~~Types,~~ Abstraction, ~~and Parametric Polymorphism~~

John C. Reynolds
Presented by Dietrich Geisler

Abstraction

Dietrich Geisler

(With apologies to John C. Reynolds)

What is Abstraction?

1. Define complex numbers
2. When are they equal?

Professor Descartes

1. Pairs of real numbers

2. Equality of components

1. Pairs of real numbers; first component is nonnegative

2. Equality of first component AND second component differs by multiple of 2π

Professor Bessel

Some Context

Published in 1983

Previous Papers:

Recursive Functions (1960)
Axiomatic Basis (1969)
CBN and CBV (1975)

Intel 80286 Processor:

10 MHz clock rate
No memory cache

Higher-level languages:

Scheme	1973
ML	1975
C++	1980

Sets and Types

If $e_1 \in E_{\pi,\omega\to\omega'}$, and $e_2 \in E_{\pi\omega}$ then

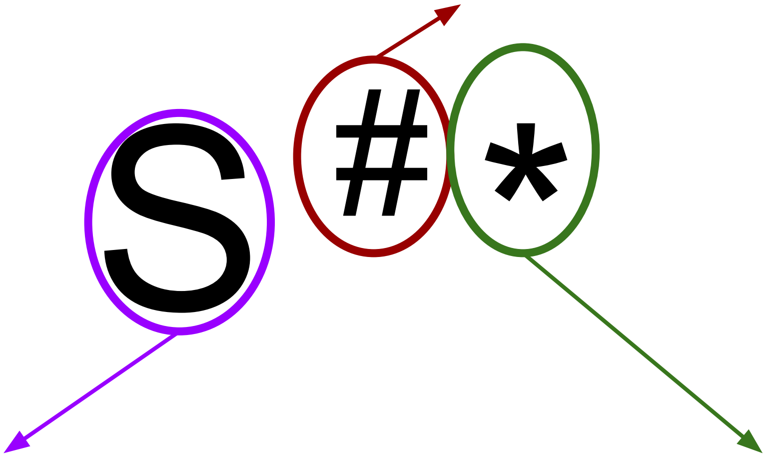
$$e_1(e_2) \in E_{\pi\omega'},$$

If e_1 has type $\omega\to\omega'$ and e_2 has type ω

Then the result of applying e_1 to e_2 has type ω'

Some Notation

Extension to constants, pairs, and functions
e.g. $S^{\#} (\omega \times \omega') = S^{\#}\omega \times S^{\#}\omega'$



Set Assignment
(e.g. $S(\tau) = \{\mathbf{0}, \mathbf{1}, \mathbf{2}\}$)

Extension to a context
(Works pointwise over the map)

$$S^{\#*}\pi = \prod_{v \,\varepsilon\, \mathrm{dom}\, \pi} S^{\#}(\pi v) \,.$$

Some Semantics

If $k \in K_{\omega}$

then $\mu_{\pi\omega}[k]\ S\ \eta = \alpha_{\omega}\ k$

$$\frac{\eta \vdash k : \omega}{\eta \vdash \alpha_{\omega}(k)}$$

If $v \in \mathrm{dom}\ \pi$

then $\mu_{\pi,\pi v}[v]\ S\ \eta = \eta\ v$

$$\frac{\eta \vdash v : \omega}{\eta \vdash \eta(v)}$$

Semantics of Pairs

If $e \in E_{\pi\omega}$ and $e' \in E_{\pi\omega'}$, then

$$\mu_{\pi,\omega\times\omega'}[\![<e, e'>]\!] S\ \eta = <\mu_{\pi\omega}[\![e]\!] S\ \eta,\ \mu_{\pi\omega'}[\![e']\!] S\ \eta>$$

$$\frac{\eta \vdash e_1 : \omega \qquad \eta \vdash e_2 : \omega'}{\eta \vdash < e_1, e_2 >: \omega \times \omega'}$$

How to compare set assignments?

Sets are related using pairs of set elements under **Rel**(s_1, s_2)

Functions and pairs are related if each component is related

R is the pointwise relation between two set interpretations of types S_1, S_2

What is an Abstraction? (Formally)

<u>Abstraction Theorem</u> Let R be a relation assignment between set assignments S_1 and S_2. For all $\pi \in \Omega^*$, $\omega \in \Omega$, $e \in E_{\pi\omega}$, and $\langle \eta_1, \eta_2 \rangle \in R^{\#*}\pi$,

$$\langle \mu_{\pi\omega} [\![e]\!] S_1 \eta_1, \mu_{\pi\omega} [\![e]\!] S_2 \eta_2 \rangle \in R^{\#}\omega \, .$$

Evaluating expressions maps related arguments to related results

Extending this to a Typing Theorem

<u>Pure Type Definition Theorem</u> Let S be a set assignment, ω_1, ω_2 ε Ω, and r be a relation between $S^{\#}\omega_1$ and $S^{\#}\omega_2$. For all π ε Ω^*, τ ε T, ω' ε Ω, e ε $E_{\pi-\tau,\omega'}$, and η ε $S^{\#*}\pi$,

$$\langle \mu_{\pi,(\omega'/\tau\to\omega_1)}[\![\underline{\text{lettype}}\ \tau = \omega_1\ \underline{\text{in}}\ e]\!]\ S\ \eta,$$
$$\mu_{\pi,(\omega'/\tau\to\omega_2)}[\![\underline{\text{lettype}}\ \tau = \omega_2\ \underline{\text{in}}\ e]\!]\ S\ \eta\rangle$$
$$\varepsilon\ [IA \mid \tau: r]^{\#}\ \omega'\ ,$$

where IA is the relation assignment such that IA τ = I(S τ) for all τ ε T.

What Happened to this work?

Some was folded into System F

Rust is starting to use some relational proofs

Ideas behind free theorems (e.g. properties $\lambda f : \alpha \to \alpha$?)