Functional Programming

Theorems for Free!

Jevgeni Kabanov

Department of Computer Science
University of Tartu

Universal Types

Outline

Universal types introduce types as first-class language members:

- Type parametrization

$$\mathsf{double} = \Lambda X.\ \lambda f^{X \to X}.\ \lambda a^{X}.\ f\ (f\ a)$$

- Type application

$$\mathsf{double}\ [\mathsf{Nat}]\ (\lambda x^{\mathsf{Nat}}.\ (x + 2))\ 1$$

- Typing

$$\mathsf{double} : \forall X.(X \to X) \to X \to X$$

Lambda calculus with universal types is called System F. We will also use the notation $\mathsf{double}_{\mathsf{Nat}}$ to denote a parametrized function.

Universal Types

Outline

- Universal types are more powerful than Hindley-Milner types
- However they cannot be inferred and need to be provided by the programmer
- This makes them less than comfortable in real life
- Haskell type system has System F extensions
- After type inference stage GHC translates every program to a simpler language based on System F

Deriving theorems

Example

Say r is a function of type

$$r : \forall X. X^{\star} \rightarrow X^{\star}$$

Where $X^{\star}$ is a list of Xs. Then we can derive that for all types A and A' and for all total functions $a : A \rightarrow A'$ we have:

$$a^{\star} \circ r_A = r_{A'} \circ a^{\star}$$

Where $\_^{\star}$ is equivalent to Haskell $map :: (a \rightarrow b) \rightarrow [\,a\,] \rightarrow [\,b\,]$.

Types as sets

Definition

We can interpret any type as a corresponding set:

- Nat is $\{n \mid n \in 0 \ldots\}$ and Bool $= \{\text{True}, \text{False}\}$.
- If A and B are types then $A \times B = \{(a, b) \mid a \in A \wedge b \in B\}$.
- If A is a type then $A^\star$ is a set of lists with elements from A.
- $A \rightarrow B$ is a set of functions from A to B.
- If X is a type variable and $A(X)$ is a type dependent on X then the type $\forall X.A(X)$ is a set of functions that take a set B and return an element of $A(B)$.

Relations

Definition

Let's recall relations:

- If A and A' are sets, we write $\mathcal{A} : A \sim A'$ to show that $\mathcal{A}$ is a relation between A and A', that is $\mathcal{A} \subseteq A \times A'$.
- We write $(x, y) \in \mathcal{A}$ if x and y are related by $\mathcal{A}$.
- Identity relation $\mathsf{Id}_A : A \sim A$ is defined as $\mathsf{Id}_A = \{(x, x) \mid x \in A\}$, in other words $(x, y) \in \mathsf{Id}_A \equiv x = y$.
- Any function $a : A \to A'$ can be interpreted as a relation $a = \{(x, a\ x) \mid x \in A\}$, in other words $(x, x') \in a \equiv a\ x = x'$.

Identity and Cartesian

Definition

We can also interpret any type as a corresponding relation:

- Constant types are identity relations, $\mathsf{Id}_{\mathsf{Bool}} : \mathsf{Bool} \sim \mathsf{Bool}$, $\mathsf{Id}_{\mathsf{Nat}} : \mathsf{Nat} \sim \mathsf{Nat}$.
- For any relations $\mathcal{A} : A \sim A'$ and $\mathcal{B} : B \sim B'$ relation $\mathcal{A} \times \mathcal{B} : (A \times B) \sim (A' \times B')$ is defined as

$$((x, y), (x', y')) \in \mathcal{A} \times \mathcal{B} \equiv (x, x') \in \mathcal{A} \wedge (y, y') \in \mathcal{B}$$

If a and b are functions then $(a \times b)\ (x, y) = (a\ x, b\ y)$.

Lists

Definition

For any relation $\mathcal{A} : A \sim A'$ the relation $\mathcal{A}^\star : A^\star \sim A'^\star$ is defined as

$$([x_1, \ldots, x_n], [x'_1, \ldots, x'_n]) \in \mathcal{A}^\star \equiv \\ (x_1, x'_1) \in \mathcal{A} \wedge \ldots \wedge (x_n, x'_n) \in \mathcal{A}$$

If a is a function then $a^\star$ is a map defined by $a\ [x_1, \ldots, x_n] = [a\ x_1, \ldots, a\ x_n]$.

Functions

Definition

For any relation $\mathcal{A} : A \sim A'$ and $\mathcal{B} : B \sim B'$ relation $\mathcal{A} \to \mathcal{B} : (A \to B) \sim (A' \to B')$ is defined as

$$(f, f') \in \mathcal{A} \to \mathcal{B} \equiv \forall (x, x') \in \mathcal{A} : (f\ x, f'\ x') \in \mathcal{B}$$

If a and b are functions, then $a \to b$ is not necessarily a function, but

$$\begin{aligned}(f, f') \in a \to b &\equiv a\ x = x' \wedge b\ (f\ x) = f'\ x' \\ &\equiv b\ (f\ x) = f'\ (a\ x) \\ &\equiv f' \circ a = b \circ f\end{aligned}$$

Universal types

Definition

Let $\mathcal{F}(\mathcal{X})$ be a relation depending on $\mathcal{X}$. The $\mathcal{F}$ corresponds to a function from relations to relations, so that for each $\mathcal{A} : A \sim A'$ there exists $\mathcal{F}(\mathcal{A}) : F(A) \sim F'(A')$. Then the relation $\forall \mathcal{X}.\mathcal{F}(\mathcal{X}) : \forall X.F(X) \sim \forall X'.F'(X')$ is defined as:

$$(g, g') \in \forall \mathcal{X}.\mathcal{F}(\mathcal{X}) \equiv \forall \mathcal{A} : A \sim A', (g_A, g'_{A'}) \in \mathcal{F}(\mathcal{A})$$

Parametricity

Theorem: Parametricity

If t is a closed term of type T, then $(t, t) \in \mathcal{T}$, where $\mathcal{T}$ is the relation corresponding to the type T.

Rearrangement theorem

Derivation

Let r be a closed term of type $\forall X.X^\star \to X^\star$. Parametricity gives that $(r, r) \in \forall \mathcal{X}.\mathcal{X}^\star \to \mathcal{X}^\star$. By definition of $\forall$ on relations it is equivalent to

$$\forall \mathcal{A} : A \sim A', (r_A, r_{A'}) \in \mathcal{A}^\star \to \mathcal{A}^\star$$

By definition of $\to$ on relations:

$$\forall \mathcal{A} : A \sim A', \forall (xs, xs') \in \mathcal{A}^\star, (r_A\ xs, r_{A'}\ xs') \in \mathcal{A}^\star$$

Let's restrict $\mathcal{A}$s to be functions $a : A \to A'$ and specialize:

$$\begin{aligned} &\forall a \forall xs : xs' = a^\star\ xs \Rightarrow a^\star\ (r_A\ xs) = r_{A'}\ xs' \\ &\equiv \forall a : a^\star\ (r_A\ xs) = r_{A'}\ (a^\star\ xs) \\ &\equiv \forall a : a^\star \circ r_A = r_{A'} \circ a^\star \end{aligned}$$

Map theorem

Derivation

Let m be a closed term of type $\forall X.\forall Y.(X \to Y) \to (X^\star \to Y^\star)$. Parametricity gives that $(m, m) \in \forall \mathcal{X}.\forall \mathcal{Y}.(\mathcal{X} \to \mathcal{Y}) \to (\mathcal{X}^\star \to \mathcal{Y}^\star)$. Taking $\mathcal{X} = a, \mathcal{Y} = b$ and applying definition of $\forall$ twice we get:

$$\forall a \forall b : (m_{AB}, m_{A'B'}) \in (a \to b) \to (a^\star \to b^\star)$$

Further applying the definition of $\to$:

$$\begin{aligned}
&\forall a \forall b \forall (f, f') \in (a \to b) : (m_{AB}\ f, m_{A'B'}\ f') \in (a^\star \to b^\star) \\
&\qquad \equiv \forall a \forall b : f' \circ a = b \circ f \Rightarrow (m_{AB}\ f, m_{A'B'}\ f') \in (a^\star \to b^\star) \\
&\qquad \equiv \forall a \forall b : f' \circ a = b \circ f \Rightarrow m_{A'B'}\ f' \circ a^\star = b^\star \circ m_{AB}\ f
\end{aligned}$$

Map corollary

Derivation

Out previous result was for all a and b

$$f' \circ a = b \circ f \Rightarrow m_{A'B'}\ f' \circ a^{\star} = b^{\star} \circ m_{AB}\ f$$

Taking $A' = B' = B$, $b = f' = \mathsf{Id}_B$, $a = f$ we get

$$\mathsf{Id}_B \circ f = \mathsf{Id}_B \circ f \Rightarrow m_{BB}(\mathsf{Id}_B) \circ f^{\star} = (\mathsf{Id}_B)^{\star} \circ m_{AB}(f)$$

The premiss is obviously a tautology and since $(\mathsf{Id}_B)^{\star} = \mathsf{Id}_{B^{\star}}$ the result can be rewritten as

$$m_{AB}(f) = m_{BB}(\mathsf{Id}_B) \circ f^{\star}$$

Which means that any function m of type $\forall X.\forall Y.(X \to Y) \to (X^{\star} \to Y^{\star})$ is equivalent to map up to element rearrangement.

Sort and Dup theorem

Derivation

Let s be of type $\forall X.(X \to X \to \mathsf{Bool}) \to (X^\star \to X^\star)$ (examples are **sort** that sorts the list after a ordering and **dup** that removes adjacent dublicates after equivalence). Then for all a:

$$\forall x, y \in A : x \prec y = a\ x \prec'\ a\ y \Rightarrow a^\star \circ s_A\ (\prec) = s_{A'}\ (\prec') \circ a^\star$$

For **sort** this means that map commutes with sort if f preserves ordering:

$$\forall x, y \in A : x < y = a\ x <'\ a\ y \Rightarrow a^\star \circ s_A\ (<) = s_{A'}\ (<') \circ a^\star$$

For **dup** this means that map commutes with dup if f preserves equivalence:

$$\forall x, y \in A : x \equiv y = a\ x \equiv'\ a\ y \Rightarrow a^\star \circ s_A\ (\equiv) = s_{A'}\ (\equiv') \circ a^\star$$

Fold theorem

Derivation

fold type is $\forall X.\forall Y.(X \to Y \to Y) \to Y \to X^\star \to Y$. Applying the definition of $\forall$ twice and specializing to functions $a : A \to A'$, $b : B \to B'$:

$$(\mathbf{fold}_{AB}, \mathbf{fold}_{A'B'}) \in (a \to b \to b) \to b \to a^\star \to b$$

Applying definition of $\to$ twice we get that for all $(\oplus, \oplus') \in (a \to b \to b)$:

$$u' = b\ u \Rightarrow (\mathbf{fold}_{AB}\ (\oplus)\ u, \mathbf{fold}_{A'B'}\ (\oplus')\ u') \in a^\star \to b$$

Whereas $\forall (\oplus, \oplus') \in (a \to b \to b)$ can be interpreted as

$$\forall x \in A, \forall y \in B : b\ (x \oplus y) = (a\ x) \oplus' (b\ y)$$

Fold theorem

Derivation

The resulting theorem for **fold** looks like:

$$\forall x \in A, \forall y \in B : b\ (x \oplus y) = (a\ x) \oplus' (b\ y) \wedge u' = b\ u$$
$$\implies b \circ \mathbf{fold}_{AB}\ (\oplus)\ u = \mathbf{fold}_{A'B'}\ (\oplus')\ u' \circ a^{\star}$$

Although it seems complicated, it states that if a and b provide a homomorphism between algebra structures $(A, B, \oplus, u)$ and $(A', B', \oplus', u')$ then $a^{\star}$ and b provide a homomorphism between algenra structures $(A^{\star}, B, \mathbf{fold}_{AB}(\oplus)u)$ and $(A', B', \mathbf{fold}_{A'B'}(\oplus')u')$.
Similarly to map we can prove that every function f of fold type can be expressed as:

$$f_{AB}\ c\ n = \mathbf{fold}_{AB}\ c\ n \circ f_{AA^{\star}}\ \mathrm{cons}_A\ \mathrm{nil}_A$$

Finally

Outline

- Parametricity breaks in the presence of fixpoint combinator, it needs additionally for qualified functions to be strict ($f\bot = \bot$). Since Haskell provides recursive definitions this must be taken into account.
- Since every polymorphic type gives rise to a theorem, this approach can yield a lot more results, though most of them are less useful.
- It can also help to make steps in some more powerful theorem, only requiring parametricity (e.g. Hindley/Milner to Girard/Reynolds type system isomorphism).
- Theorems can be (and are) generated completely automatically! Try `http://haskell.as9x.info/cgi-bin/ftonline.pl`