

# The Expression Problem, Scandinavian Style

Erik Ernst

Department of Computer Science

University of Aarhus

Denmark

MASPEGHI @ ECOOP 2004



# Overview

- The expression problem
- Basic class family
- Adding a new operation
- Adding a new data structure
- Combining the extensions
- Conclusion



# Overview

- The expression problem
- Basic class family
- Adding a new operation
- Adding a new data structure
- Combining the extensions
- Conclusion



# The Expression Problem

- Imagine a library supporting a data-structure  $T$  in some variants  $T_1 \dots T_n$ , each supporting the operations  $f_1 \dots f_k$

| $T \setminus f$ | $f_1$ | $f_2$ | $\dots$ | $f_k$ |
|-----------------|-------|-------|---------|-------|
| $T_1$           | *     | *     | $\dots$ | *     |
| $\dots$         |       |       |         |       |
| $T_n$           | *     | *     | $\dots$ | *     |

- OO: Easy to add new row. Functional: Easy to add new column.
- Problem: Hard to make both easy!
- Rest of presentation: Example of how to do it in gbeta



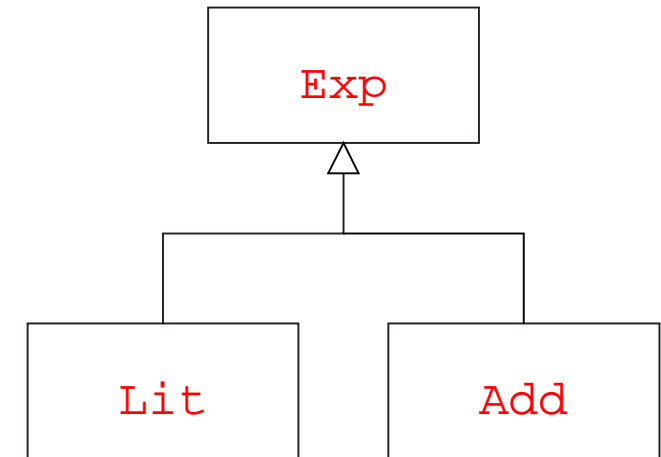
# Overview

- The expression problem
- **Basic class family**
- Adding a new operation
- Adding a new data structure
- Combining the extensions
- Conclusion



## Basic Class Family: The language 'Lang'

```
class Lang {  
    virtual class Exp {  
        String toString() {}  
    }  
    virtual class Lit extends Exp {  
        int value;  
        Lit(int value) { this.value=value; }  
        String toString() { return value; }  
    }  
    virtual class Add extends Exp {  
        Exp left,right;  
        Add(Exp left, Exp right) { this.left=left; this.right=right; }  
        String toString() { return left + "+" + right; }  
    }  
}
```





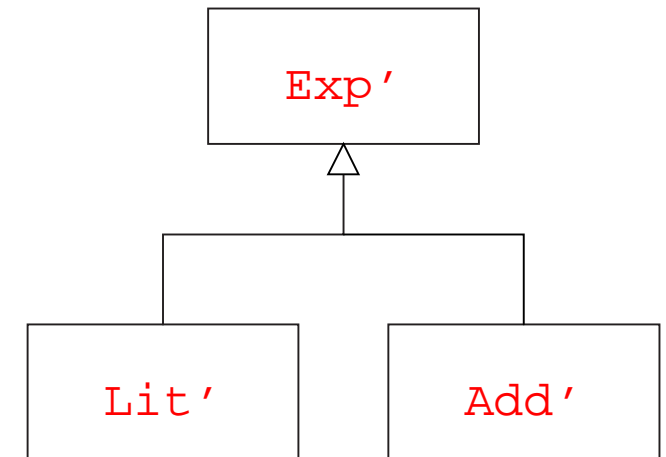
# Overview

- The expression problem
- Basic class family
- Adding a new operation
- Adding a new data structure
- Combining the extensions
- Conclusion



## Adding a new operation 'eval'

```
class LangEval extends Lang {  
  refine class Exp {  
    int eval() {}  
  }  
  refine class Lit {  
    int eval { return value; }  
  }  
  refine class Add {  
    int eval { return left.eval()+right.eval(); }  
  }  
}
```





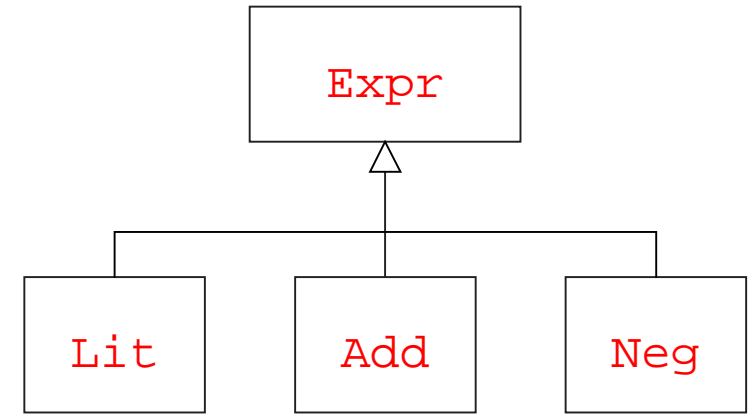
# Overview

- The expression problem
- Basic class family
- Adding a new operation
- Adding a new data structure
- Combining the extensions
- Conclusion



## Adding a new data structure 'Neg'

```
class LangNeg extends Lang {  
    virtual class Neg extends Exp {  
        Neg(Exp exp) { this.exp=exp; }  
        String toString() { return "-(" + exp + ")"; }  
        Exp exp;  
    }  
}
```



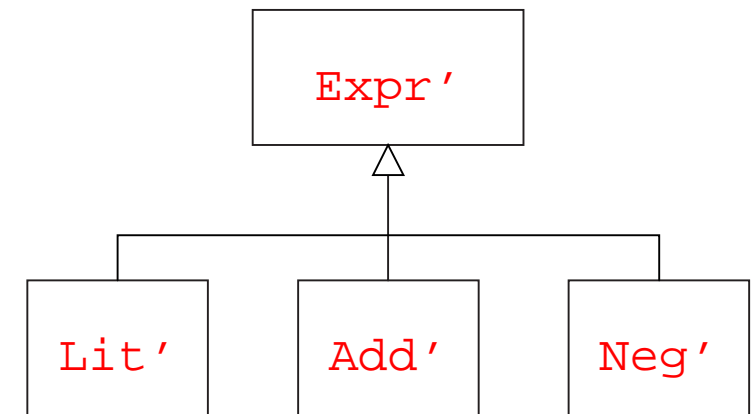


# Overview

- The expression problem
- Basic class family
- Adding a new operation
- Adding a new data structure
- Combining the extensions
- Conclusion



## Combining both extensions



```
class LangNegEval extends LangEval ⊕ LangNeg {  
  refine class Neg {  
    int eval() { return -exp.eval() }  
  }  
}
```



## Issues when combining extensions

- Problem: combining implies extending both, with each other!

| $T \setminus f$ | $f_1$ | $\dots$ | $f_k$ | $f_{k+1}$ |
|-----------------|-------|---------|-------|-----------|
| $T_1$           | *     | $\dots$ | *     | •         |
| $\dots$         |       |         |       |           |
| $T_n$           | *     | $\dots$ | *     | •         |
| $T_{n+1}$       | •     | $\dots$ | •     | ⊙         |

- The trick is that we just specify what we need, then get it



# Overview

- The expression problem
- Basic class family
- Adding a new operation
- Adding a new data structure
- Combining the extensions
- Conclusion



## Conclusion

- Higher-order hierarchies in gbeta support a straightforward solution to the expression problem
- Basic mechanism behind solution: recursive propagation and constraint-oriented semantics of virtual classes
- Not only may we add new operations as well as new data structures, independent extensions may also be combined