UCLA COMPUTER SCIENCE DEPARTMENT

Statically Scoped Object Adaptation with Expanders

Alessandro Warth
Milan Stanojevic
Todd Millstein

WHAT ARE EXPANDERS?

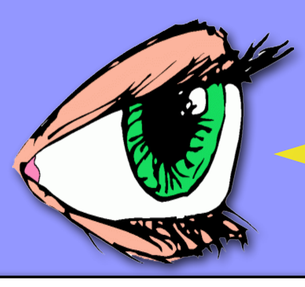
Expanders allow existing classes (more generally, whole class hierarchies) to be noninvasively updated with new methods, fields, and superinterfaces. Each client can customize its view of a class by explicitly importing any number of associated expanders. This view then applies to all instances of that class, including objects passed to the client from other components.

BENEFITS OF EXPANDERS

- Language support for the *object adapter* and *visitor* design patterns;
- *Modular reasoning* (a.k.a., the "no surprises" guarantee): an expander has no effect on the behavior of compilation units that do not explicitly `use` that expander;
- *Modular compilation*: a programmer may write an expander for *any* class (or class hierarchy), even without having access to its source code.

EXPANDERS IN ACTION

In the example below, `Client2` uses the `tcExpander` to augment its view of the `Expr` hierarchy with a new `type` field and a new method `tc`. This additional functionality enables `Client2` to typecheck all instances of `Expr`, including those created by compilation units that are unaware of `tcExpander`, like `Client1`.

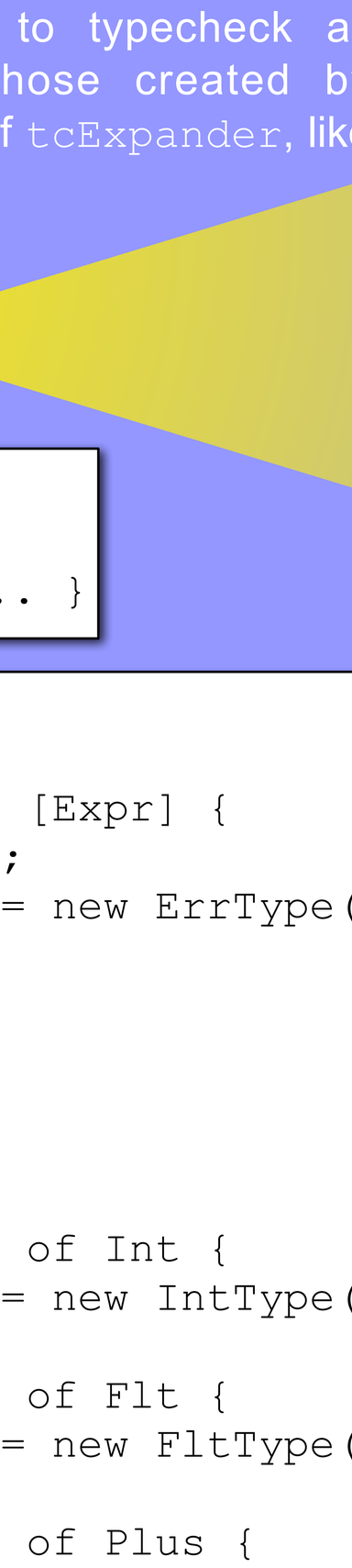


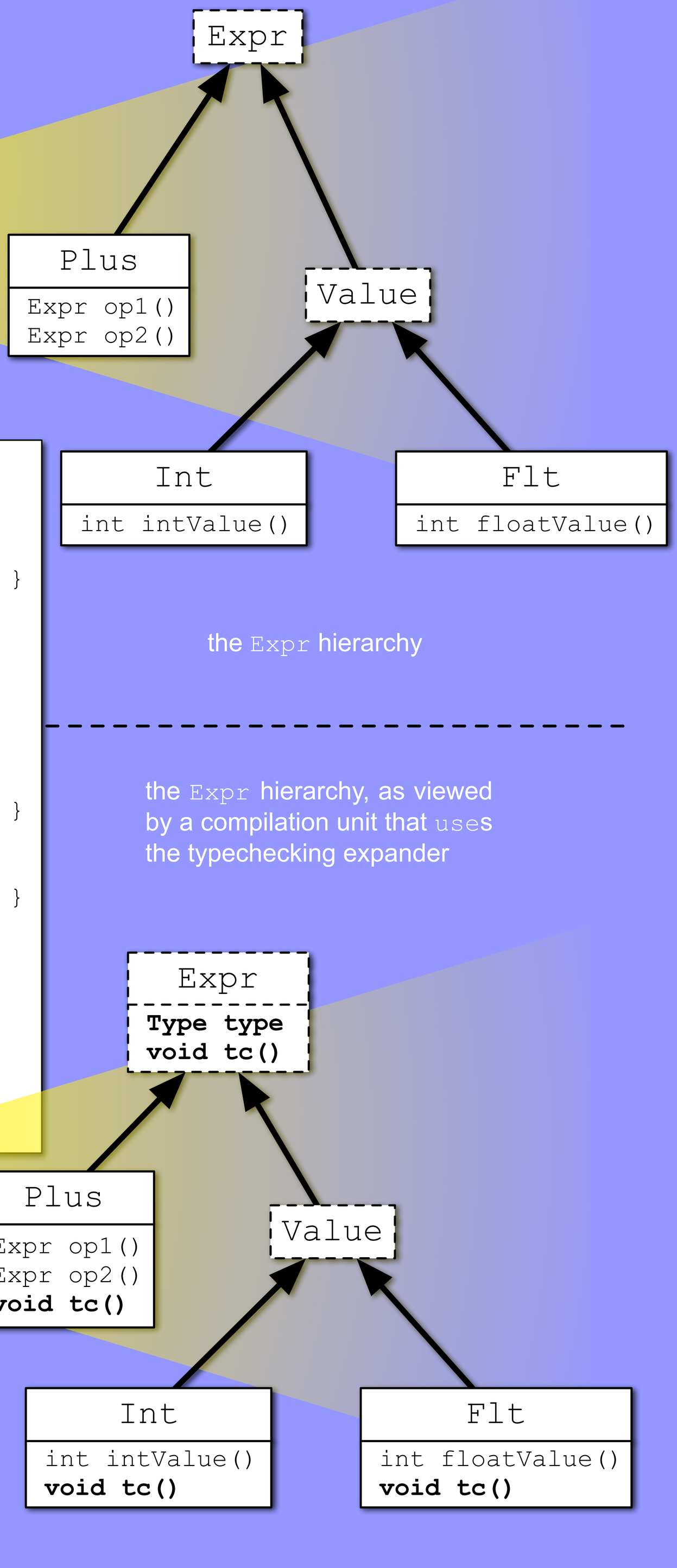
```
// Client #1
import expr.*;
class Client1 { ... }
```

```
package exprUtil;
import expr.*;
public expander tcExpander [Expr] {
  private Type type = null;
  public void tc() { type = new ErrType(); }
  public Type type() {
    if (type == null)
      tc();
    return type;
  }
}
public expander tcExpander of Int {
  public void tc() { type = new IntType(); }
}
public expander tcExpander of Flt {
  public void tc() { type = new FltType(); }
}
public expander tcExpander of Plus {
  public void tc() {
    Type t1 = op1().type();
    Type t2 = op2().type();
    // ...
  }
}
```



```
// Client #2
import expr.*;
use exprUtil.tcExpander;
class Client2 { ... }
```





the `Expr` hierarchy

the `Expr` hierarchy, as viewed by a compilation unit that `use`s the typechecking expander

For more information please visit
http://www.cs.ucla.edu/~awarth/expanders/