightarrow \tau_2, \tau'_1 \rightarrow \tau'_2) &= \text{let } S = \text{unify}(\tau_1, \tau'_1) \\ &\quad S' = \text{unify}(S\tau_2, S\tau'_2) \\ &\quad \text{in } SS' \\ \text{unify}(-) &= \text{fail}\end{aligned}$$

## 알고리즘은 충실한(*sound & complete*) 구현

안전(*sound*)

$$\mathcal{U}(V(\Gamma, e, \alpha)) = S \Rightarrow S\Gamma \vdash e : S\alpha$$

완전(*complete*)

$$\left. \begin{array}{l} \mathcal{U}(V(\Gamma, e, \alpha)) = S \\ \wedge \Gamma' = RS\Gamma \\ \wedge \tau' = RS\alpha \end{array} \right\} \Leftarrow \Gamma' \vdash e : \tau'$$

# 다른 알고리즘 I: $V$ 와 $\text{unify}$ 를 동시에

$$M : \text{TyEnv} \times \text{Exp} \times \text{Type} \rightarrow \text{Subst}$$

$$M(\Gamma, n, \tau) = \text{unify}(\iota, \tau)$$

$$M(\Gamma, x, \tau) = \text{unify}(\tau, \tau') \quad \text{if } x : \tau' \in \Gamma$$

$$M(\Gamma, \lambda x. e, \tau) = \text{let } S = \text{unify}(\alpha_1 \rightarrow \alpha_2, \tau) \quad \text{new } \alpha_1, \alpha_2 \\ S' = M(S\Gamma + x : S\alpha_1, e, S\alpha_2) \\ \text{in } S'S$$

$$M(\Gamma, e e', \tau) = \text{let } S = M(\Gamma, e, \alpha \rightarrow \tau) \quad \text{new } \alpha \\ S' = M(S\Gamma, e', S\alpha) \\ \text{in } S'S$$

$$M(\Gamma, e + e', \tau) = \text{let } S = \text{unify}(\iota, \tau) \\ S' = M(S\Gamma, e, \iota) \\ S'' = M(S'S\Gamma, e', \iota) \\ \text{in } S''S'S$$

안전(*sound*)

$$M(\Gamma, e, \alpha) = S \Rightarrow S\Gamma \vdash e : S\alpha$$

완전(*complete*)

$$\left. \begin{array}{l} M(\Gamma, e, \alpha) = S \\ \wedge \Gamma' = RS\Gamma \\ \wedge \tau' = RS\alpha \end{array} \right\} \Leftarrow \Gamma' \vdash e : \tau'$$

## 다른 알고리즘 II: $V$ 와 $\text{unify}$ 를 동시에

$$W : \text{TyEnv} \times \text{Exp} \rightarrow \text{Type} \times \text{Subst}$$

$$W(\Gamma, n) = (\iota, \emptyset)$$

$$W(\Gamma, x) = (\tau, \emptyset) \text{ if } x : \tau \in \Gamma$$

$$W(\Gamma, \lambda x.e) = \text{let } (\tau, S) = W(\Gamma + x : \alpha, e) \text{ new } \alpha \\ \text{in } (S\alpha \rightarrow \tau, S)$$

$$W(\Gamma, e e') = \text{let } (\tau, S) = W(\Gamma, e) \\ (\tau', S') = W(S\Gamma, e') \\ S'' = \text{unify}(\tau' \rightarrow \alpha, S'\tau) \text{ new } \alpha \\ \text{in } (S''S'S, S''\alpha)$$

$$W(\Gamma, e + e') = \text{let } (\tau, S) = W(\Gamma, e) \\ S' = \text{unify}(\tau, \iota) \\ (\tau', S'') = W(S'\Gamma, e') \\ S''' = \text{unify}(\tau', \iota) \\ \text{in } (\iota, S'''S''S'S)$$

안전(sound)

$$W(\Gamma, e) = (\tau, S) \Rightarrow S\Gamma \vdash e : \tau$$

완전(complete)

$$\left. \begin{array}{l} W(\Gamma, e) = (\tau, S) \\ \wedge \Gamma' = RST \\ \wedge \tau' = R\tau \end{array} \right\} \Leftarrow \Gamma' \vdash e : \tau'$$