Theorem Problem

SNU 4541.664A Program Analysis

Spring 2006

Note 17

Prof. Kwangkeun Yi

다형 타입 시스템 Polymorphic Type System

- 다형 타입 추론
- 추론 규칙의 안전성
- 추론 규칙의 구현

타입 분석의 정확도 올리기

단순 타입 시스템(*simple type system*)이 "잘 모르겠다"고 하는 경우를 줄여보자.

- 타입 시스템에서는

  "잘 모르겠다" = "타입방정식의 해가 없다"

- 다형 타입 시스템(*polymorphic type system*)($\vdash_p$)에서는 "잘 모르겠다"고 하는 경우가 적다, 단순 타입 시스템(*simple type system*)($\vdash$) 보다:

$$\Gamma \vdash e : \tau \;\Rightarrow\; \Gamma \vdash_p e : \tau$$

- $\vdash_p$는 $\vdash$의 "conservative extension" 이라고 함.

부정확한 단순 타입 시스템

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

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{\{f:\tau\to\tau\}\vdash f:\tau\to\tau}\quad\dfrac{\vdots}{\{f:\tau\to\tau\}\vdash f:\tau}\ \tau=\tau\to\tau}{\{f:\tau\to\tau\}\vdash f\ f:\tau\to\tau}}{\vdash \lambda f.f\ f:(\tau\to\tau)\to(\tau\to\tau)}}{\vdash (\lambda f.f\ f)(\lambda x.x):\tau\to\tau}\quad\vdots$$

타입을 일반화(*type generalization*) 시키면?

타입(분석 결과 즉, 요약해석 공간의 원소)에 정교한 것이 포함됨:

$$\forall\alpha.\alpha \to \iota, \quad \forall\alpha_1,\alpha_2.\alpha_1 \to \alpha_2, \quad \cdots$$

그래서

$$\dfrac{\dfrac{\dfrac{\vdots}{\{f:\forall\alpha.\alpha\to\alpha\}\vdash f:(\iota\to\iota)\to(\iota\to\iota)} \quad \dfrac{\vdots}{\{f:\forall\alpha.\alpha\to\alpha\}\vdash f:\iota\to\iota}}{\{f:\forall\alpha.\alpha\to\alpha\}\vdash f\ f:\iota}}{\vdash \lambda f.f\ f:(\forall\alpha.\alpha\to\alpha)\to\iota}$$

이고

$$\dfrac{\dfrac{\vdots}{\vdash \lambda f.f\ f:(\forall\alpha.\alpha\to\alpha)\to(\iota\to\iota)} \quad \dfrac{\vdots}{\vdash \lambda x.x:\iota\to\iota}}{\vdash (\lambda f.f\ f)(\lambda x.x):\iota\to\iota}$$

하지만 함부로 일반화를 이용하면

- 안전하지 않을 뿐더러
- 완전(*complete*)한 구현이 불가능(*undecidable*):
  - 2단(*rank 2*)이상의 다형타입(*polymorphism*)은 피해야
  - 그러나 "완전한" 면을 포기하면 불가능할 것도 없음: 프로그램분석의 기본
- 아뭏튼, 1단(*rank 1*) 다형타입(*polymorphism*)까지만 이용
  - 0단(*rank 0*) = 단순타입(*monomorphic types*)
  - 1단(*rank 1*) = "$\forall$"이 제일 바깥(*prenex forma*)인 다형타입

n단 다형타입 σ^n(*rank-n polymorphism*)

$$\begin{array}{rcl} \tau, \sigma^0 & ::= & \iota \mid \tau \to \tau \mid \alpha \\ \sigma^{n+1} & ::= & \sigma^n \mid \forall \alpha.\sigma^{n+1} \\ & \mid & \sigma^n \to \sigma^{n+1} \end{array}$$

예)

$\iota \to \iota$	0단
$\forall \alpha.(\alpha \to \alpha)$	1단
$(\forall \alpha.\alpha) \to \iota$	2단
$((\forall \alpha.\alpha) \to \iota) \to \iota$	3단

함부로 일반화하면 불안전

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{\{f : \forall\alpha.\alpha \to \iota\} \vdash f : \iota \to \iota} \quad \cdots}{\{f : \forall\alpha.\alpha \to \iota\} \vdash f\ 1 : \iota} \quad \cdots}{\{f : \forall\alpha.\alpha \to \iota\} \vdash (f\ 1, f\ \mathsf{true}) : \iota \times \iota}}{\vdash \lambda f.(f\ 1, f\ \mathsf{true}) : (\forall\alpha.\alpha \to \iota) \to (\iota \times \iota)} \quad \dfrac{\vdots}{\vdash \lambda x.x + 1 : \iota \to \iota}}{\vdash (\lambda f.(f\ 1, f\ \mathsf{true}))(\lambda x.x + 1) : \iota \times \iota}$$

혹은

$$\dfrac{\vdots}{\vdash (\lambda x.(\mathtt{let}\ y = x\ \mathtt{in}\ (y\ 1, y\ \mathsf{true})))(\lambda z.z + 1) : \iota \times \mathit{bool}}$$

다형 타입의 의미 $\gamma(\forall\alpha.\tau)$?

값들의 집합

- 모든 타입 α에 대해서 τ 타입인 값들
- 예를들어, $\forall\alpha.\alpha \to \alpha$의 의미는: 인자타입에 상관없이 일을 하고 인자타입과 같은 타입의 값을 리턴하는 함수들의 집합
- 즉,

$$\gamma(\forall\alpha.\tau) = \bigcap_{t \in SimpleType} \gamma(\{t/\alpha\}\tau)$$

예를들어,

$$\begin{aligned}
\gamma(\forall\alpha.\alpha \to \alpha) &= \textstyle\bigcap_{t \in SimpleType} \gamma(t \to t) \\
&= \gamma(\iota \to \iota) \cap \gamma(bool \to bool) \cap \cdots \\
&= \{\lambda x.x, \lambda x.1, \lambda x.x + 1, \cdots\} \cap \\
&\quad \{\lambda x.x, \lambda x.(x || \mathsf{true}), \cdots\} \cap \cdots
\end{aligned}$$

안전한 분석 디자인: let-polymorphism 예

"Hindley-Milner style let-polymorphism"

- 프로그램이 특별히 생긴 경우만 그렇게 정교한 분석이 작동하도록
- 함수가 어디서 무슨 인자로 어떻게 사용되는 지를 알 수 있는 경우 즉,

$$(\lambda x.\underbrace{\cdots x \cdots x \cdots}_{e})e'$$

즉,

$$\text{let } x = e' \text{ in } e$$

인 경우만

- 이 경우, e'이 다형타입일 수 있는 지 "안전하게" 분석한 후에, e안에서 x가 어떻게 사용되는 지 분석.
- 다형타입은 1단(*rank-1 polymorphism*)까지만:

$$\iota \to \iota, \forall\alpha.\alpha \to \alpha, \forall\alpha_1,\alpha_2.\alpha_1 \to \alpha_2$$

다형 타입 추론

- 타입(*type*)과 타입틀(*type scheme*)

$$\begin{array}{rrcl} Type & \tau & \rightarrow & \iota \mid \tau \rightarrow \tau \mid \alpha \\ TypeScheme & \sigma & \rightarrow & \tau \mid \forall \alpha.\sigma \end{array}$$

  타입틀(*type scheme*)은 단순타입과 다형타입을 포함.
  다형타입은 1단(*rank-1*)까지만 (prenex form).
- 추론규칙(*inference rules*)은 "$\Gamma \vdash e : \tau$" 꼴을 유추하는 규칙들
- 가정들 Γ
  - 변수들의 타입틀(*type scheme*)에 대한 가정
  - $x + 1 : \iota$, 가정 $x : \iota$ 아래서.
  - $(f\ 1, f\ \mathsf{true}) : \iota \times bool$, 가정 $f : \forall \alpha.\alpha \rightarrow \alpha$ 아래서

$\Gamma \vdash e : \tau$를 추론하는 규칙

$$\frac{}{\Gamma \vdash n : \iota} \qquad \frac{}{\Gamma \vdash x : \tau}\ \sigma \succ \tau, x : \sigma\ \in \Gamma$$

$$\frac{\Gamma \vdash e : \tau \quad \Gamma + x : Gen_\Gamma(\tau) \vdash e' : \tau}{\Gamma \vdash \texttt{let}\ x = e\ \texttt{in}\ e' : \tau}$$

$$\frac{\Gamma \vdash e_1 : \iota \quad \Gamma \vdash e_2 : \iota}{\Gamma \vdash e_1 + e_2 : \iota}$$

$$\frac{\Gamma \vdash e_1 : \tau' \to \tau \quad \Gamma \vdash e_2 : \tau'}{\Gamma \vdash e_1\ e_2 : \tau}$$

$$\frac{\Gamma + x : \tau \vdash e : \tau'}{\Gamma \vdash \lambda x.e : \tau \to \tau'}$$

generalization $\quad Gen_\Gamma(\tau) = \forall \alpha_1, \cdots, \alpha_n.\tau \quad \{\alpha_1, \cdots, \alpha_n\} = FTV(\tau) \setminus FTV(\Gamma)$

instantiation $\quad \sigma \succ \tau \quad \sigma = \forall \alpha_1, \cdots, \alpha_n.\tau', \tau = \{\tau_i/\alpha_i\}_i \tau$

$$FTV(\tau) = TV(\tau)$$
$$FTV(\forall \alpha.\sigma) = FTV\sigma \setminus \{\alpha\}$$
$$FTV(\Gamma) = \cup_{x:\sigma \in \Gamma} FTV(\sigma)$$

추론 규칙의 안전성: $Gen_\Gamma(\tau)$

왜 타입 τ를 일반화 $\forall\alpha.\tau$ 시키는데 α가 Γ 에 나타나면 제외?

$Gen_\Gamma(\tau) = \forall\alpha_1, \cdots, \alpha_n.\tau$ 여기서 $\{\alpha_1, \cdots, \alpha_n\} = FTV(\tau) \backslash FTV(\Gamma)$

- Γ에 가정$(x : \sigma)$이 첨가 되는 경우는 $\lambda x.e$의 경우
- Γ에 있는 가정을 사용하는 경우는 함수안에서 함수의 인자를 분석할 때
- 함수의 인자 타입을 일반화시키고나서 함수 내부가 분석되면 않되
- 함수가 호출되면서 전달받는 실제인자는 일반화된 타입의 값이 아닐 수 있기 때문

예를 들어,

$$\lambda x.(\texttt{let}\ y = x\ \texttt{in}\ (y\ 1, y\ \texttt{true}))$$

이고 이 함수가 $\lambda z.z + 1$에 적용되면?

추론 규칙의 안전성 증명: 추론되는 대로 실행된다

타입이 있으면 문제없이 진행:

Theorem (Progress)

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

진행은 타입을 보존:

Theorem (Subject Reduction, Preservation)

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

추론 규칙의 구현

- 단순타입 유추 알고리즘 M과 W의 "자연스러운" 확장
- M_p나 W_p도 모두 충실한 구현. 예를들어,

안전(*sound*)

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

완전(*complete*)

$$\left.\begin{array}{l} W_p(\Gamma, e) = (\tau, S) \\ \wedge\ \Gamma' = RS\Gamma \\ \wedge\ R(Gen_{S\Gamma}(\tau)) \succ \tau' \end{array}\right\} \Leftarrow \quad \Gamma' \vdash e : \tau'$$

참고: "Proofs about a Folklore Let-Polymorphic Type Inference Algorithm", Oukseh Lee and Kwangkeun Yi, *TOPLAS*, 20(4), 1998