3 Liquid Types

3.1 Defining the Type System

First, we define some notations:

- $\mathbb{N}$ are the natural numbers starting from 1.
- $\mathbb{N}_0$ are the natural numbers starting from 0.
- $\mathbb{N}_a^b := \{i \in \mathbb{N}_0 | a \leq i \land i \leq b\}$ are the natural numbers between a and b.
- We'll use "." to separate a quantifier from a statement: $\forall a.b$ and $\exists a.b$.
- Functions will be written as $a_1 \to \cdots \to a_n \to b$ instead of $a_1 \times \cdots \times a_n \to b$.

For this thesis we will use a Hindley-Milner type system [DM82].

Axiom 3.1: Types

Let $n \in \mathbb{N}; \quad \forall i \in \mathbb{N}_a^b.k_i \in \mathbb{N}_0; \quad \forall i \in \mathbb{N}_a^b.C_i$ be a unique symbol.

—

T is a *type* if either

- (*Type variable*) T is a symbol.
- (*Type application*) T has the form $T ::= C_1\, T_{1,1} \dots T_{1,k_1} \mid \dots \mid C_n\, T_{n,1} \dots T_{n,k_n}$

  where $\forall i \in \mathbb{N}_1^n \forall j \in \mathbb{N}_0^{k_i}.T_{i,j}$ is a type application or type variable.
- (*Quantified type*) T has the form $T ::= \forall a_1 \dots \forall a_n.T'$

  where T' is a type application or a type variable and $\forall i \in \mathbb{N}_1^n.a_i$ is a type variable.

  For applied quantified types, the quantifier moves to the upper most level.

We write $v : T$ to declare that v has the type T.

Example 3.1

Let $T ::= C\ a$ be a type application or a type variable.

—

We will later see that a may be substituted by the quantified type $\forall a.a$. This would lead to $T ::= C\ (\forall a.a)$, but as quantifiers always move to the upper most level, it results in $\forall a.T ::= C\ a$ instead.

Example 3.2

The symbol `string` is a valid type. It can be though of as a type, whose implementation is unknown. For real programming languages this is not allowed.

Example 3.3

$\mathit{Bool} ::= \mathit{True} \mid \mathit{False}$ is a valid type application.

$\forall a. List\ a ::= Empty \mid Cons\ a\ (List\ a)$ is a valid quantified type.

Example 3.4

The natural numbers and the integers can be defined as types using the peano axioms [Pea89]:

- 1 is a natural number.
- Every natural number has a successor.

These axioms can be used for the definition of the type application.

$$Nat ::= 1 \mid Succ\ Nat$$

For integers, we can use the property that they contain 0 as well as all positive and negative numbers.

$$Int ::= 0 \mid Pos\ Nat \mid Neg\ Nat$$

Definition 3.1: Sets of Types

We define

- $\mathcal{V}$ as the set of all values of type variables.
- $\mathcal{A}$ as the set of all values of type applications.
- $\mathcal{Q}$ as the set of all values of quantified types.
- $\mathcal{T} ::= \mathcal{V} \cup \mathcal{A} \cup \mathcal{Q}$ as the set of all values of all types.

Instead of writing "let a be a type application or a type variable" we can now just write $a \in \mathcal{M}$.

Definition 3.2: partial function

Let $T_1 \in \mathcal{T}; \quad T_2 \in \mathcal{T}$.

—

We define $f \subseteq T_1 \times T_2$ as a partial function from T_1 to T_2.

When ever we write $f \subseteq T_1 \times T_2$, we assume that f is univariant:

$$(x, y_1) \in f \wedge (x, y_2) \in f \Rightarrow y_1 = y_2$$

.

Definition 3.3: Sort, Value, Constructor

Let $n \in \mathbb{N}; \quad \forall j \in \mathbb{N}_1^n k_j \in \mathbb{N}; \quad i \in \mathbb{N}_1^n; \quad \forall j : \mathbb{N}_0^{k_i}.t_j : T_{i,j}; \quad T \in \mathcal{M}$
$T ::= C_1\ T_{1,1} \ldots T_{1,k_1} \mid \ldots \mid C_n\ T_{n,1} \ldots T_{n,k_n}$.

—

We call

- $C_i\ T_{i,1} \dots T_{i,k_1}$ a *sort* of T,
- $C_i\ t_0 \dots t_{k_i}$ a *value* of T,
- C_i a *terminal* of T,
- the function

$$\begin{aligned} C_i &: T_{i,1} \to \dots \to T_{i,k_1} \to T \\ C_i(t_1, \dots, t_n) &:= C_i\ t_1 \dots\ t_n \end{aligned}$$

  a *constructor* of T.

Definition 3.4: Bounded, Free, Set of free variables

Let $n \in \mathbb{N}; \quad a \in \mathcal{V}; \quad k \in \mathbb{N}_1^n \to \mathbb{N}_0; \quad \forall i \in \mathbb{N}_1^n \forall j \in \mathbb{N}_1^n . T_{i,k(j)} \in \mathcal{T};$
$A ::= C_1\ T_{1,1} \dots\ T_{1,k(1)} \dots C_n\ T_{n,1} \dots\ T_{n,k(n)} \in \mathcal{A}; \quad T \in \mathcal{T};$
$P = \forall a.T \in \mathcal{Q}$.

—

$a \in \mathcal{V}$ is called *bounded*. Unbounded type variables are called *free*.

The set of all free type variables of a type is defined as follows:

$$\begin{aligned} \text{free}(a) :=& a \\ \text{free}(A) :=& \bigcup_{i \in \mathbb{N}_0^n} \bigcup_{j \in \mathbb{N}_0^{k_i}} \text{free}(T_{i,j} : \mathcal{V}) \\ \text{free}(P) :=& \text{free}(T) \setminus \{a\} \end{aligned}$$

A type can be substituted by replacing a bounded type variable with a mono type:

Definition 3.5: Type substitution

Let $n \in \mathbb{N}; \quad \Theta \subseteq \mathcal{V} \times \mathcal{T}; \quad a \in \mathcal{V}; \quad k \in \mathbb{N}_1^n \to \mathbb{N}_0; \quad \forall i \in \mathbb{N}_1^n \forall j \in \mathbb{N}_1^n . T_{i,k(j)} \in \mathcal{T};$
$A ::= C_1\ T_{1,1} \dots\ T_{1,k(1)} \dots C_n\ T_{n,1} \dots\ T_{n,k(n)} \in \mathcal{A}; \quad T \in \mathcal{T}; \quad P = \forall b.T \in \mathcal{Q};$
$S \in \mathcal{T}$.

—

We define the substitute of a type $[.]_\Theta : \mathcal{T} \to \mathcal{T}$ as

$$
\begin{aligned}
[a]_\Theta &:= \begin{cases} S & \text{if } (a, S) \in \Theta \\ a & \text{else} \end{cases} \\
[A]_\Theta &:= C_1\ [T_{1,1}]_\Theta \dots [T_{1,k_1}]_\Theta \\
&\quad \mid \dots \\
&\quad \mid C_n\ [T_{n,1}]_\Theta \dots [T_{n,k_n}]_\Theta \\
[P]_\Theta &:= \begin{cases} [T]_\Theta & \text{if } \exists (b, \_) \in \Theta \\ \forall b.[T]_\Theta & \text{else.} \end{cases}
\end{aligned}
$$

Θ is called the set of substitutions.

The type substitution gives raise to a partial order $\sqsubseteq$:

Axiom 3.2: Type Order

Let $T_1 \in \mathcal{T}$; $T_2 \in \mathcal{T}$; $\forall i \in \mathbb{N}_0^n . a_i \in \mathcal{V}$; $\forall i \in \mathbb{N}_0^n . S_i \in \mathcal{T}$; $\Theta = \bigcup \{(a_i, S_i)\}$

—

We define the partial order $\sqsubseteq$ such that

$$\frac{T_2 = [T_1]_\Theta \quad \forall i \in \mathbb{N}_0^m . b_i \notin \text{free}(\forall a_1 \dots \forall a_n . T) \quad m \leq n}{\forall a_1 \dots \forall a_n . T_1 \sqsubseteq \forall b_1 \dots \forall b_m . T_2}$$

The rule can be read as follows:

- First replace all bounded variables with types.
- Next rebound any new variables (variables that were previously not free).

Axiom 3.3: Product Type

Let $n \in \mathbb{N}$; $\forall i \in \mathbb{N}_1^n . T_i : \mathbb{T}, l_i$ be a unique symbol.

—

We call $T = \{l_1 : T_1, .., l_n : T_n\}$ a *product type*. We say that $T \in \mathcal{A}$.

- We call $\forall i \in \mathbb{N}_1^n . l_i$ the *labels* of the product type.
- The values of a product type have the form $\{l_1 = t_1, \dots, l_n = t_n\}$ where $\forall_{i \in \mathbb{N}_1^n} . t_i : T_i$.
- The types T_i are unordered: $\{a : T_1, b : T_2\} = \{b : T_2, a : T_1\}$.

For ordered product types we write

$T_1 \times \dots \times T_n := \{1 : T_1, \dots, n : T_n\}$.

Values of a ordered product type have the form $(t_1, \dots, t_n)$, where $\forall_{i \in \mathbb{N}_1^n} . t_i : T_i$.

We most general example of a product type is a record. Tuples can be represented as ordered product types.

Axiom 3.4: Functions

Let $T_1 \in \mathcal{T}; \quad T_2 \in \mathcal{T}$.

—

Functions from one type T_1 to another T_2 are mono types. We use the notation $T_1 \to T_2$ to describe function types.

Example 3.5

Let $T_1 \in \mathcal{T}; \quad T_2 \in \mathcal{T}; \quad T_3 \in \mathcal{T}$.

—

Then $(T_1 \times T_2 \to T_3)$ is isomorphic to $T_1 \to (T_2 \to T_3)$. This was originally proven by Gottlob Frege [Sch24]. This method for translating multivariable functions into single variable functions is nowadays called *currying* and named after Haskell Curry who further developed the theory [CF58].

References

[CF58] H.B. Curry and R. Feys. *Combinatory Logic*. Combinatory Logic v. 1. North-Holland Publishing Company, 1958. URL: https://books.google.at/books?id=fEnuAAAAMAAJ.

[DM82] Luís Damas and Robin Milner. "Principal Type-Schemes for Functional Programs". In: *Conference Record of the Ninth Annual ACM Symposium on Principles of Programming Languages, Albuquerque, New Mexico, USA, January 1982*. 1982, pp. 207–212. DOI: 10.1145/582153.582176. URL: https://doi.org/10.1145/582153.582176.

[Pea89] G. Peano. *Arithmetices principia: nova methodo*. Trans. by Vincent Verheyen. Fratres Bocca, 1889. URL: https://github.com/mdnahas/Peano_Book/blob/master/Peano.pdf.

[Sch24] M. Schönfinkel. "Über die Bausteine der mathematischen Logik". In: *Mathematische Annalen* 92.3 (Sept. 1924), pp. 305–316. ISSN: 1432-1807. DOI: 10.1007/BF01448013. URL: https://doi.org/10.1007/BF01448013.