Classboxes:
An Experiment in Modeling Compositional Abstractions using Explicit Contexts

Markus Lumpe
Department of Computer Science
Iowa State University

Jean-Guy Schneider
Faculty of Information & Communication Technologies
Swinburne University of Technology
Do we really need a specially designed composition language?

Systematic method for construction large software systems
What are the obstacles?

- Dependence on position and arity
- Changes may affect the whole system
- The world is dynamic
And the winner is ...

Forms
What are forms?

- First-class namespaces with a small set of purely asymmetric operators
- Component interfaces, components, and composition mechanisms
- Compile-time and run-time entities
What else can we say about forms?

• Forms are not bound to a particular computational model.

• Forms have to be combined with a concrete target system.
The $\lambda F$-Calculus

$F, G, H ::= \langle \rangle$

| empty form |

| $\mid X$ | form variable |

| $\mid F\langle l = V \rangle$ | binding extension |

| $\mid F \oplus G$ | form extension |

| $\mid F \setminus G$ | form restriction |

| $\mid F \Rightarrow l$ | form dereference |

| $\mid F[G]$ | form context |

| $\mid M. l$ | projection |

| $\mid \lambda (X) M$ | abstraction |

| $\mid M N$ | application |

| $\mid M[F]$ | $\lambda F$-context |

$V ::= \varepsilon$

empty value

$a$ abstract value

$F, G, H ::= \langle \rangle$

empty form
How can we represent objects in the $\lambda F$-calculus?

\[
C_\alpha = \begin{array}{l}
\text{let} \\
\Delta_C = \lambda (\text{State}) (\text{[Methods}_C]) \text{[State]} \\
G_C = \lambda (\gamma) \lambda (I) P_\beta \oplus \Delta_C \langle I \oplus (\text{State}_C) \rangle \\
W_C = \lambda (\gamma) \mu_{\text{self}} \langle (\gamma \rightarrow C_\alpha).G (\beta \oplus \gamma) \rangle \text{[self]} \\
in \\
\langle G = G_C, W = W_C \rangle
\end{array}
\]
How can we denote changes?

Class extensions are used to add or refine features of existing classes in a namespace.
How can we manage changes?

Class extensions are only visible in the namespace in which they are defined.
A new module system: *Classboxes*

- Classboxes define explicitly named scopes.
- Classboxes support import and local refinement of classes.
What are the available operations?

- Import a classes
- Introduction of subclasses
- Extension of classes
- Inclusion of new behavior
A Point Class Hierarchy

OriginalCB

Point

BoundedPoint

LinearBPoint

LinearCB

Point

BoundedPoint

LinearBPoint

ColorCB

Point

BoundedPoint

LinearBPoint

+Color

TraceCB

Point

BoundedPoint

LinearBPoint

+Trace
Extension

\[
\mathcal{B}^{E_{\beta}} = \\
\text{let} \\
\Delta_B = \lambda(\text{State}) (\text{Methods}_B) [\text{State}] \\
G_B = \lambda(\text{Class}) \\
\lambda(\gamma) \lambda(I) \\
\text{let} \\
P = \mu_{\text{self}} (\text{Class}.G \gamma) [\text{self}] \ I \\
in \\
P \oplus \Delta_B I \oplus (\text{State}_B \langle \text{super} = P \rangle) \\
in \\
\langle G = G_B \rangle
\]
Inclusion

\[ B^I_\beta = \]

let

\[ \Delta_B = \lambda (\text{State}) \left( \textbf{Methods}_B \right) \left[ \text{State} \right] \]

\[ G_B = \lambda (\text{Class}) \]

\[ \lambda (\gamma) \lambda (I) \]

let

\[ P = (\text{Class}.G \gamma) I \]

in

\[ P \oplus \Delta_B \langle I \oplus (\textbf{State}_B <\text{original} = P>) \rangle \]

\[ \langle G = G_B \rangle \]
How can we apply extension and inclusion?

Extension:

\[ C_\alpha = (\text{lookupClass} \langle C, \alpha' \rangle) \langle G = \text{BE}_\beta.G (\text{lookupClass} \langle C, \alpha' \rangle) \rangle \]

Inclusion:

\[ C_\alpha = \]

\[
\begin{align*}
\text{let} & \\
G_C &= \lambda(g) (\text{BI}_\beta.G (\text{lookupClass} \langle C, \alpha' \rangle)) \gamma \\
W_C &= \lambda(g) (\text{lookupClass} \langle C, \alpha' \rangle).W (\beta \oplus g) \\
\text{in} & \\
\langle G = G_C, W = W_C \rangle
\end{align*}
\]
Why is the encoding of classboxes in the $\lambda F$-calculus useful?

- Expressiveness of the $\lambda F$-calculus
- Precise semantics of classboxes
- Discovery of new operations
Can our results be applied to an industry-strength language?

- Yes, C# with explicit class extensions:

```csharp
namespace ColorCB
{
    using System.Drawing;
    using Point = OriginalCS.Point append Color;
    using LinearBoundedPoint = LinearCB.LinearBoundedPoint;

    extension Color {
        private Color color;

        public Color Color { get { return color; } 
                                  set { color = value; } }
    }
}
```