References

Advanced Compiler Construction and Program Analysis

Lecture 6

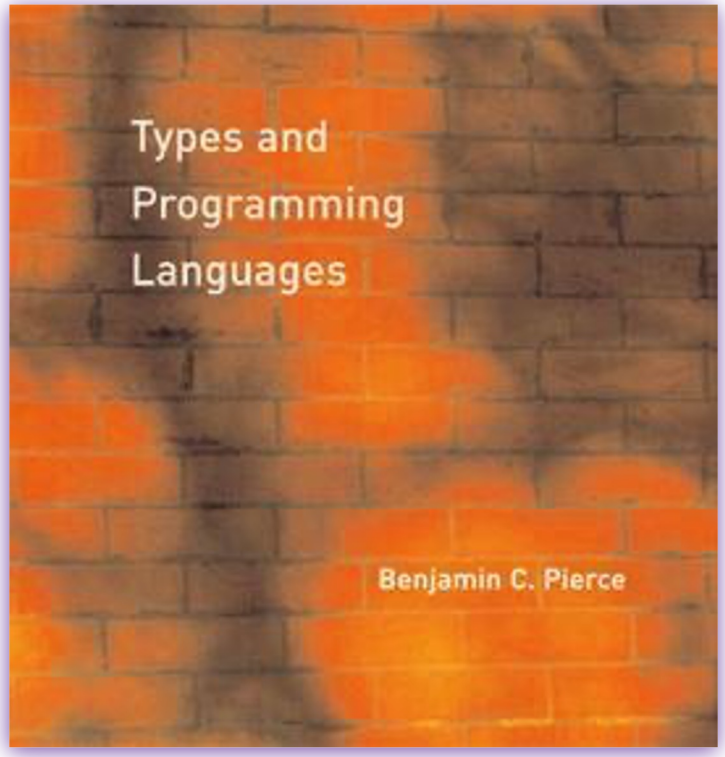
Innopolis University, Spring 2022

The topics of this lecture are covered in detail in…

Benjamin C. Pierce.

Types and Programming Languages

MIT Press 2002



References: syntax

```
t ::= …                    terms
    ref t                  new reference
    !t                     dereference
    t := t                 assignment
    l                      store location

T ::= …                    types
      Ref T                reference type
```

References: basics

Assuming, we have top-level named declarations:

`r = ref 5`	*declares*	`r : Ref Nat`
`!r`	*evaluates to*	`5 : Nat`
`r := 7`	*evaluates to*	`unit : Unit`
`!r`	*evaluates to*	`7 : Nat`

Side effects and sequencing

We will rely on sequencing operator, introduced in Lecture 2.

```
(r := succ (!r); !r)
```

⟶ `8`

Side effects and sequencing

We will rely on sequencing operator, introduced in Lecture 2.

```
(r := succ (!r); !r)
```

⟶ `8`

```
(λ_:Unit. !r) (r := succ (!r))
```

⟶ `8`

Aliasing: exercise

Exercise 6.1. Draw diagrams, explaining evaluation of the following terms:

```
{ref 0, ref 0}
```

```
(λx:Ref Nat. {x,x}) (ref 0)
```

Shared state: exercise

Exercise 6.2. Are the following programs equivalent?

```
(r := 1; r := !s)
```

```
r := !s
```

Shared state: counter

```
c = ref 0
inc = λx:Unit. (c := succ (!c); !c)
dec = λx:Unit. (c := pred (!c); !c)

inc unit          increments counter and returns 1
inc unit          increments counter and returns 2
dec unit          decrements counter and
```

References to compound types

```
Nats = Ref (Nat → Nat)

init = λ_:Unit. ref (λn:Nat.0)
lookup = λa:Nats. λn:Nat. (!a) n;
```

Exercise 6.3. Implement `update`.

Explicit de-allocation

Exercise 6.4. Allowing explicit **`free`** operator can break type safety, as the same location can be used for different variables of types, for example, `Ref Nat` and `Ref Bool`. Demonstrate on a specific example, how exactly this can break type safety.

References: typing rules

$$\frac{\Gamma \vdash t : T}{\Gamma \vdash \texttt{ref}\ t : \texttt{Ref}\ T}$$

$$\frac{\Gamma \vdash t : \texttt{Ref}\ T}{\Gamma \vdash \texttt{!}t : T}$$

$$\frac{\Gamma \vdash t_1 : \texttt{Ref}\ T \qquad \Gamma \vdash t_2 : T}{\Gamma \vdash t_1 := t_2 : \texttt{Unit}}$$

References: store

To understand, how to evaluate !x, we need to understand, where to take the value of a reference from. For that purpose, we extend operational semantics, to include a **store**.

$$\frac{t_1 \mid \mu \longrightarrow u_1 \mid \mu'}{t_1\ t_2 \mid \mu \longrightarrow u_1\ t_2 \mid \mu'}$$

$$\frac{t_2 \mid \mu \longrightarrow u_2 \mid \mu'}{t_1\ t_2 \mid \mu \longrightarrow t_1\ u_2 \mid \mu'}$$

$$(\lambda x.t_1)\ t_2 \mid \mu \longrightarrow [x \mapsto t_2]t_1 \mid \mu$$

References: evaluation (dereference)

$$\frac{t \mid \mu \longrightarrow u \mid \mu'}{!t \mid \mu \longrightarrow !u \mid \mu'}$$

$$\frac{\mu(l) = v}{!l \mid \mu \longrightarrow v \mid \mu}$$

References: evaluation (assignment)

$$\frac{t_1 \mid \mu \longrightarrow u_1 \mid \mu'}{t_1 := t_2 \mid \mu \longrightarrow u_1 := t_2 \mid \mu'}$$

$$\frac{t_2 \mid \mu \longrightarrow u_2 \mid \mu'}{l := t_2 \mid \mu \longrightarrow l := t_2 \mid \mu'}$$

$$l := v \mid \mu \longrightarrow \mathtt{unit} \mid \mu[l \mapsto v]$$

References: evaluation (new reference)

$$\frac{t \mid \mu \longrightarrow u \mid \mu'}{\texttt{ref}\ t \mid \mu \longrightarrow \texttt{ref}\ u \mid \mu'}$$

$$\frac{l \text{ is free in } \mu}{\texttt{ref}\ v \mid \mu \longrightarrow l \mid \mu[l \mapsto v]}$$

References: store typings

$\Sigma ::=$	***store typings***
$\quad \emptyset$	*empty store*
$\quad \Sigma, l:T$	*location typing*

$$\frac{\Sigma(l) : T}{\Gamma \mid \Sigma \vdash l : \mathsf{Ref}\ T}$$

References: type safety (preservation)

Exercise 6.5. Explain why the following formulation of preservation is wrong:

$$\text{If } \Gamma \mid \Sigma \vdash t : T \text{ and } t \mid \mu \longrightarrow t' \mid \mu', \text{ then } \Gamma \mid \Sigma \vdash t' : T$$

References: type safety (preservation)

Theorem 6.6.

If

1. $\Gamma \mid \Sigma \vdash t : T$

2. $\Gamma \mid \Sigma \vdash \mu$

3. $t \mid \mu \longrightarrow t' \mid \mu'$

then, for some $\Sigma' \supseteq \Sigma$,

$\Gamma \mid \Sigma' \vdash t' : T$ and $\Gamma \mid \Sigma' \vdash \mu'$

References: type safety (progress)

Theorem 6.6.
Suppose t is a closed, well-typed term
(that is, $\emptyset \mid \Sigma \vdash t : T$ for some T and Σ).
Then either t is a value or else,
for any store μ such that $\emptyset \mid \Sigma \vdash \mu$,
there is some term t' and store μ'

with $t \mid \mu \longrightarrow t' \mid \mu'$.

Summary

- ❑ References

See you next time!