True separate compilation of Java classes

Davide Ancona, Giovanni Lagorio, Elena Zucca

DISI - University of Genova, Italy

APPSEM-II Workshop - Nottingham

This talk is based on “True Separate Compilation of Java Classes” [in PPDP02] and “Stronger Typings for Separate Compilation of Java-like Languages” [submitted].
Plan of the talk

• What is “true” separate compilation

• Why existing compilers/formal type systems for Java do not support/model true separate compilation

• Our contribution - A type system for a Java subset supporting true separate compilation
Compilation of a self-contained program

$S$ source program (closed), $B$ binary program (closed)

$\vdash S \leadsto B \quad$ type-checking $+$ generation of binary code
"True" separate compilation (Cardelli POPL’97)

intra-checking \( \Gamma \vdash S : \tau \rightsquigarrow B \)

\( S \) source fragment, \( \tau \) inferred type, \( B \) binary fragment,
\( \Gamma \) type environment = information on missing fragments

linkset = \( f_1 \mapsto \Gamma_1 \vdash S_1 : \tau_1 \rightsquigarrow B_1, \ldots, f_n \mapsto \Gamma_n \vdash S_n : \tau_n \rightsquigarrow B_n \)

inter-checking = \( \tau_i \) conforms to assumptions on \( f_i \) in \( \Gamma_j \), \( \forall i, j \)

e.g., if \( \Gamma_j = \ldots, f_i : \tau, \ldots \), then \( \tau = \tau_i \)
Motivations

Assume we modify some fragments

• if interchecking still holds we don’t need to recompile other fragments;

• if it doesn’t we get information on what needs to be recompiled.

Applications: smart selective recompilation
Do Java existing compilers support true separate compilation?

No: when we compile class C depending on $C_1, \ldots, C_n$

- $C_1, \ldots, C_n$ must be available at least in binary form (no separate interface files)

- type-checking (compilation) is propagated to some of $C_1, \ldots, C_n$
  compilation = intra-checking and some inter-checking interleaved $\Rightarrow$ standard compilers are not safe
Do Java existing formal type systems model true separate compilation?

No: existing formal definitions of Java type system

- extract a standard type environment $\Gamma$ from a program, roughly, associating to each class its parent and method signatures

- check consistency of $\Gamma$

- check each class body against $\Gamma$

So:
- inter-checking trivial

- intra-checking not abstract as it could be: each fragment is intra-checked against an overspecified type environment
True separate compilation for Java

Which is the “minimal” type information needed for intra-checking a Java class?

class C extends Parent {
    T id(T x) { return x; }
    T1 m1(T2 x) { return x ; }
    T1 m2(T2 x) { return new Used().g(x); }
}

Five kinds of judgments expressing “local” type requirements:

1) $\Gamma \vdash \exists T$

2) $\Gamma \vdash T_2 \leq T_1$
**True separate compilation for Java**

class C extends Parent {
    ...
    T1 m2(T2 x) { return new Used().g(x);}
}

Class *Used* must declare/inherit a method $\alpha \ g(\beta)$ with

$\Gamma \vdash T2 \leq \beta$

$\Gamma \vdash \alpha \leq T1$

But $\alpha, \beta$ must be known at compile-time since in bytecode method invocations are annotated with method descriptors
For instance, class C can be typechecked in the following environment (1):

```java
...
class T1{}
class T2 extends T1{}
class T3 extends T2{}
class Used { T3 g(T1 x) {...} }
----> new Used()[Used,T1,T3].g(x)
```

and also in this environment (2):

```java
...
class T1{}
class T2 extends T1{}
class Used {
    T2 g(T2 x) {...}
    int f() {...} }
----> new Used()[Used,T2,T2].g(x)
```
True separate compilation for Java

Judgment

3) \( \Gamma \vdash C.m(\bar{T}) \xrightarrow{res} \langle \bar{T}', T_{ret} \rangle \)

In the example:

\( \Gamma_1 \vdash Used.g(T2) \xrightarrow{res} \langle T1, T3 \rangle \)

\( \Gamma_2 \vdash Used.g(T2) \xrightarrow{res} \langle T2, T2 \rangle \)
True separate compilation for Java

class C extends Parent {
    T id(T x) { return x; }
    T1 m1(T2 x) { return x; } 
    T1 m2(T2 x) { return new Used().g(x); }
}

Apparently no requirements on Parent...yet: a “wrong” Parent

class Parent extends C {
    Parent m2(T2 x) {...}
}

Judgments:
4) \( \Gamma \vdash \text{Parent} \not\preceq \text{C} \)

5) \( \Gamma \vdash \text{Parent} \circ T1 \, m2(T2) \)
What we have done

- Formal definition of true separate compilation (intra-checking)
  \[ \Gamma \vdash S : \tau \rightsquigarrow B \]
  for a small (but significant) Java subset (source + bytecode)

- Formal definition of inter-checking

- Sound relation with standard compilation with standard environment
Further work

- Work in progress: Implementation of separate compilation for Java on top of a standard Java compiler
  Idea: if $\Gamma \vdash S : \tau \leadsto B$, then we can construct a collection of classes satisfying $\Gamma$ and give them to a standard compiler

- Work in progress: “smart” compilation manager for Java (the full language)