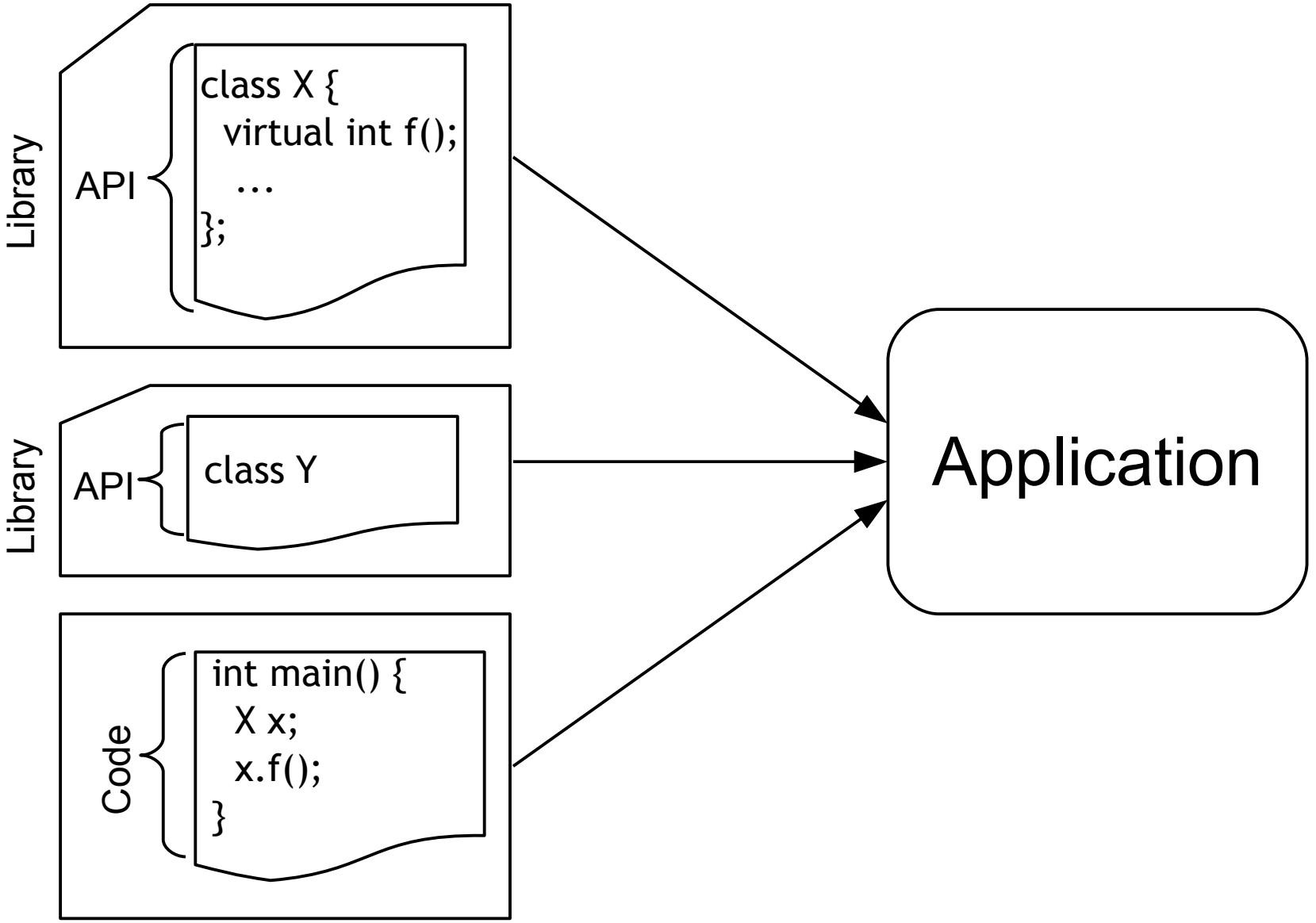
ABI Compatibility Through A Customizable Language

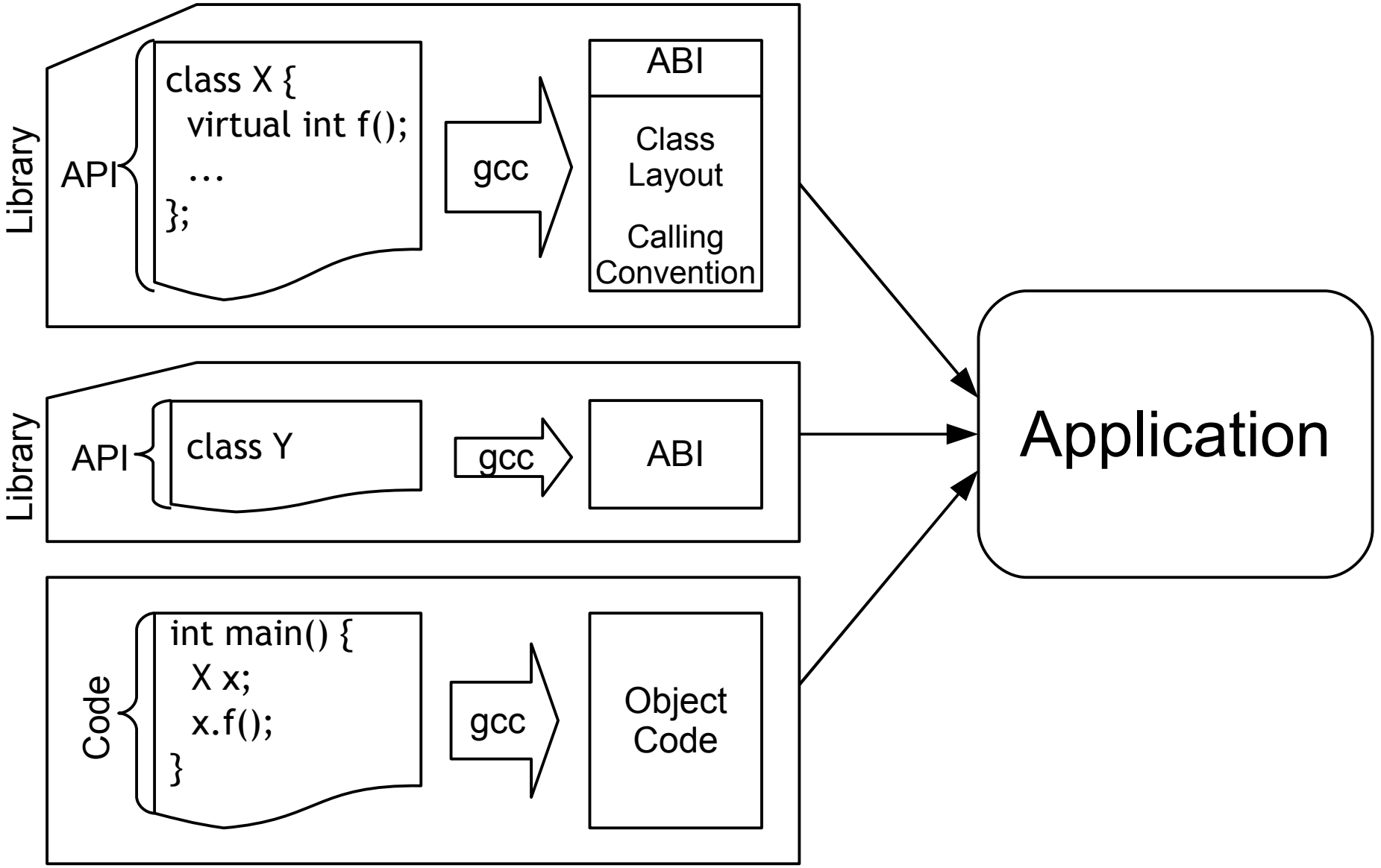
Kevin Atkinson,
Matthew Flatt, Gary Lindstrom

University of Utah

Application Programmer Interface (API)

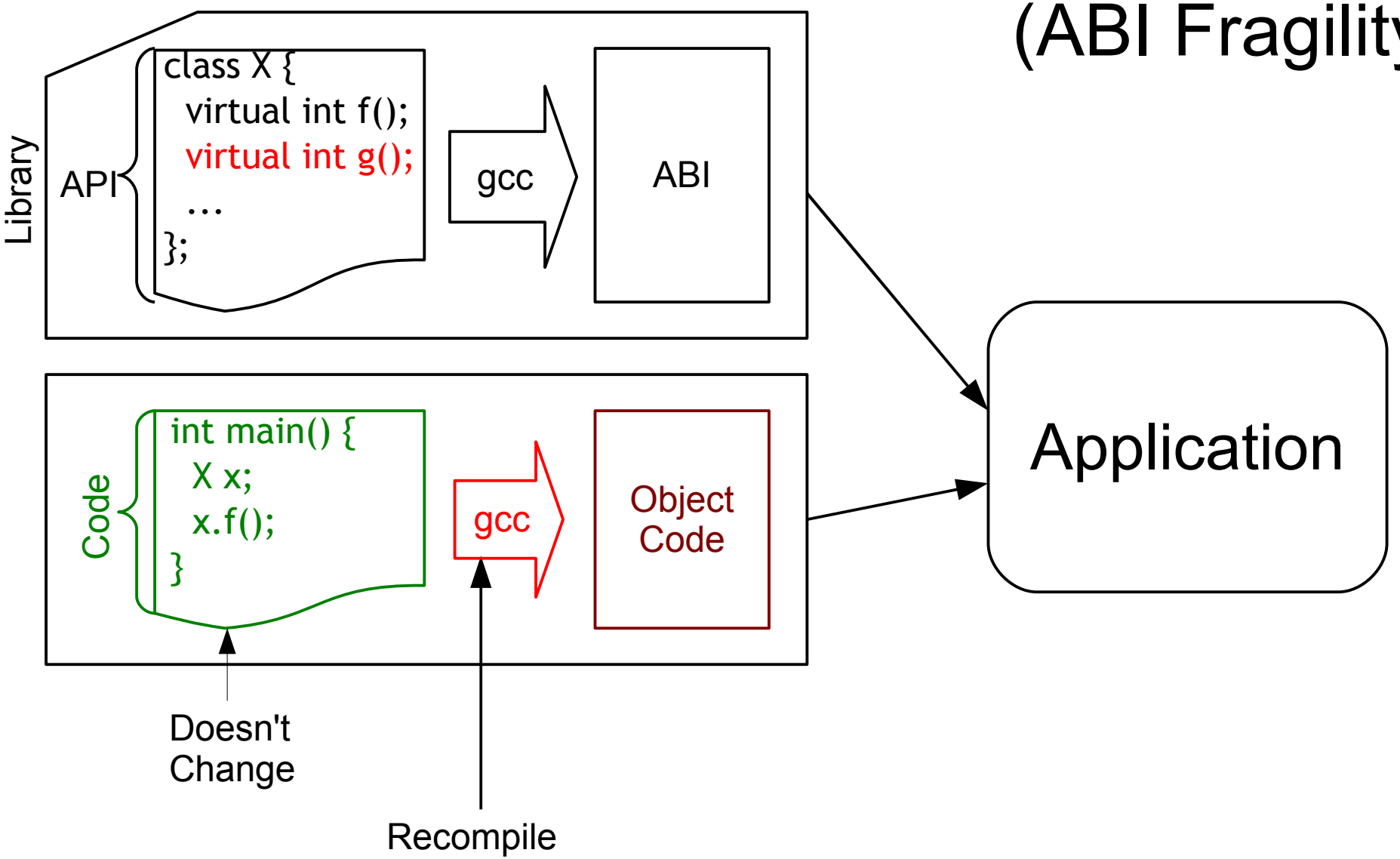


Application Binary Interface (ABI)

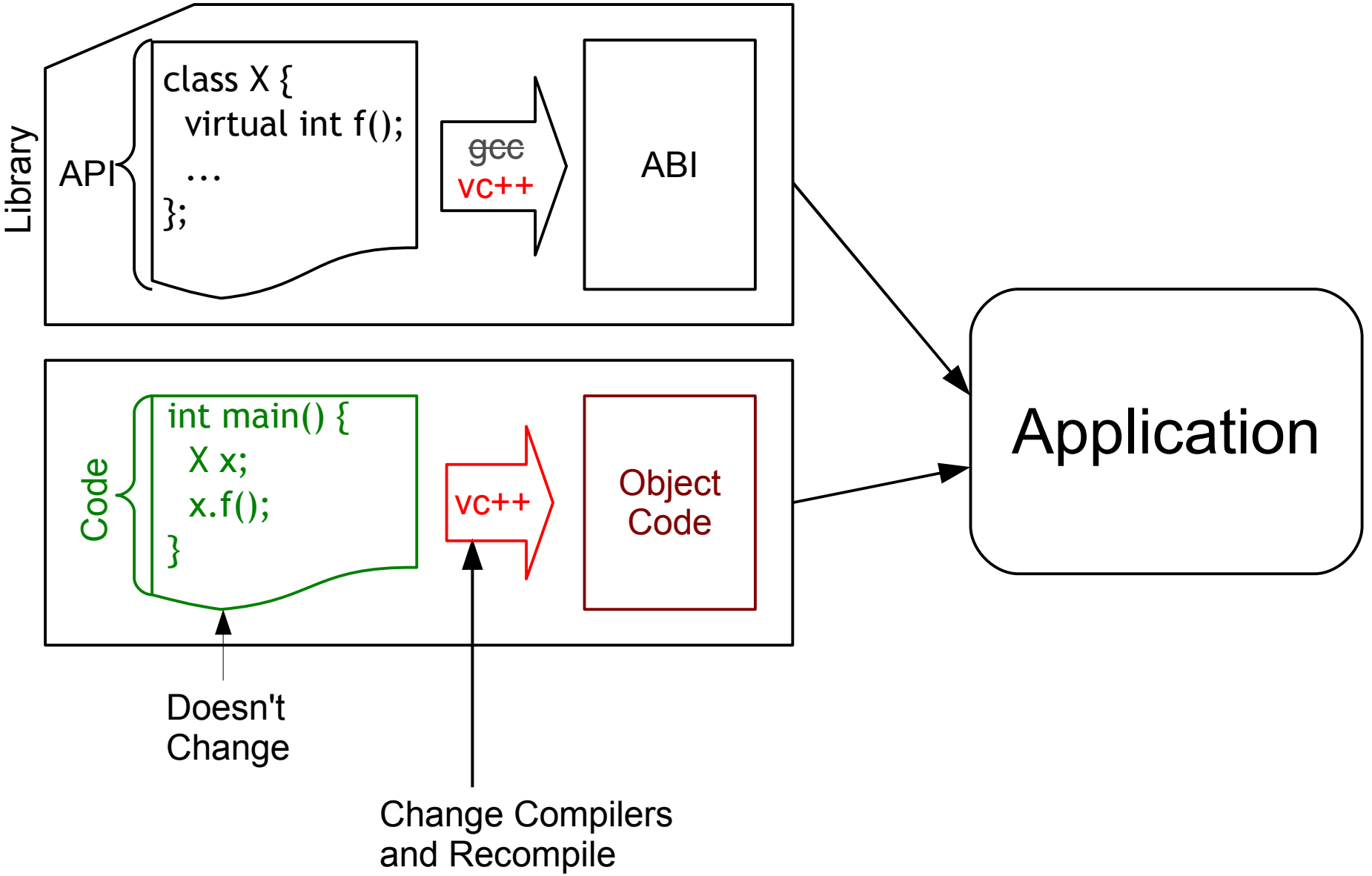


Source vs Binary Compatibility

(ABI Fragility)



ABI Incompatibility



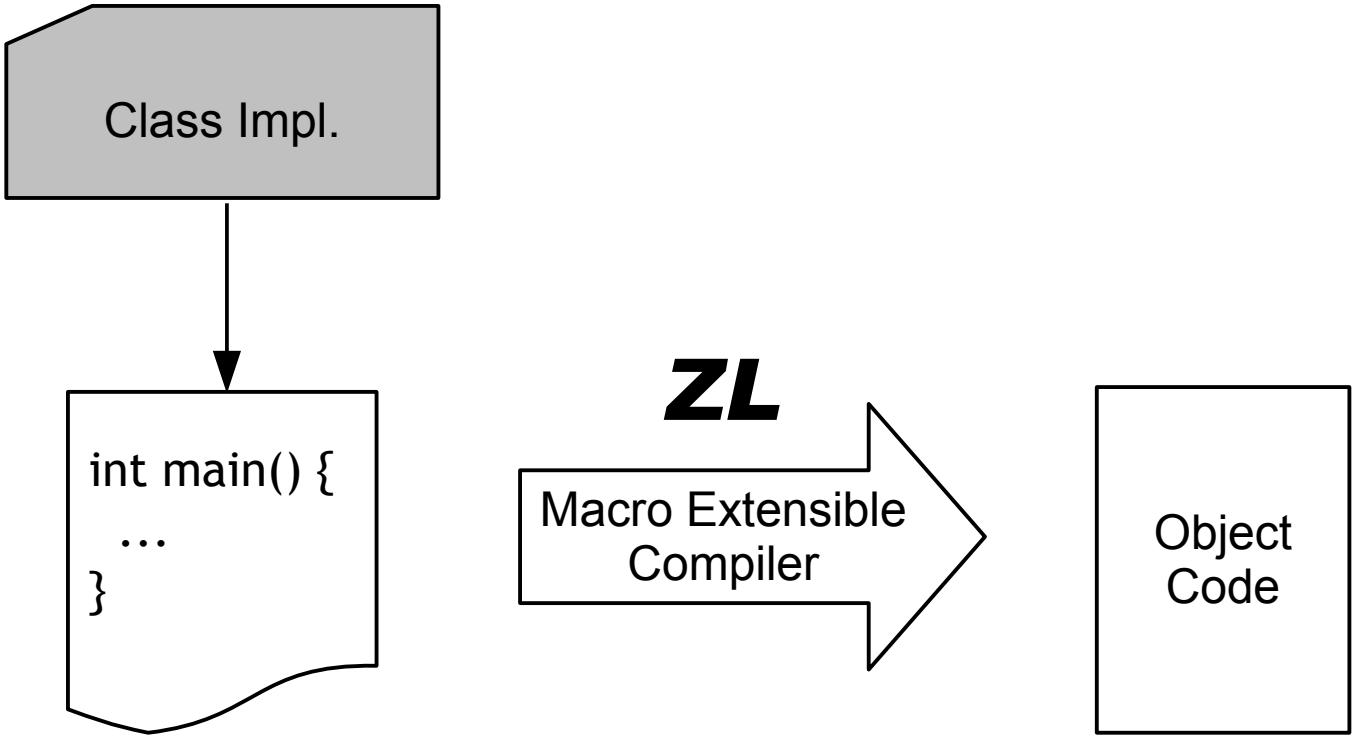
Previous Work

- Some Standardization Attempts
  - Itanium C++ ABI
- Still Two Common ABIs:
  - GCC
  - Visual C++

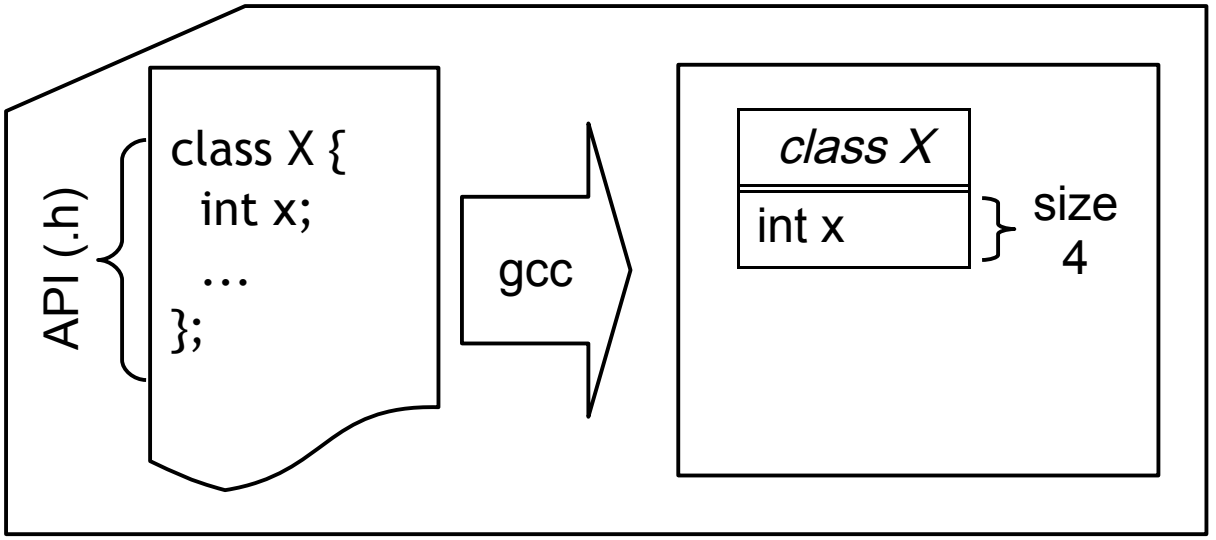
- Some Work Towards Less Fragile ABIs
  - Delta C++ (Palay, 1992)
  - Object Binary Interface (Williams and Kindel, 1994)
- Sacrifice Performance
- Not Commonly Used

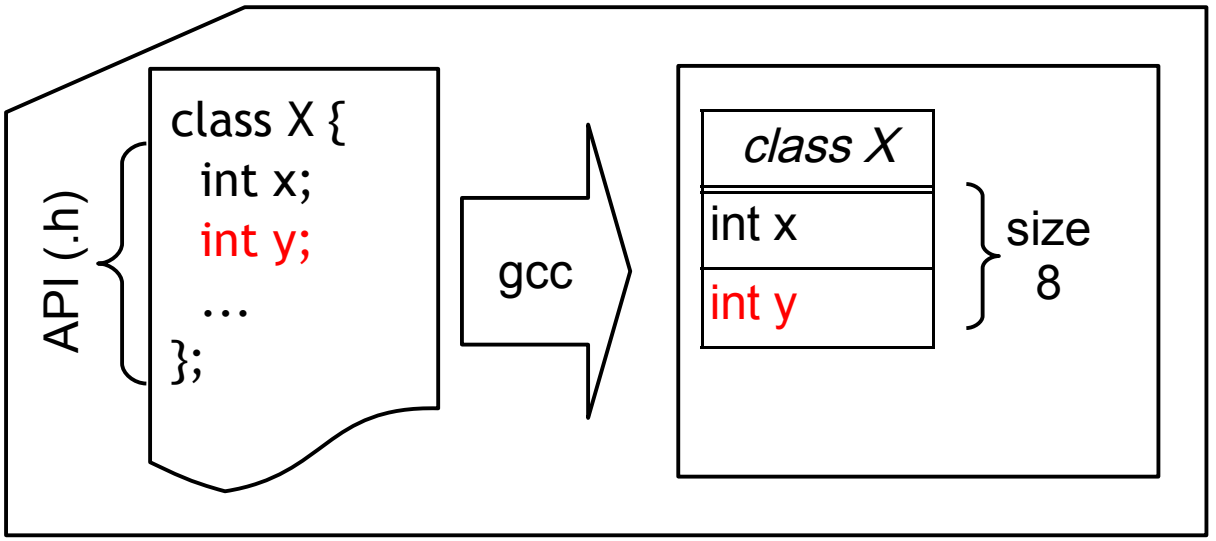
Our Contribution

ABI Compatibility Through Macros



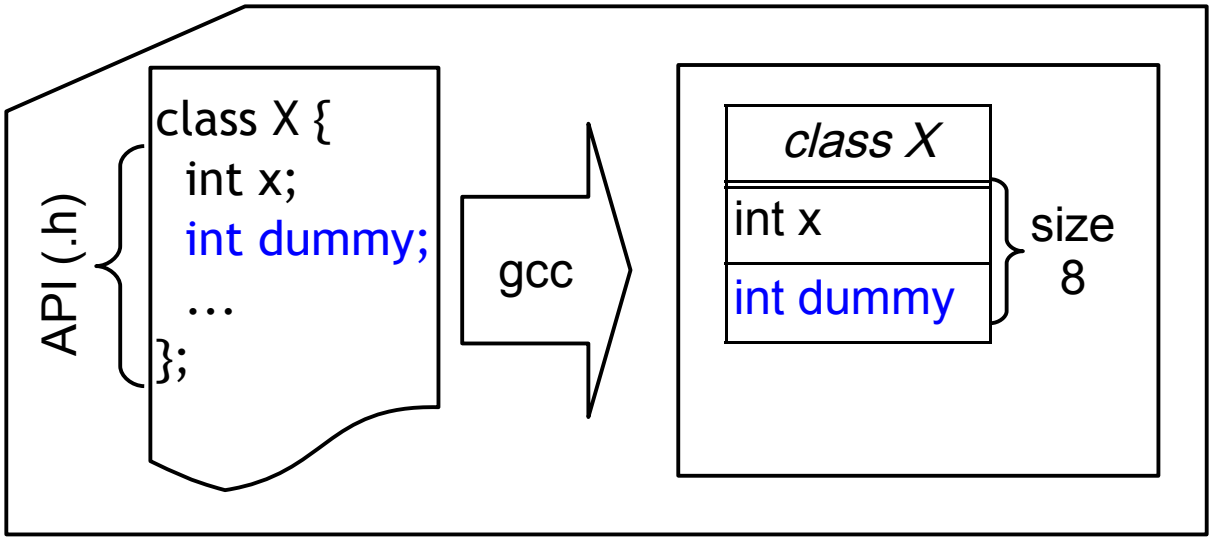
Breaking Binary Compatibility

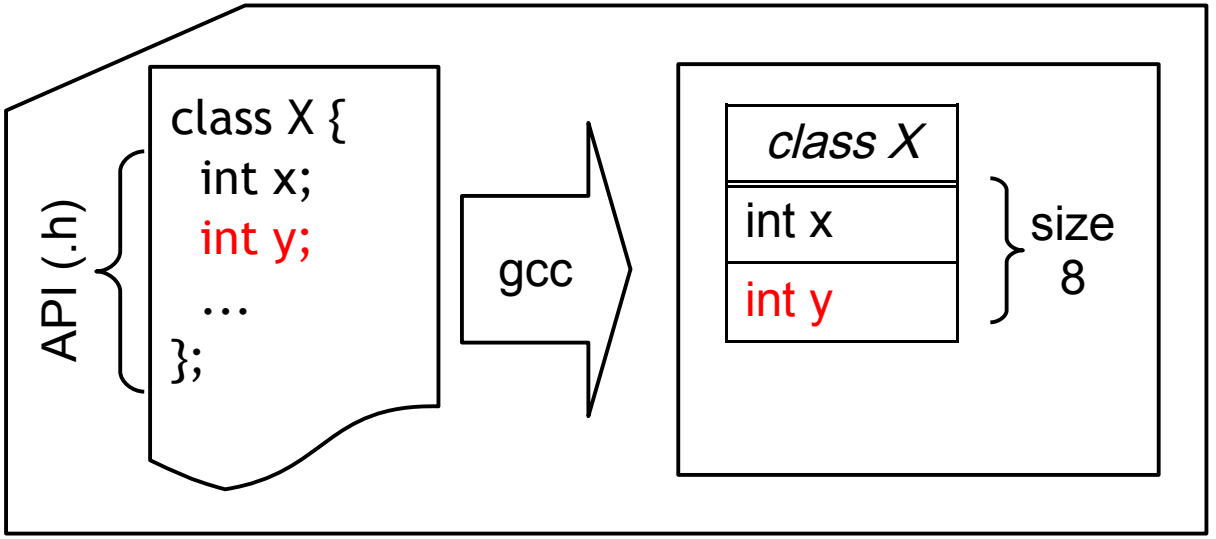




Preserving Binary Compatibility

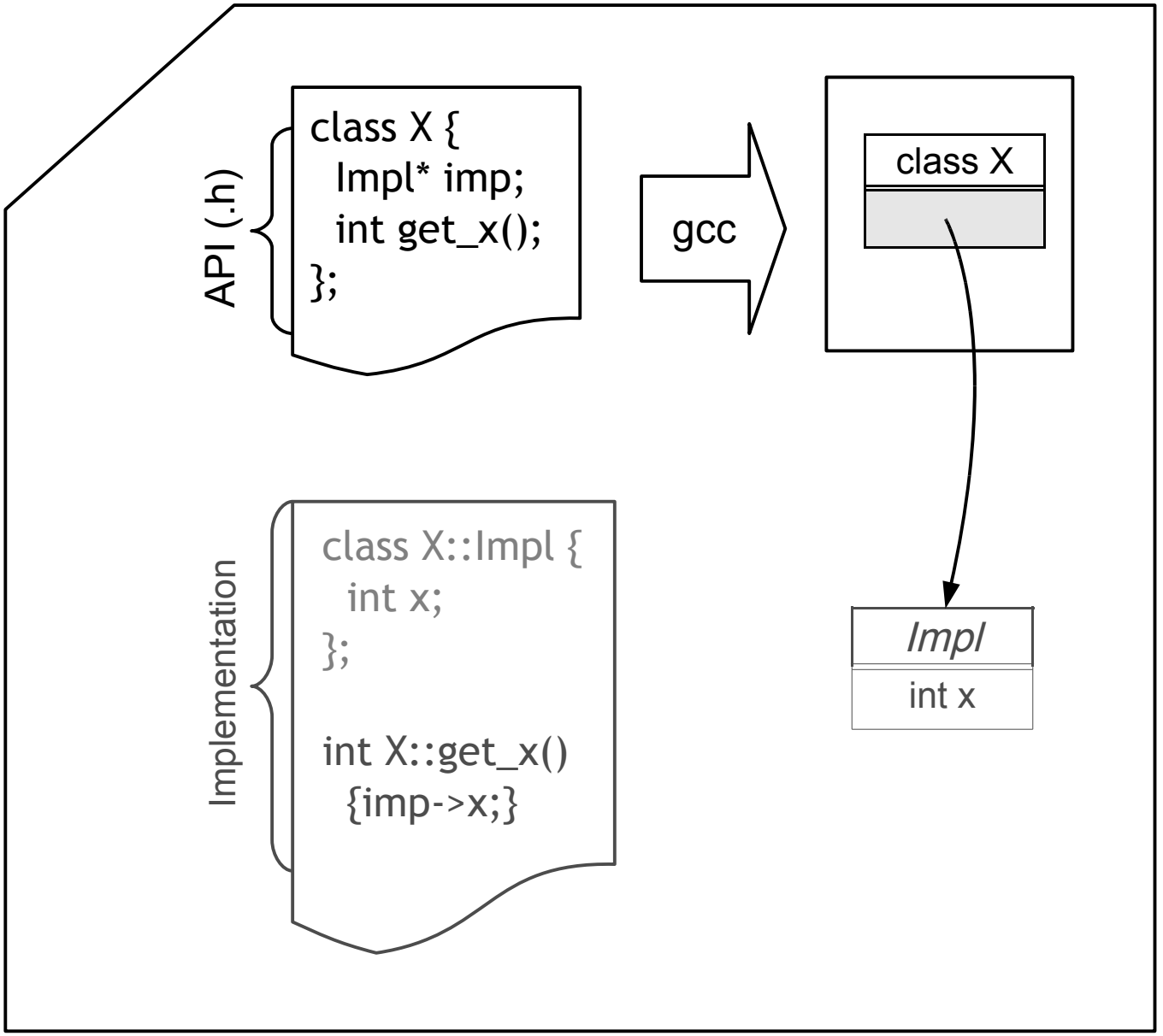
Dummy Member



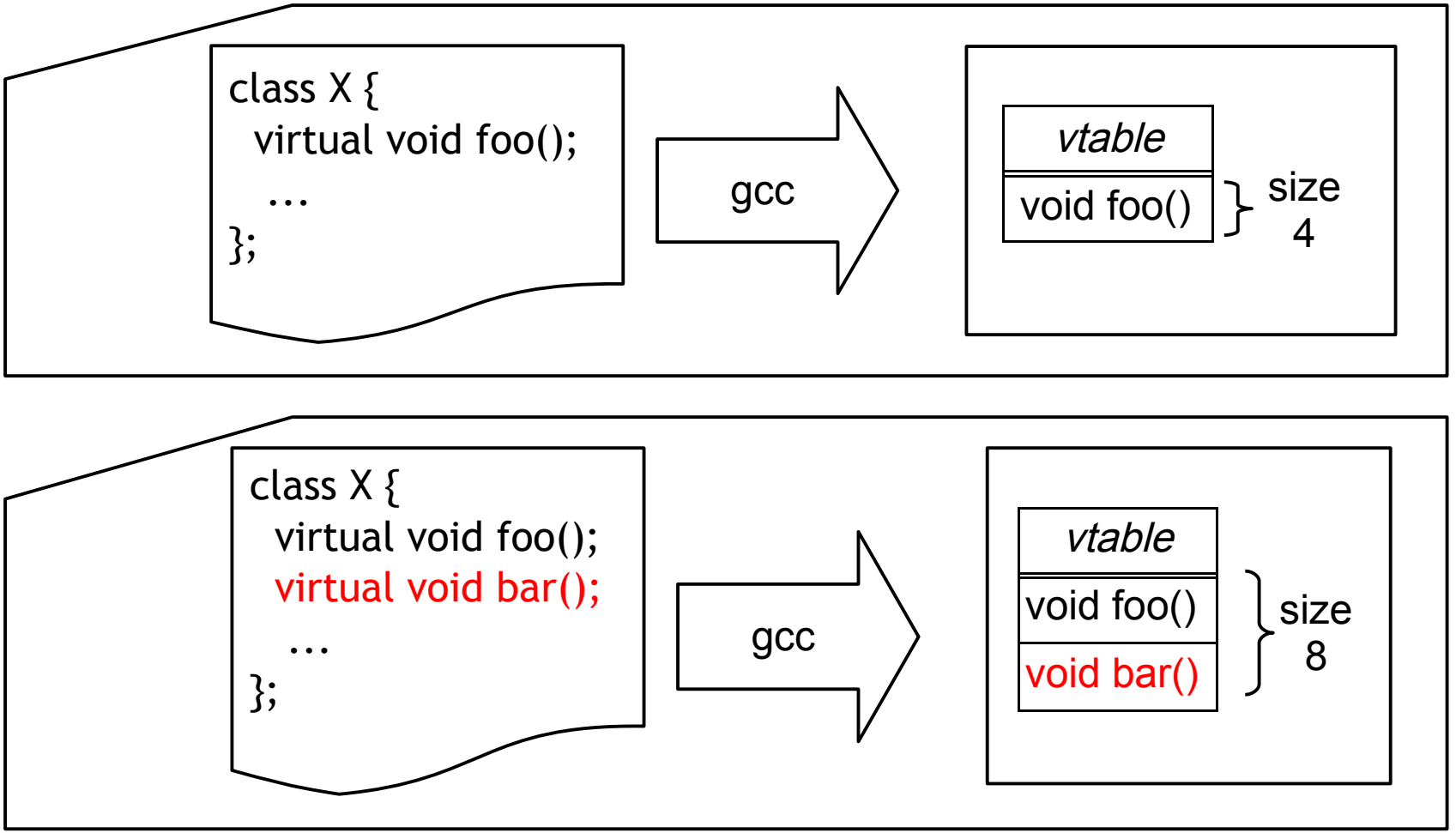


Preserving Binary Compatibility

Pointer to Implementation

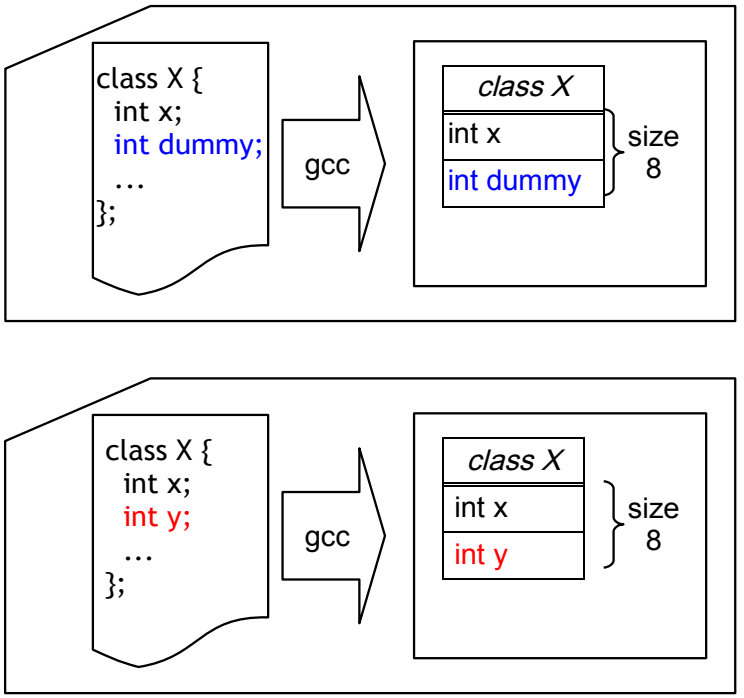


Breaking Binary Compatibility

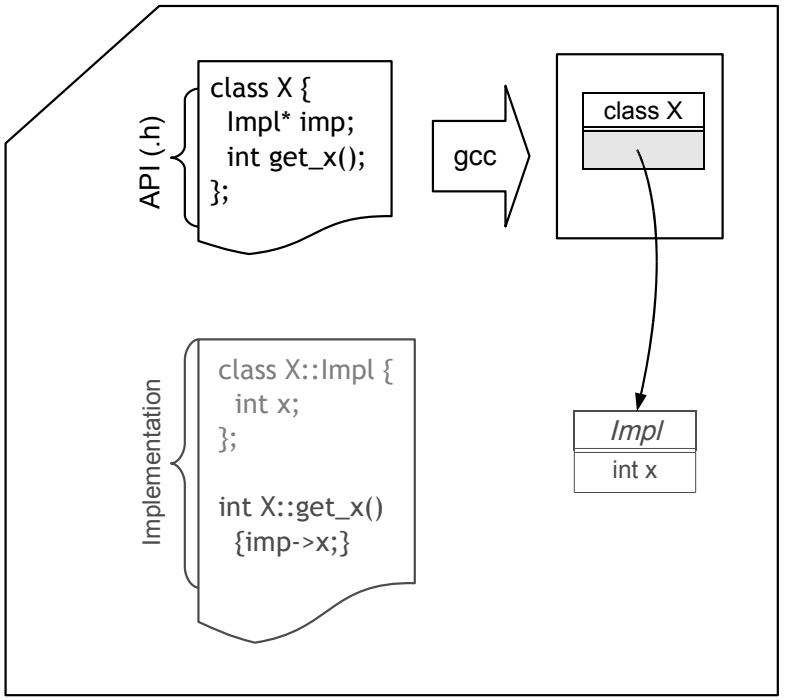


- For Preserving: Options Limited

Dummy Member



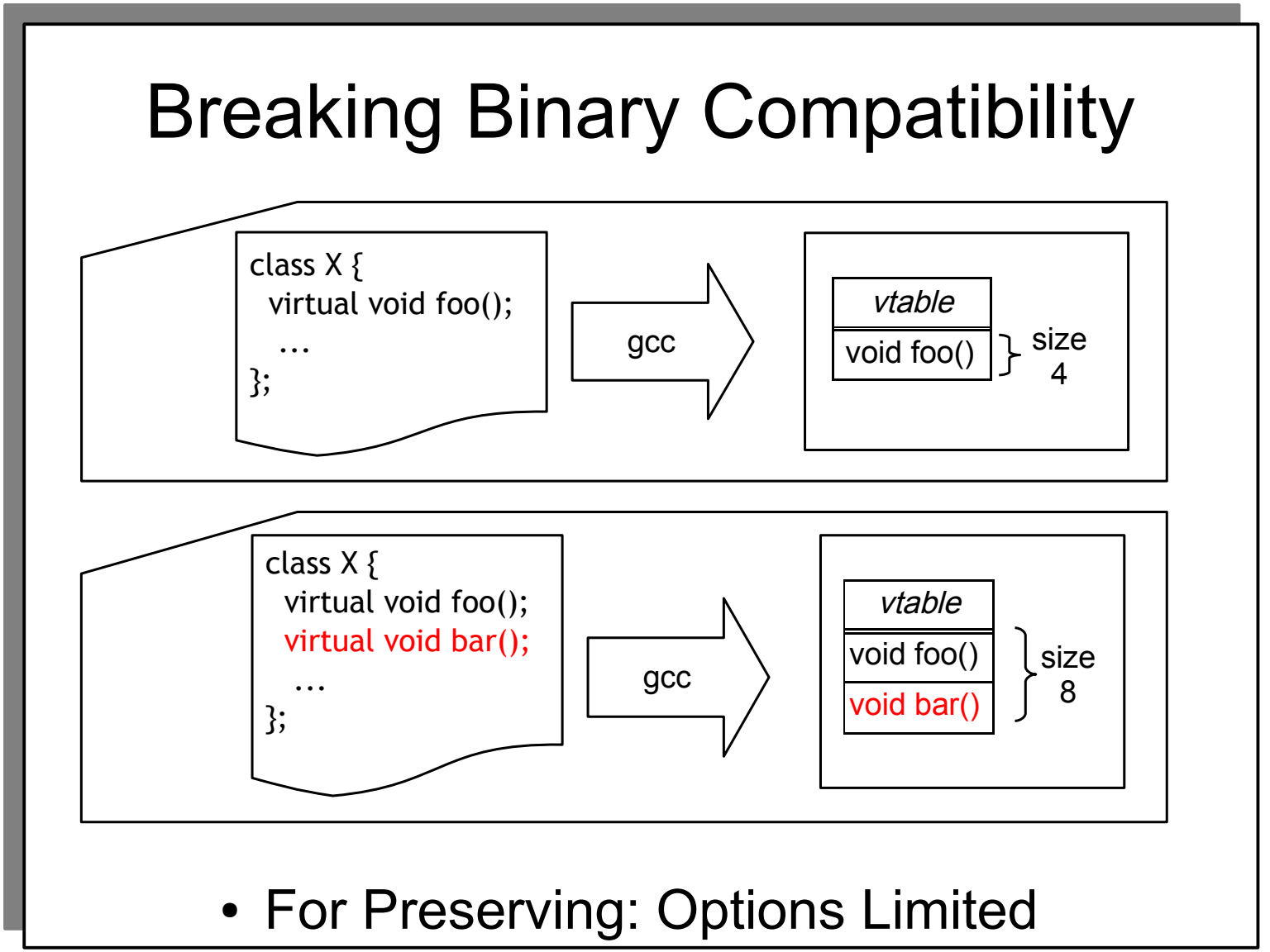
Pointer to Impl.

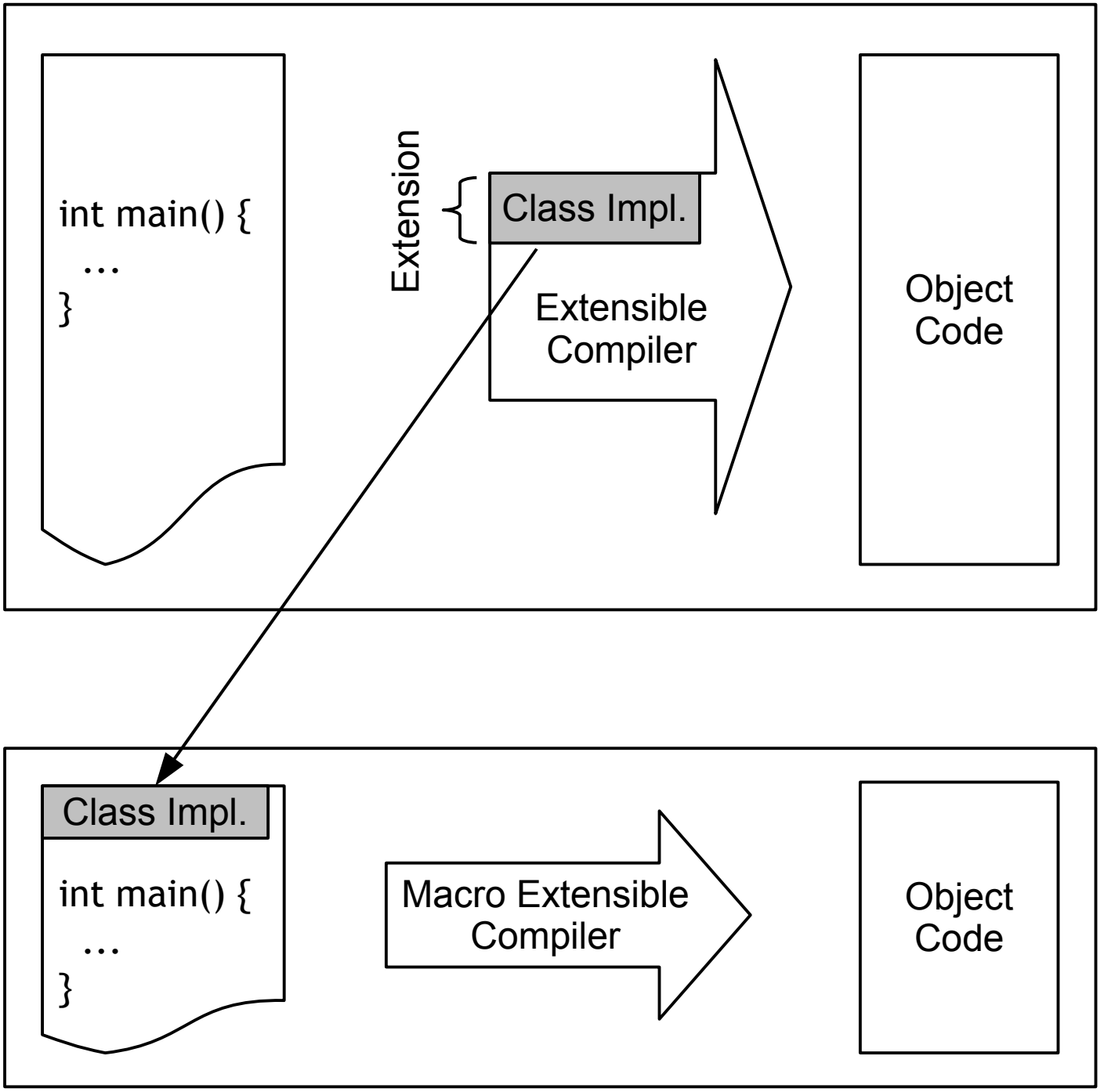


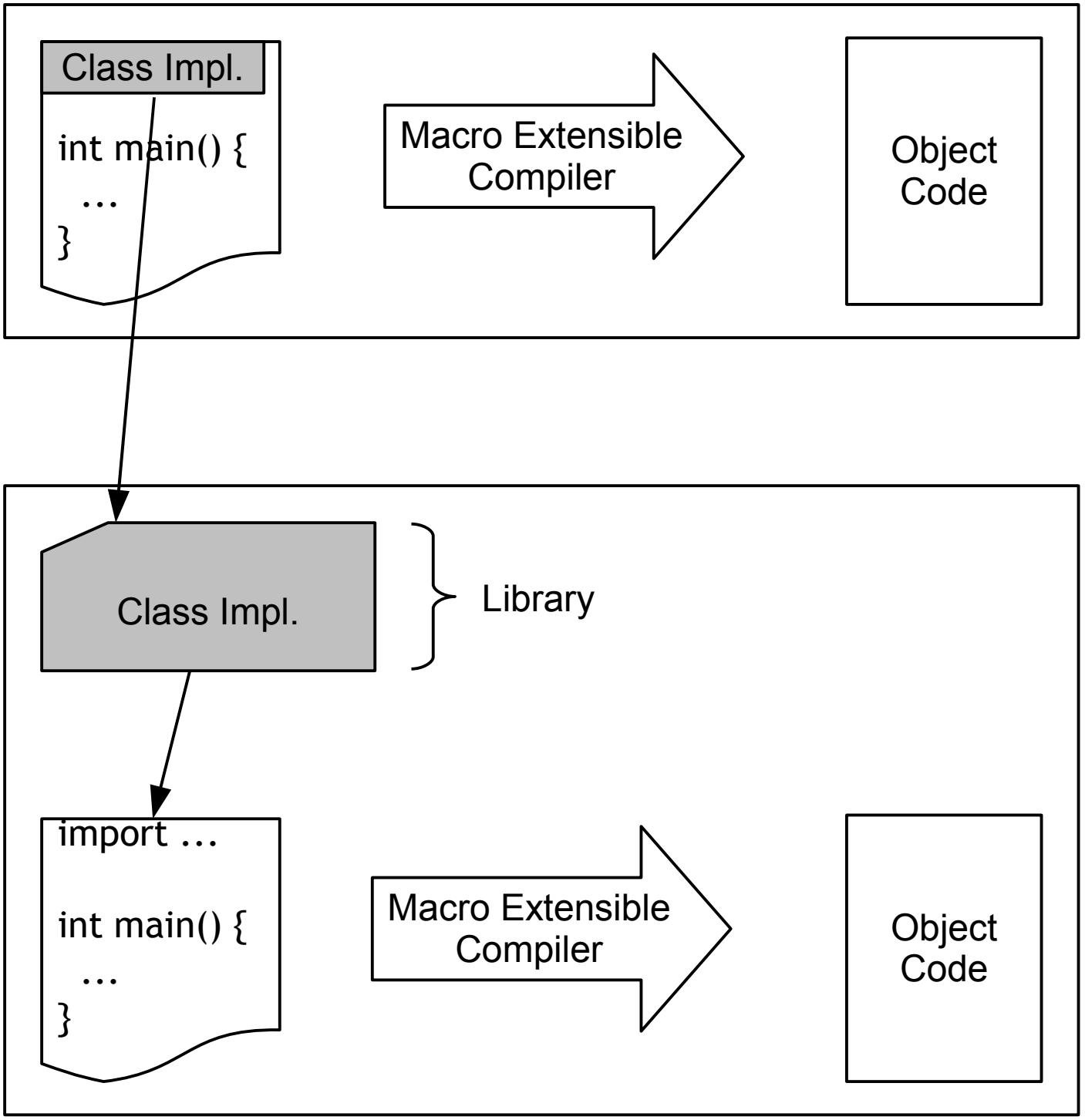
…

Automate These Patterns

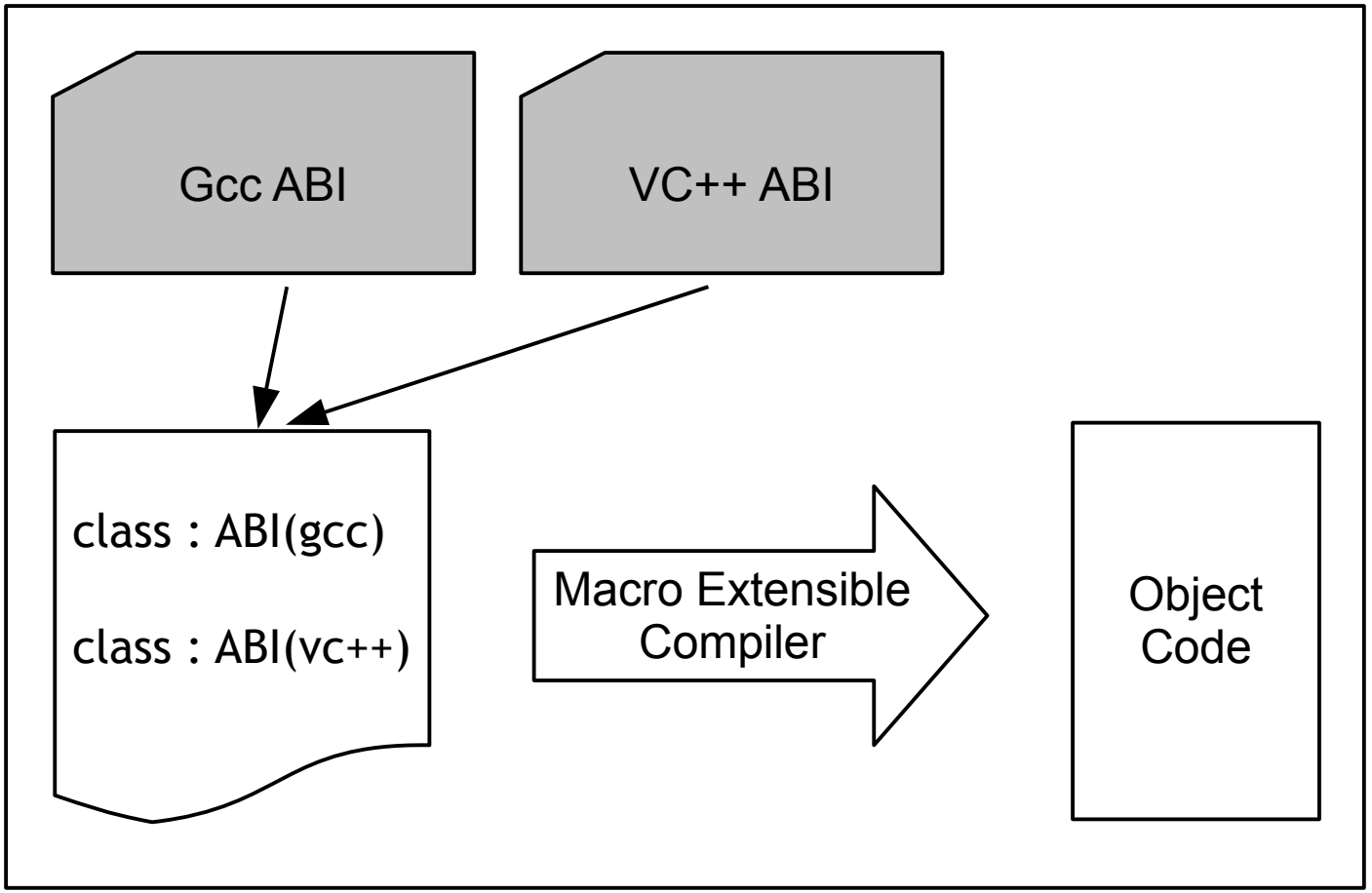
Automation is Not Enough...



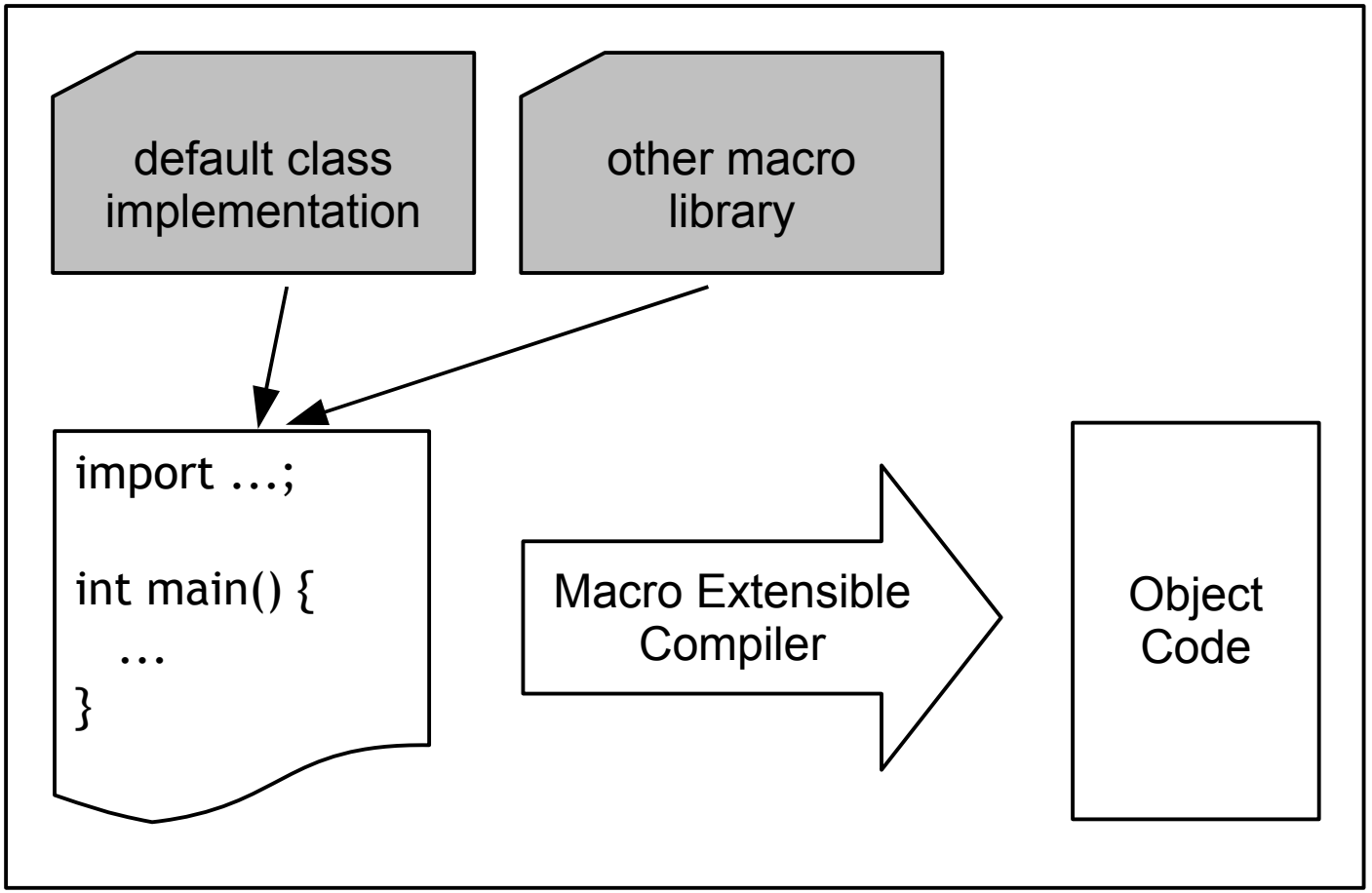

int main() {
…
}
Extension
Class Impl.
Extensible
Compiler
Object
Code
Class Impl.
int main() {
…
}
Macro Extensible
Compiler
Object
Code


Class Impl.
int main() {
…
}
Macro Extensible Compiler
Object Code
Class Impl.
Library
import …
int main() {
…
}
Macro Extensible Compiler
Object Code

Two ABI's at Once



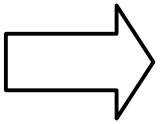
ZL



C Like Core w/ Minimal Amount of C++

Hygienic Pattern Macros

macro or(*x*, *y*) { ({typeof(*x*) t = *x*; t ? t : *y*;}); }

or(0.0,t)  ({typeof(0.0) t_0 = 0.0; t_0 ? t_0 : t;});

Procedural Macros

```
Syntax* foreach(Syntax*) {...}
make_macro foreach

int main() {
  Container c;
  ...
  foreach(el, c, el.print());
}
```

Syntax Forms:
- new_mark
- syntax
- make_macro

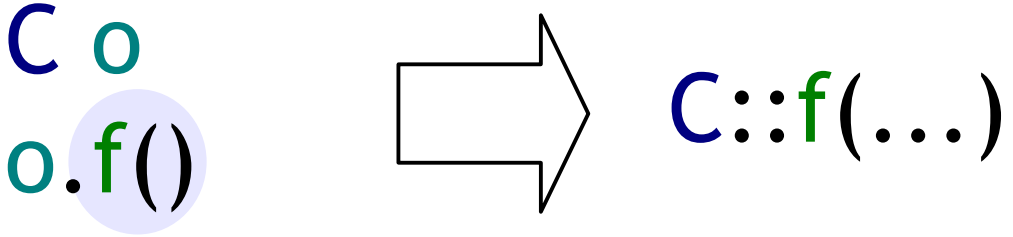
Callbacks:
- match
- replace
- ct_value
- error

The Class Macro

```
import_file "class-simple.zlh";

...

.class ParseClass {
  ...
  virtual Syntax * parse(Syntax * p) {...}
  virtual void init() {...}
  virtual void init_parent_info() {...}
  ...
};

Syntax * parse_class(Syntax * p) {
  ParseClass * pc = new ParseClass;
  return pc->parse(p);
}
make_macro class parse_class;
```

900 Lines of Code

Expanding Method Calls

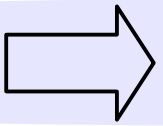


What is f?

User Types
(ZL's Notion of Classes)

```
user_type C {

  struct Data {
    int i;
  };
  associate_type struct Data;

  int foo(int j)

  macro i(:this ths) {...}
  macro f(x, :this ths) {...}
}
```

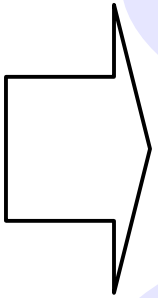
```
void main() {
  C o;
  o.f(x);
  int x = o.i;
}
```



```
void main() {
  C o;
  C::f(x, :this = &o);
  int x = C::i(:this = &o);
}
```

Classes

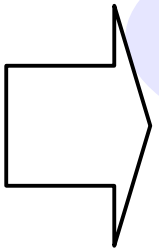
```
class C {

  int i;

  int f(int j)
    {return i + j;}

};
```



```
user_type C {

  struct Data {
    int i;
  };
  associate_type struct Data;
  int f`internal(C * this, int j)
    {return i + j;}

  macro i (:this ths = this)
    {(*(C *)ths)..i;}
  macro f(j, :this ths = this)
    {f`internal(ths, j);}
}
```

Inheritance

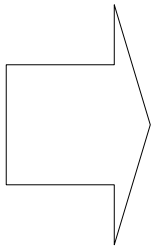
```
class D : public C
{

 int j;

};
```



```
user_type D {

  import C;

  struct Data {
    struct C::Data parent;
    int j;
  };
  associate_type struct Data;

  make_subtype C _up_cast _down_cast;
  macro _up_cast ...
  macro _down_cast ...

  macro j (:this ths = this) {...}

}
```

Virtual Methods

```
class D : public C
{
  virtual
    void f(int j);
};
```

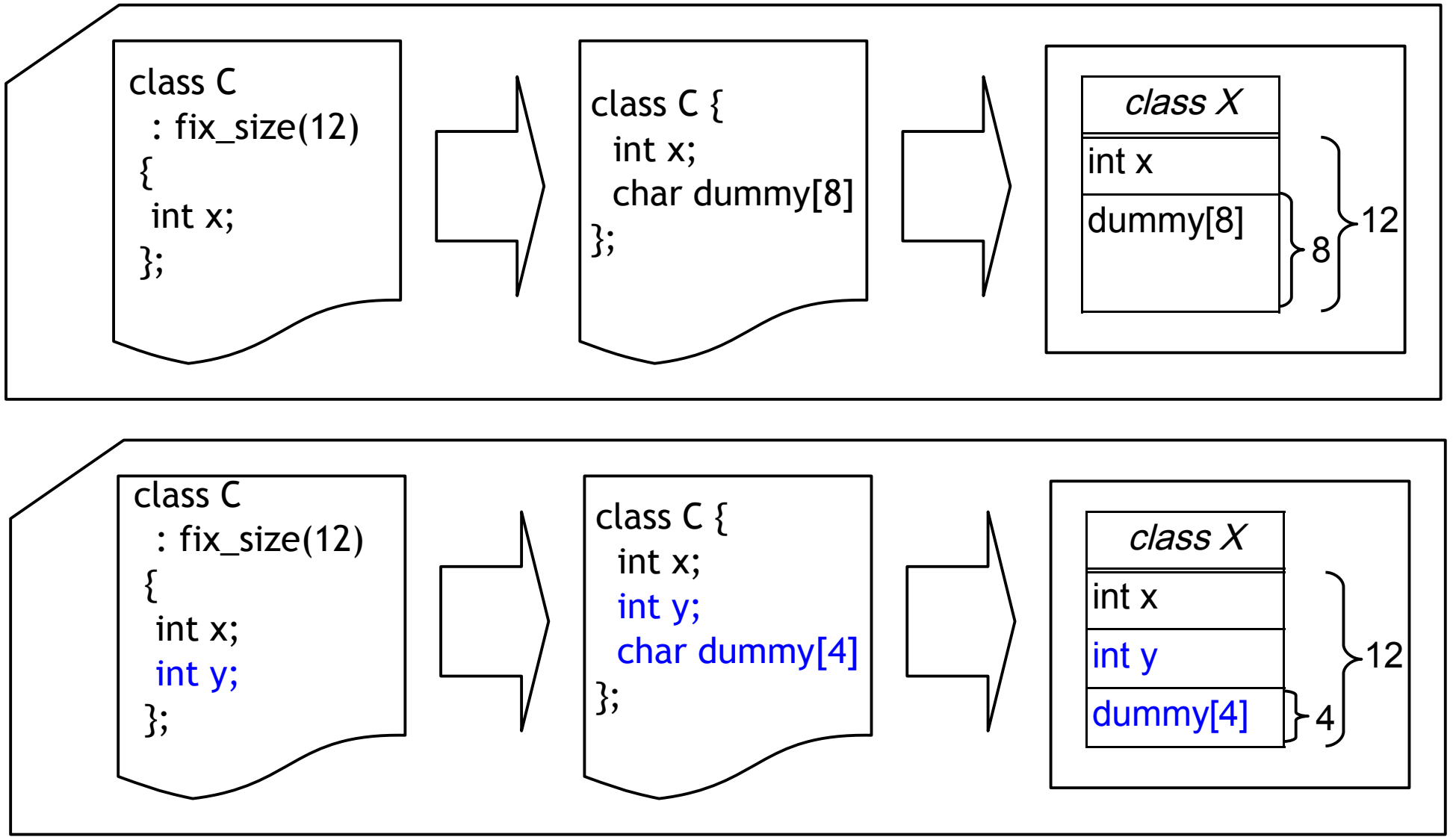


```
user_type D {
  ...

  class VTable : public C::VTable {
    void (*f)(int j);
  };

  struct Data {
    VTable* _vptr;
    ...
  };
  associate_type struct Data;

  macro f(j, :this ths = this)
    {_vptr->f(ths, j);}

  ...
}
```

Examples

Mitigating the ABI Problem Through Macros

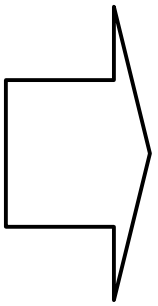
Example: Fixing Size of Class

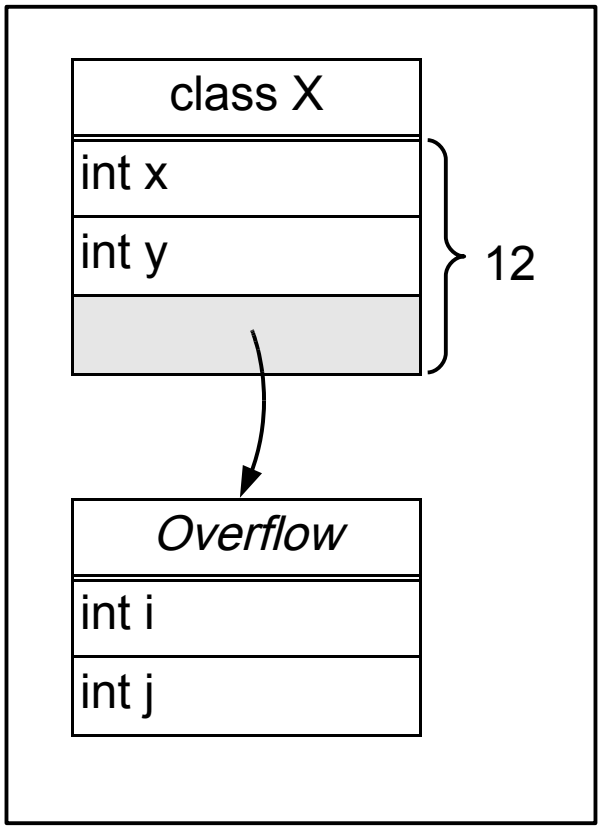


(Complete Macro In Paper, 36 Lines of Code)

Example: Allowing More Fields

```
class C :
  fix_size(12)
{
  int x;
  int y;
  int i;
  int j;
};

int main() {
  C obj;
  obj.x;
  obj.i;
}
```

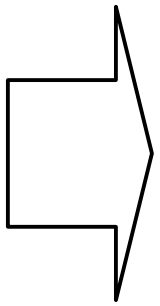


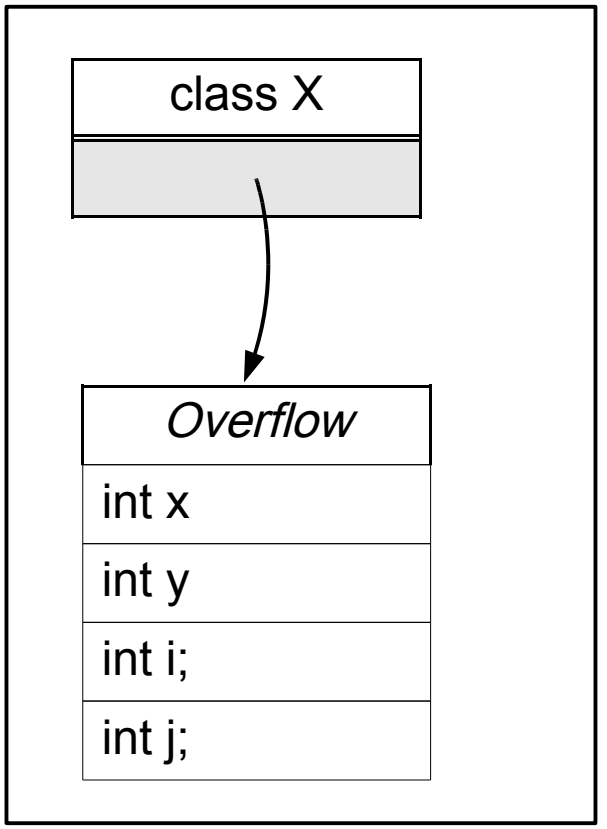


(Expanded Macro, Around 100 Lines of Code)

Example: Pointer to Impl.

```
class C :
  fix_size(0)
{
  int x;
  int y;
  int i;
  int j;
};

int main() {
  C obj;
  obj.x;
  obj.i;
}
```

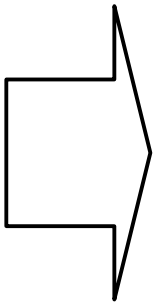




(Same Macro, Around 100 Lines of Code)

Fixing Virtual Table Size

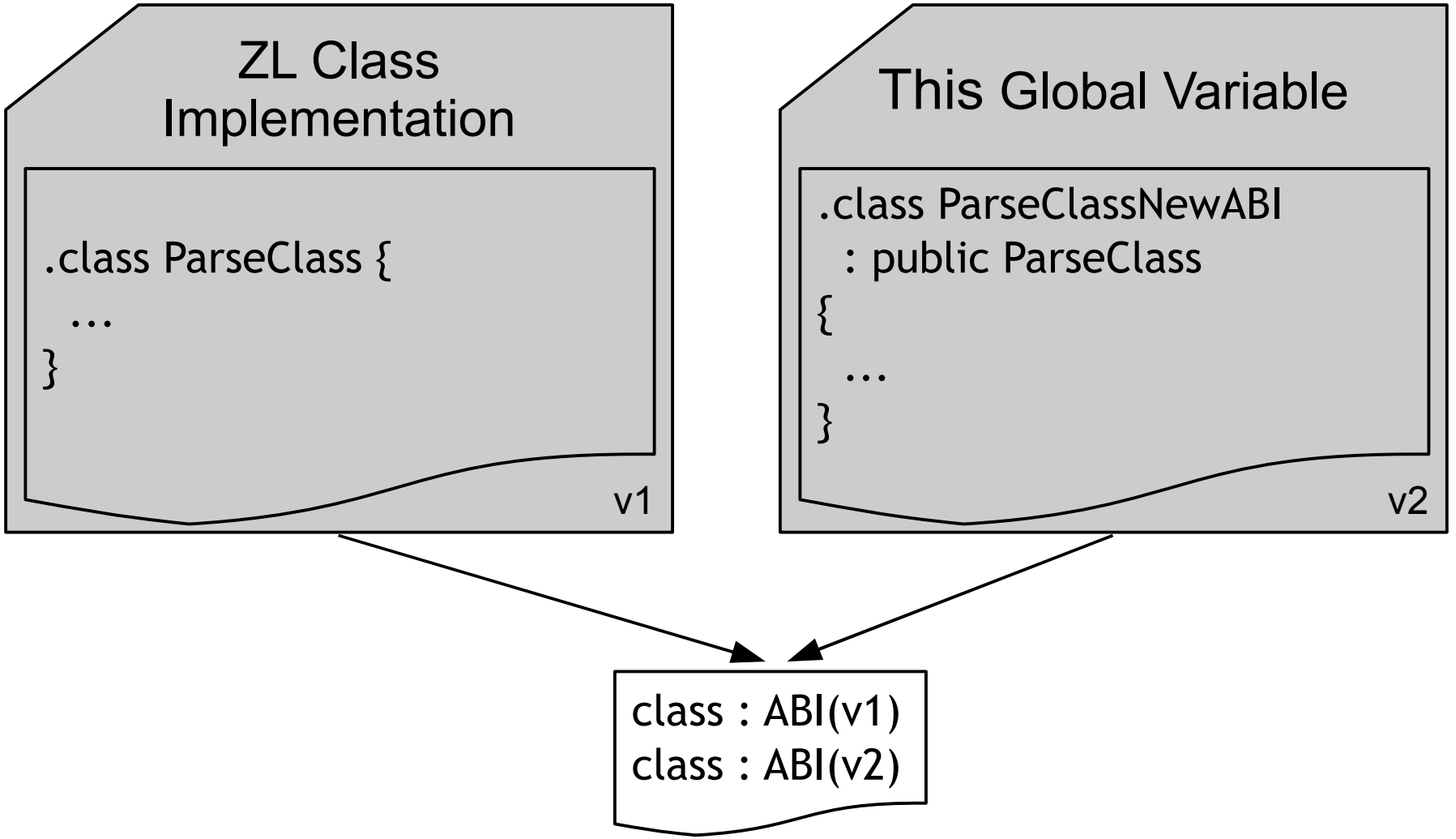
```
class X
  : fix_vtable_size(8)
{
  virtual void f();
  virtual void g();
  virtual void h();
};
```



```
macro alt_vtable (name, body) {
  class name : fix_size(8) {body};
}

class X : vtable_class(alt_vtable) {
{
  virtual void f();
  virtual void g();
  virtual void h();
};
```

(Modified Macro, Around 120 Lines of Code)

Matching Other ABIs



Implemented In Under 45 Lines of Code

ZL Implementation Status

- Prototype Compiler
- Most Of C
- Enough C++ to Demonstrate Approach
- Compiled Several Real Applications
  - C: bzip2, gzip
  - C++: randprog
- Compile Time 2-3 slower
- No Impact on Performance

Conclusion

Demonstrated
ABI Compatibility Through Macros