

A Theory of Type Polymorphism in Programming: The Cheat Sheet

Pete Bevin

April 2015

1 Overview of the Paper

1. Introduction

2. Illustrations of the Type Discipline

3. **A Simple Applicative Language and its Types**

- The `exp` language, its semantics, and what type errors it can raise at runtime (3.1, 3.2)
- Types in general (3.3, 3.4)
- What it means to be well-typed (3.5)
- Substitutions (3.6)
- The Semantic Soundness theorem (if we can assign a type, the runtime won't raise errors) (3.7)

4. **A Well-Typing Algorithm and its Correctness**

- Algorithm W
- The Syntactic Soundness theorem
- Algorithm J

5. Types in Extended Languages

- Tuples, union types, and lists
- Assignable variables and assignments
- Recursive type declarations

2 Symbols

Term	Definition in Paper	Meaning
B_0	$T = \{\text{true}, \text{false}, \perp_T\}$	Boolean types (including \perp)
B_1, \dots, B_n	Other types	Integers, reals, strings, pairs of types, etc.
V		
W	Wrong	The error type
η	Environment	In the paper, this is a function from <code>id</code> to V , but we would be more likely to model it as a <code>map</code> .
\mathcal{E}	Semantic Function	<code>eval</code>
$\mathcal{E}[[T]]\eta$	Semantic evaluation	<code>(eval exp env)</code> where <code>eval</code> is \mathcal{E} , <code>exp</code> is T , and <code>env</code> is η
$\eta[[x]]$	Type lookup	The type of the <code>Exp</code> variable x .
ι_n	Type of B_n	For example, B_0 is \mathbb{B} , B_1 might be \mathbb{N} , B_2 might be \mathbb{R} , etc.
$v\mathbf{ED}$	[16] page 34	Value v has type D - like <code>instanceof</code> in Java.
$v \mid D$		Type cast (error value if not possible, but we always check first with $v\mathbf{ED}$)
$p \mid e$	Prefixed expression	An expression showing what <code>λ</code> , <code>fix</code> , and <code>let</code> bindings are in effect
$\bar{p} \mid \bar{e}$	Typed expression	A prefixed expression augmented with type information at each level
\sqsubseteq	No definition given	Measure of “defined”ness – e.g., in B_0 , $\perp_T \sqsubseteq \text{true}$ and $\perp_T \sqsubseteq \text{false}$, but $\text{false} \not\sqsubseteq \text{true}$.
\sqsubset	No definition given	

3 Concepts

Page	Concept	Meaning
361	Prefix	A list of the variable bindings in effect for an expression
361	Active	A member of the prefix that is not shadowed by a later member with the same name
362	Generic type variable	One that does not occur in the type of any <code>λ</code> or <code>fix</code> binding above it
362	Standard typing	You can safely ignore this: it’s a purely technical constraint.
362	Well-typed	An expression is well-typed if it can be assigned a type and the resulting type obeys certain laws. See Proposition 3 on page 362.
364	Semantic Soundness Theorem	A well-typed program cannot “go wrong”
367	Syntactic Soundness Theorem	If algorithm \mathcal{W} accepts a program, then it is well-typed.