The Implicit Calculus

Wontae Choi

@ ROSAEC 2011 Summer Workshop

This work is collaboration with Wonchan Lee, Bruno Oliveira, Tom Schrijvers, and Kwangkeun Yi

명시적이지 않은 프로그래밍 언어

최원태

@ ROSAEC 2011년 여름 연수회

이 작업은 이원찬, 부르노, 톰, 이광근 교수님과 함께합니다.

Sorry. I'll use Korean from now.

목차

- 연구 동기
- 명시적이지 않은 프로그램이란
- 예제를 통한 언어 소개
- 겹침문제

연구배경

명시적이지 않은 프로그래밍을 이미 쓰고있다

- Haskell — Type Class
- Scala — Implicit Parameter
- C++ — Concept, Implicit Type Casting

해결되지 않은 겹침 문제

목표

해결되지 않은 겹침 문제

겹침문제가 깔끔하게 해결된
명시적이지 않은 프로그래밍 언어 만들기

명시적이지 않다?

밥을 달라는 경우

명시적인

"정확하게 X를 달라"

명시적이지 않은

"적당한 것을 달라"

명시적이지 않다?

프로그램으로 옮기면

명시적인 프로그램

"정확하게 X를 달라"

```
void 배고프다( ){
    접시1 = 끓여라(신라면);
    접시2 = 꺼내라(묵은지);
}
```

명시적이지 않은 프로그램

"적당한 것을 달라"

```
void 배고프다( ){
    접시1 = ?먹거리;
    접시2 = ?찬거리;
}
```

명시적이지 않은 프로그래밍

"타입을 줄테니 적당한 것을 달라"

```
void 배고프다( ){
    접시1 = ?먹거리;
    접시2 = ?찬거리;
}
```

+ 어떻게 만드는지(규칙)알려줘야한다

우리의 접근방법

"규칙타입을 줄테니 적당한 것을 달라"

```
void 배고프다( ){
    접시1 = ?{}=>먹거리;
    접시2 = ?{}=>찬거리;
}
```

+ 기반이 되는 규칙을 알려줘야한다

규칙 타입

$$\begin{array}{lll} RuleType & \rho := & \pi \Rightarrow \tau \\ Ctx & \pi \in & 2^{\rho} \end{array}$$

"타입 π 인 규칙집합을 넘겨주면 τ 타입인 값을 계산"

수도꼭지 : { } => 물

라면봉지 : { } => 면

국수끓여 : {{}=>물, {}=>면} => 먹거리

규칙 질의

환경이 다음과 같을 때

수도꼭지 :{ } => 물
라면봉지 :{ } => 면

국수끓여 :{{}=>물, {}=>면} => 먹거리

다음과 같이 질의

?({}=>먹거리)

“다른 재료 없이, 먹을거 만들어줄 수 있어?”

규칙 질의

환경이 다음과 같을 때

수도꼭지 :{ } => 물
라면봉지 :{} => 면

국수끓여 :{{}=>물, {}=>면} => 먹거리

다음과 같이 질의

?({}=>먹거리)

규칙 질의

환경이 다음과 같을 때

수도꼭지 :{ } => 물
라면봉지 :{} => 면

국수끓여 :{{}=>물, {}=>면} => 먹거리

다음과 같이 질의

?({}=>먹거리)

규칙 질의

환경이 다음과 같을 때

수도꼭지 :{ } => 물
라면봉지 :{} => 면

국수끓여 :{{}=>물, {}=>면} => 먹거리

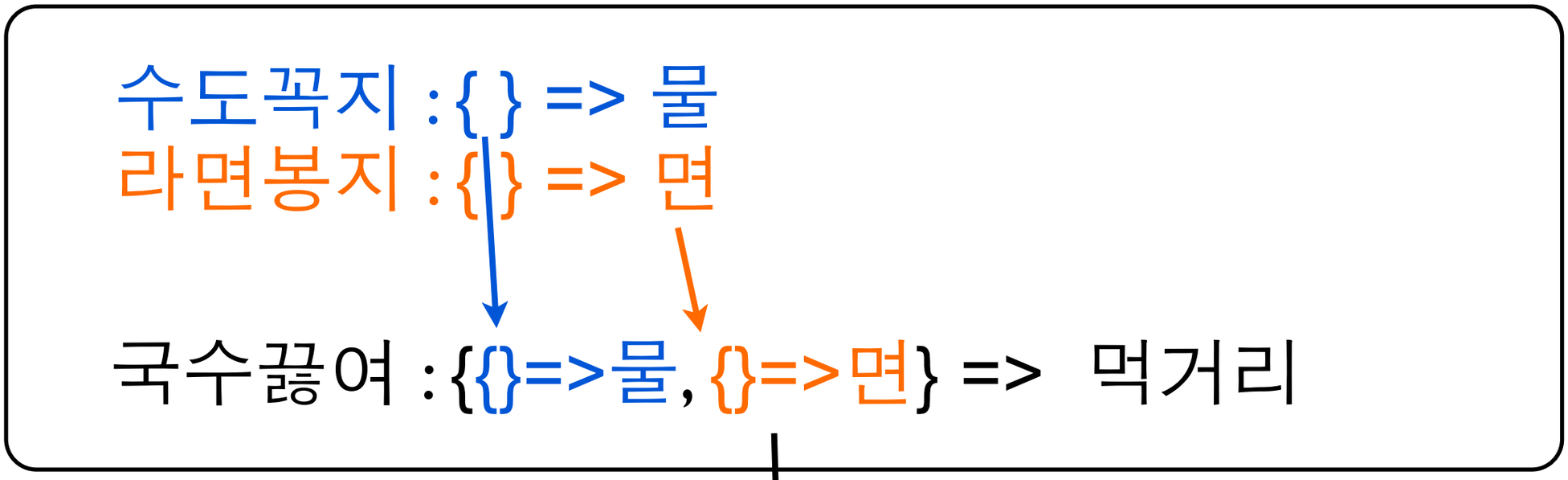
다음과 같이 질의

?({}=>먹거리)

“응. 먹을거 만들어줄 수 있어”

규칙 사용

환경이 다음과 같을 때



질의&사용

(?{}=>먹거리) with {}

“여기 라면 있다”

명시적이지 않은 언어 λ_ρ

부제 : 규칙을 만들고 사용하는 방법에 대한 언어

$Type$	$\tau :=$	α		
	$\mid$	$\forall \vec{a}.\pi \Rightarrow \tau$	$(= \rho)$	규칙 타입
Ctx	$\pi \in$	2^ρ		
$Expr$	$e :=$	$\texttt{rule}\ (\forall a.\pi \Rightarrow \tau)\ e$		규칙 정의
	$\mid$	$e\ \texttt{with}\ \{\rho_i : e_i\}$		규칙 사용
	$\mid$	$?\rho$		규칙 요구

단순타입 λ_ρ의 문법

예제(단순타입)

- 규칙 정의

$$\texttt{rule}\ (\{\} \Rightarrow \texttt{int})\ 1$$

- 규칙 사용

$$(\texttt{rule}\ (\{\} \Rightarrow \texttt{int})\ 1)\ \texttt{with}\ \{\} \quad \rightsquigarrow \quad 1$$

- 규칙 질의

$$\texttt{rule}\ (\{\rho\} \Rightarrow \rho)\ ?\rho$$

$$\texttt{rule}\ (\{\rho_1, \{\rho_1\} \Rightarrow \rho_2\})\ (?\{\} \Rightarrow \rho_2)$$

$$\texttt{rule}\ (\{\rho_1, \{\rho_1, \rho_2\} \Rightarrow \rho_3\})\ (?\{\rho_2\} \Rightarrow \rho_3)$$

다형타입 확장

$$
\begin{array}{lrll}
\mathit{Type} & \tau := & \alpha & \\
 & \mid & a & \\
 & \mid & \forall \vec{a}.\pi \Rightarrow \tau & (= \rho) \\
 & & & \\
\mathit{Ctx} & \pi \in & 2^{\rho} & \\
 & & & \\
\mathit{Expr} & e := & \texttt{rule}\ (\forall a.\pi \Rightarrow \tau)\ e & \\
 & \mid & e\ \texttt{with}\ \{\rho_i : e_i\} & \\
 & \mid & ?\rho & \\
 & \mid & e[\vec{\tau}] &
\end{array}
$$

다형타입 λ_{ρ}의 문법

유한한 규칙, 무한한 타입

예제

$$\{\forall a.\{\{\} \Rightarrow a\} \Rightarrow a\ \texttt{list},\ \{\} \Rightarrow \texttt{int}\}$$

처리가능한 질의

$$\{\} \Rightarrow \texttt{int}$$
$$\{\} \Rightarrow \texttt{int list}$$
$$\{\} \Rightarrow \texttt{int list list}$$
$$\vdots$$

유한한 규칙, 무한한 타입

예제

$$\{\forall a.\{\{\} \Rightarrow a\} \Rightarrow a\ \texttt{list},\ \{\} \Rightarrow \texttt{int}\}$$

처리가능한 질의

$$\{\} \Rightarrow \texttt{int}$$
$$\{\} \Rightarrow \texttt{int list}$$
$$\{\} \Rightarrow \texttt{int list list}$$
$$\vdots$$

아뿔싸 겹침문제가!

겹침문제

규칙이 겹치는 경우

$\{\forall\alpha.\alpha \to \alpha, \texttt{int} \to \texttt{int}\} \vdash_r \texttt{int} \to \texttt{int}$

De-facto Standard

제일 비슷한 것을 고른다

겹침문제

De-facto Standard

제일 비슷한 것을 고른다

고르기 애매한 경우도 있다

$$\{\forall\alpha.\alpha \to \texttt{int}, \forall\alpha.\texttt{int} \to \alpha\} \vdash_r \texttt{int} \to \texttt{int}$$

타입검사는 의미가 애매한 프로그램을 잡아야

겹침문제에 대한 해법

Haskell C++	가장 비슷한 것을 고른다 (애매한 경우가 많다)
Scala	비슷하고 가까운곳에서 정의된 것을 고른다 (고르는 기준이 애매하다)
우리 해법	거리만 생각한다 거리가 같고 겹치는 경우를 허용하지 않는다

"거리가 같고 겹치는 경우는 많지 않을것"

현재 상황

- “타입을 줄테니 값을 달라” 를 지원하는 다형타입 규칙 언어의 실행의미와 타입검사를 만들었다.
- 타입이 맞는 프로그램은 겹칩문제가 없다!

정의 (아름다운)

정의 (아름다운)

은(는) 포스터 세션에서

그 밖의 중요한 사항

타입검사가 끝나지 않을 가능성

알고리즘 제시

구현 가능한 언어인가?

system F로의 변환 제시

Haskell, Scala, C++을 포괄하는가?

Haskell + Scala 예제가 우리 언어로 변환됨

질문 있으신가요?