Judgmental Subtyping System with Intersection and Union Types

서정봉, POSTECH

라이트닝 목차

- 하고 싶은 것
- 재료
- 문제점
- 현재 접근법
- 질의 응답

하고 싶은 것(1): 보다 유연한 언어

- 잡종 데이터 모음

```
type dyn = MyInt of int | MyString of string

let ex_list : dyn list = [MyInt 1; MyString “hello world”]
```

- 타입 따라 '그때 그때 달라요' 함수

```
let inc : dyn → dyn = function
| MyInt x → MyInt (x + 1)
| MyString x → MyString (x^ “+”)

let result = List.map inc ex_list
(* val result : dyn list = [MyInt 2; MyString “hello world+”] *)
```

하고 싶은 것(1): 보다 유연한 언어

- 잡종 데이터 모음

  $\text{type dyn} = \text{int} \vee \text{string}$

  let ex_list : dyn list = [1; “hello world”]

- 타입 따라 '그때 그때 달라요' 함수

  $\text{let inc} : (\text{int} \to \text{int}) \wedge (\text{string} \to \text{string}) = \text{function}$
  $\mid \text{x:int} \to \text{x} + 1$
  $\mid \text{x:string} \to \text{x}\,\hat{}\,\text{“+”}$

  let result = List.map inc ex_list
  (* val result : dyn list = [2; “hello world+”] *)

하고 싶은 것(2): 보다 정교한 타입

- 프로그램 명세를 보다 정교히 표현하는 타입

두 문자열을 합치는 함수:

$$bitstring * bitstring \rightarrow bitstring$$

$$\Downarrow$$

결과로 나오는 값이 특정 조건(parity condition)을 만족함을 표현:

$$(even * even \rightarrow even) \wedge (even * odd \rightarrow odd)$$
$$\wedge (odd * even \rightarrow odd) \wedge (odd * odd \rightarrow even)$$

재료1: Intersection Types

- Introduction Rule

$$\frac{\Gamma \vdash M : A_1 \qquad \Gamma \vdash M : A_2}{\Gamma \vdash M : A_1 \wedge A_2} \wedge I$$

- Elimination Rules

$$\frac{\Gamma \vdash M : A_1 \wedge A_2}{\Gamma \vdash M : A_i} \wedge E_i \quad \text{or} \quad \frac{\Gamma \vdash M : A_1 \wedge A_2 \qquad A_1 \wedge A_2 \leq A_1}{\Gamma \vdash M : A_1} \; sub$$

- Subtyping

$$\frac{}{A_1 \wedge A_2 \leq A_i} \qquad \frac{A \leq C_1 \qquad A \leq C_2}{A \leq C_1 \wedge C_2}$$

재료2: Union Types

- Introduction Rules

$$\frac{\Gamma \vdash M : A_i}{\Gamma \vdash M : A_1 \vee A_2} \vee I_i \quad \text{or} \quad \frac{\Gamma \vdash M : A_1 \qquad A_1 \leq A_1 \vee A_2}{\Gamma \vdash M : A_1 \vee A_2} \; sub$$

- Elimination Rule

$$\frac{\Gamma \vdash M : A_1 \vee A_2 \qquad \begin{array}{c} \Gamma, x_1 : A_1 \vdash \mathcal{E}[x_1] : C \\ \Gamma, x_2 : A_2 \vdash \mathcal{E}[x_2] : C \end{array}}{\Gamma \vdash \mathcal{E}[M] : C} \vee E$$

- Subtyping

$$\frac{}{A_i \leq A_1 \vee A_2} \qquad \frac{A_1 \leq C \qquad A_2 \leq C}{A_1 \vee A_2 \leq C}$$

해결해야 할 문제점들

- 복잡해지는 서브타입 관계들

$$
\begin{aligned}
&A \wedge B \wedge C \leq B \wedge C \wedge A \\
&(A_1 \to C) \wedge (A_2 \to C) \leq (A_1 \vee A_2) \to C \\
&(A_1 \vee A_2) \to C \leq (A_1 \to C) \wedge (A_2 \to C) \\
&A \vee (B \wedge C) \leq (A \vee B) \wedge (A \vee C) \\
&\qquad\qquad\qquad\vdots
\end{aligned}
$$

- 어려운 계산법 정의
- Effect와 언어의 안전성

현재 접근법

- 타입 생성자의 의미를 내포하는 Subtyping judgment

$$A_1, \cdots, A_n \preceq B_1, \cdots, B_m$$

:= $A_1 \wedge \cdots \wedge A_n$ 타입을 가지는 코드는
$B_1 \vee \cdots \vee B_m$ 타입도 가진다.

- 새로운 Judgment를 이용한 시스템 설계
- Sequent calculus 스타일

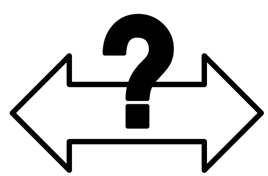
현재 접근법 (2)

- Subsumption 를 분리 또는 제약

$$\frac{\Gamma \vdash M : A \qquad A \leq B}{\Gamma \vdash M : B}\ sub$$

$$\frac{x : A \in \Gamma}{\Gamma \vdash x : A}\ \text{Var}$$

$$\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.\, M : A \to B}\ \to\text{I}$$



$$\frac{x : A \leq \{A_1, \cdots, A_n\} \in \Delta}{\Delta \vdash_{\leq} x : A_i}\ \text{Var}$$

$$\frac{\Delta, x : A \leq \{A_1, \cdots, A_n\} \vdash_{\leq} M : B}{\Delta \vdash_{\leq} \lambda x.\, M : A \to B}\ \to\text{I}$$

<Source language>

<Target language>