

# COERCION AND TYPE INFERENCE (SUMMARY)

- Artikeln handlar om automatisk koersion, typkonvertering
- Typsystem med fyra regler för typhärdning
- Koersion medför icke-komplett typsystem
- Semantiskt komplett med en femte regel
- Algoritm för typkontroll

## Introduktion

$$\begin{aligned} \llbracket x \rrbracket p &= p(x) \\ \llbracket e_1 e_2 \rrbracket p &= \text{Fun}(\llbracket e_1 \rrbracket)(\llbracket e_2 \rrbracket) \\ \llbracket \lambda x.e \rrbracket p &= \text{Graph}(\text{"funktionens värde vid } d \in D \text{ är } \llbracket e \rrbracket p(d/x)\text{"}) \end{aligned}$$

datam definierad enligt

Givet en lambda-modell och en variabelomgivning  $p$ , så är betydelsen av en lambda-

ekvivalensklasser. Applikation definieras som  $\text{Fun}([d])([e]) = [de]$ .

$(D, \text{Fun}, \text{Graph})$  med en mängd ("domän")  $D$  och mappningarna  $\text{Fun} : D \rightarrow [D \rightarrow D]$  och  $\text{Graph} : [D \rightarrow D] \rightarrow D$ . Den modell som används är *termmodellen*, byggd på Vanliga uttryck lambda-kalkyl,  $c ::= x \mid e_1 e_2 \mid \lambda x.e$ . En lambda-modell är en trippel

## Lambda-modeller

## Reduktionsregler

$$\begin{array}{l}
 (\alpha) \quad \lambda x.e \rightarrow \lambda y.e([y/x]) \quad \text{om } y \text{ inte fri i } e \\
 (\beta) \quad (\lambda x.e) f \rightarrow e[f/x] \\
 (\eta) \quad \lambda x.ex \rightarrow e \quad \text{om } x \text{ inte fri i } e
 \end{array}$$

En term som inte kan reduceras är i *normalform*.

## Typuttryck, koersionsmängder

$d \subseteq \tau$  innebär att uttryck av typen  $d$  kan kohäraseras (?) till uttryck av typen  $\tau$ .

$$\llbracket a \rrbracket \eta = \eta(a)$$

$$\llbracket \sigma \rightarrow \tau \rrbracket \eta = \{d \mid \forall d_1 \in \llbracket \sigma \rrbracket \eta, \text{Fun}(d)(d_1) \in \llbracket \tau \rrbracket \eta\}$$

En koersionsmängd är en mängd av koersioner  $\alpha \subseteq \beta$  där  $\alpha, \beta$  är atomära typer. En modell och typomgivning satisfierar en koersionsmängd  $C$  om

$$\llbracket \alpha \rrbracket \eta \subseteq \llbracket \beta \rrbracket \eta \text{ för alla } \alpha \subseteq \beta \in C$$

En typindelning  $A$  mappar variabler till typuttryck, och satisfieras av en modell, omgivning och typomgivning om

$$d(x) \in \llbracket \sigma \rrbracket \eta \text{ närligst } A(x) = \sigma$$

## Typinferens, regler

$C, A \mid e : \sigma$  innebär givet kohersionerna  $C$  och tilldelningen  $A$  att  $e$  har typen  $\sigma$ .

$$\text{(var)} \quad C, A \mid x : \sigma \quad \text{om } A(x) = \sigma$$

$$\text{(app)} \quad \frac{C, A \mid e_1 : \sigma \rightarrow \tau \quad C, A \mid e_2 : \sigma}{C.A \mid e_1 e_2 : \tau}$$

$$\text{(abs)} \quad \frac{C, A \mid \lambda x. e : \sigma \rightarrow \tau}{C, A[\sigma/x] \mid e : \tau}$$

$$\text{(coerce)} \quad \frac{C, A \mid e : \tau}{C, A \mid e : \sigma} \quad C \mid \sigma \sqsubseteq \tau$$

$$\text{(equal)} \quad \frac{C, A \mid e : \sigma \quad C, A \mid f : \sigma}{C, A \mid e = f}$$

# Inferensregler för konsekvenser av coercion

Coercion and Type Inference (Summary)

Author: John C. Mitchell

Exempel:

$$\frac{\text{int} \sqsubseteq \text{real} \quad \text{real} \sqsubseteq \text{complex}}{\text{int} \sqsubseteq \text{complex}}$$

(trans)

$$\frac{\sigma \sqsubseteq \tau \quad \tau \sqsubseteq \rho}{\sigma \sqsubseteq \rho}$$

(arrow)

$$\frac{\sigma_1 \sqsubseteq \sigma \quad \tau \sqsubseteq \tau_1 \quad \sigma \rightarrow \tau \quad \sigma_1 \rightarrow \tau_1}{\sigma_1 \rightarrow \tau_1}$$

som kan typas, alltså inkomplett system utan ekvivalensregeln.

$$\lambda y. y$$

har subterm utan normalform, men är semantiskt ekvivalent med

$$(\lambda x. \lambda y. y)(\lambda x. x x \lambda x. x x)$$

## Icke-kompletthet map. semantiken

- Baserad på de fyra första reglerna
- Kan lägga in typkonversioner och anrop vid kompileringstillfället
- En term har en typ endast om alla subtermer har en typ

## Algorithm för typkontroll