Lesson 12
Existential Types

2/29
Chapter 24

Existential Types

• Existential packages
• Existential types and abstract types
• Existential types and objects
• Encoding existentials with universals
Existential types:

\[ T ::= \ldots \mid \Box X.T \]

Pierce uses the alternative, nonstandard notation \{\Box X, T\} to suggest that the existential value is a mixed type-value tuple.

A value of type \( \Box X.T \) is a package with a witness type \( T' \) for \( X \) and a value term \( t : [X \Rightarrow T']T \).

- \( \text{pack } X = T' \text{ with } t \text{ as } T : [X \Rightarrow T']T \) (conventional notation)
- \( \{T', t\} \text{ as } \{\Box X, T\} \) (Pierce’s notation)

The \textit{Intro} typing rule for existential types:

\[
\frac{}{\Gamma \vdash t_1 : [X \Rightarrow U]T_1} \quad (T\text{-Pack})
\]

\[
\frac{}{\Gamma \vdash \{U, t_1\} \text{ as } \{\Box X, T_1\} : \{\Box X, T_1\}}
\]
Examples of Existential types

\(*\text{Nat}, 3\) as \(\{\exists X, X\} : \{\exists X, X\}\)
\(*\text{Bool}, \text{true}\) as \(\{\exists X, X\} : \{\exists X, X\}\)

\(p = (*\text{Nat}, \{a = 5, f = \lambda x : \text{Nat}. \text{succ } x\})\) as \(\{\exists X, \{a : X, f : X \rightarrow \text{Nat}\}\}\)
\(p : \{\exists X, \{a : X, f : X \rightarrow \text{Nat}\}\}\)

\(q = (*\text{Bool}, \{a = \text{true}, f = \lambda x : \text{Bool}. 0\})\) as \(\{\exists X, \{a : X, f : X \rightarrow \text{Nat}\}\}\)
\(q : \{\exists X, \{a : X, f : X \rightarrow \text{Nat}\}\}\)

The type part is hidden (opaque, abstract), and the value part provides an interface for interpreting the hidden type.

Unpacking existential values

Unpacking an existential: \(\text{let } \{X,x\} = t_1 \text{ in } t_2\)

\[\quad \frac{\emptyset \vdash t_1 : \{\exists X, T_1\} \quad \emptyset, X, x : T_1 \vdash t_2 : T_2}{\emptyset \vdash \text{let } \{X,x\} = t_1 \text{ in } t_2 : T_2} \quad \text{ (T- Unpack)}\]

Type variable \(X\) cannot occur in \(T_2\) -- it is not in scope (i.e. doesn't appear in the context \(\emptyset\)). This means that the name \(X\) of the existential witness type cannot "escape" from the let expression.

Also, within the body \(t_2\), the type \(X\) is abstract and can only be used through the interface provided by \(x : T_1\).
Abstract types as Existential types

```
Counter = {[Counter, {new: Counter, get: Counter -> Nat, inc: Counter -> Counter}]

counter0 = {*Nat,
new = 1,
get = \x : Nat. x,
inc = \x : Nat. succ x} as Counter

let {C, cops} = counter0 in
let add2 = \c : C. cops.inc(cops.inc c)
in cops.get(add2(cops.new))
==> 4: Nat
```

Abstract types as Existential types

The idea:
```
let {X,x} = t₁ in t₂
```

The existential value packages the representation type of the abstract type (the existential witness), with the interface values through which to use the type.

The client code (t₂, the body of the let), does not have access to the representation type, and can only refer to it through the bound (and opaque) name X.

All code that interacts via the abstract type has to be included in t₂. This limitation can be overcome using the dot notation.
Object types as Existential types

Counter = {[S, {state: S, methods: {get: S -> Nat, inc: S -> S}}]}

\[c = \{\text{Nat}, \{\text{state} = 1, \text{methods} = \{\text{get} = \lambda x : \text{Nat}. x, \text{inc} = \lambda x : \text{Nat}. \text{succ} x\}\}\} \text{ as Counter}\]

c : Counter

sendget = \lambda x : \text{Counter}. let \{S, body\} = x in body.methods.get(body.state)

sendget c ==> 1: Nat

Object types as Existential types

The idea:

The witness type of the existential value is the representation of the internal state of the object, and it is hidden by the existential package.

The methods operate on this internal state.
Lesson 12: Existential Types

Encoding Existentials with Universals

Let
\[ \{ \Box X, T \} \overset{\text{def}}{=} Y. (\Box X. T \to Y) \to Y \]

Then
\[ \{^* S, t \} \overset{\text{def}}{=} Y. \Box f : (\Box X. T \to Y). f [S] (t) \]

and
\[ \text{let } \{ X, x \} = t_1 \text{ in } t_2 \overset{\text{def}}{=} t_1 [T_2] (\Box X. \Box f : T_1. t_2) \]

where \( t_1 : \{ \Box X \} \) and \( t_2 : T_2 \).