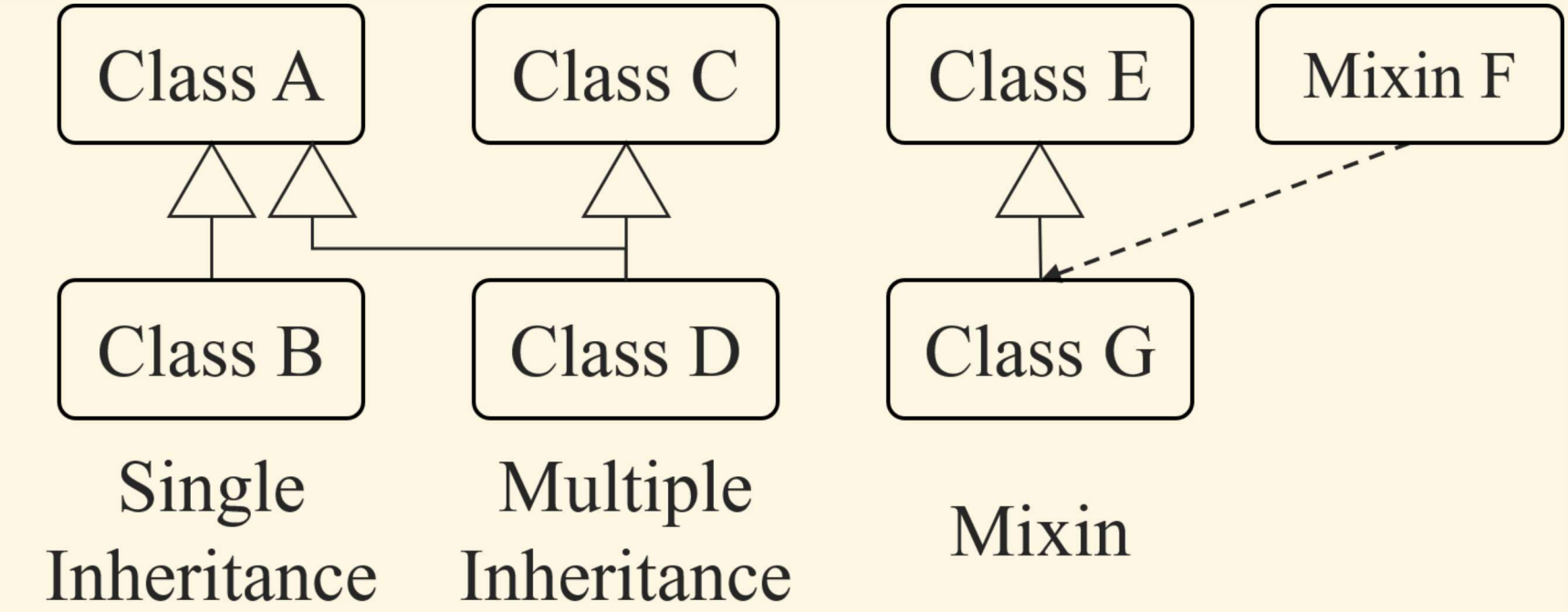
Traits: Composable Units of Behavior

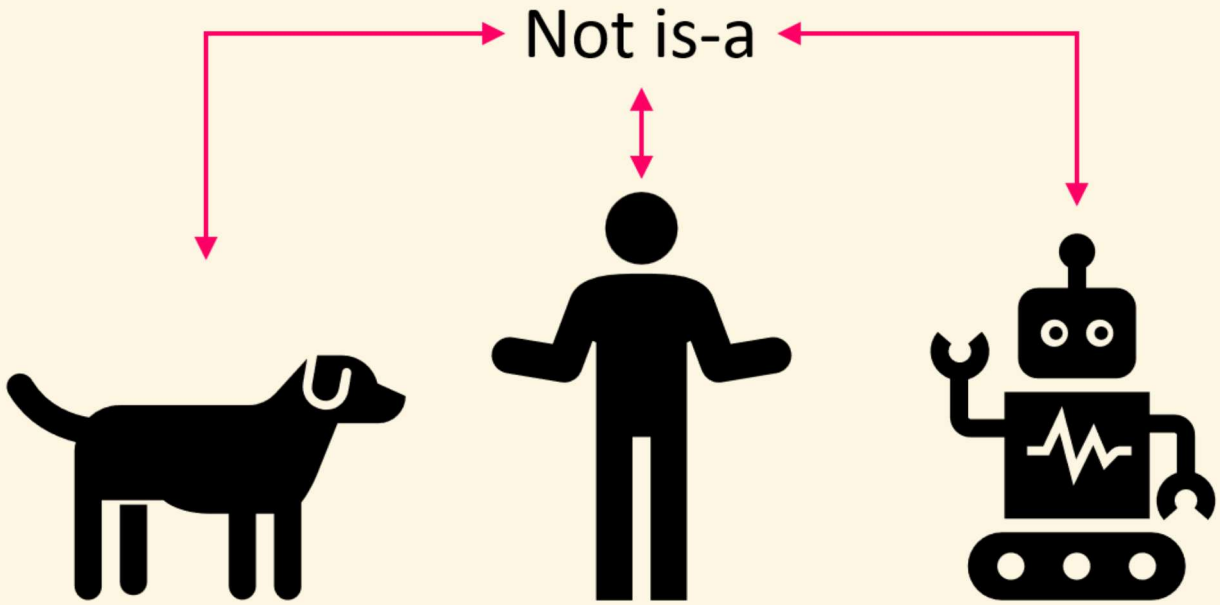
ECOOP 2003

Nathanael Schärli, Stéphane Ducasse, Oscar Nierstrasz and Andrew Black

Background

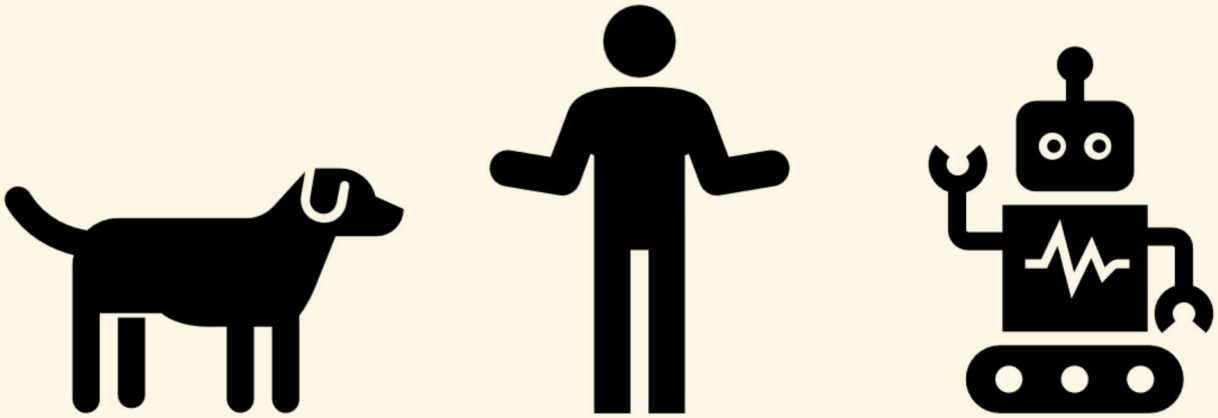

Class A
Class C
Class E
Mixin F
Class B
Class D
Class G
Single Inheritance
Multiple Inheritance
Mixin

Issue: limited compositional power



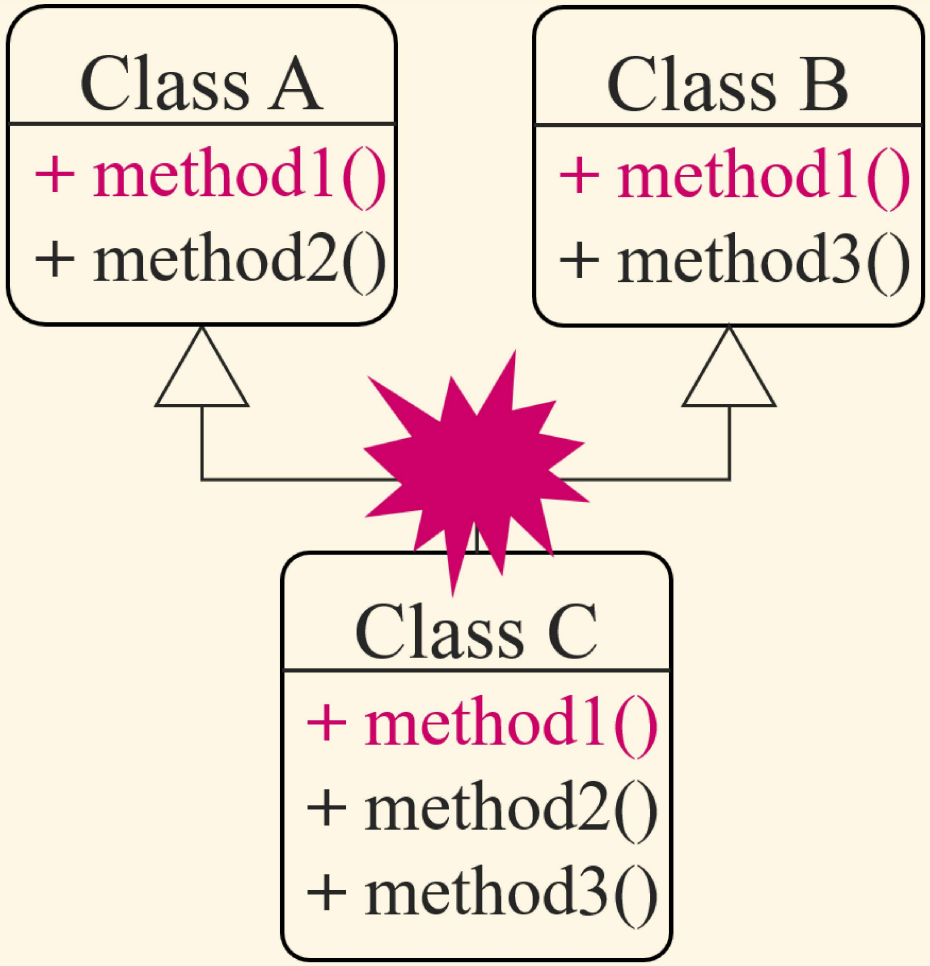
Run	Run	Run
Walk	Walk	Walk
Bark	Talk	Photon Power Beam

Issue: limited compositional power

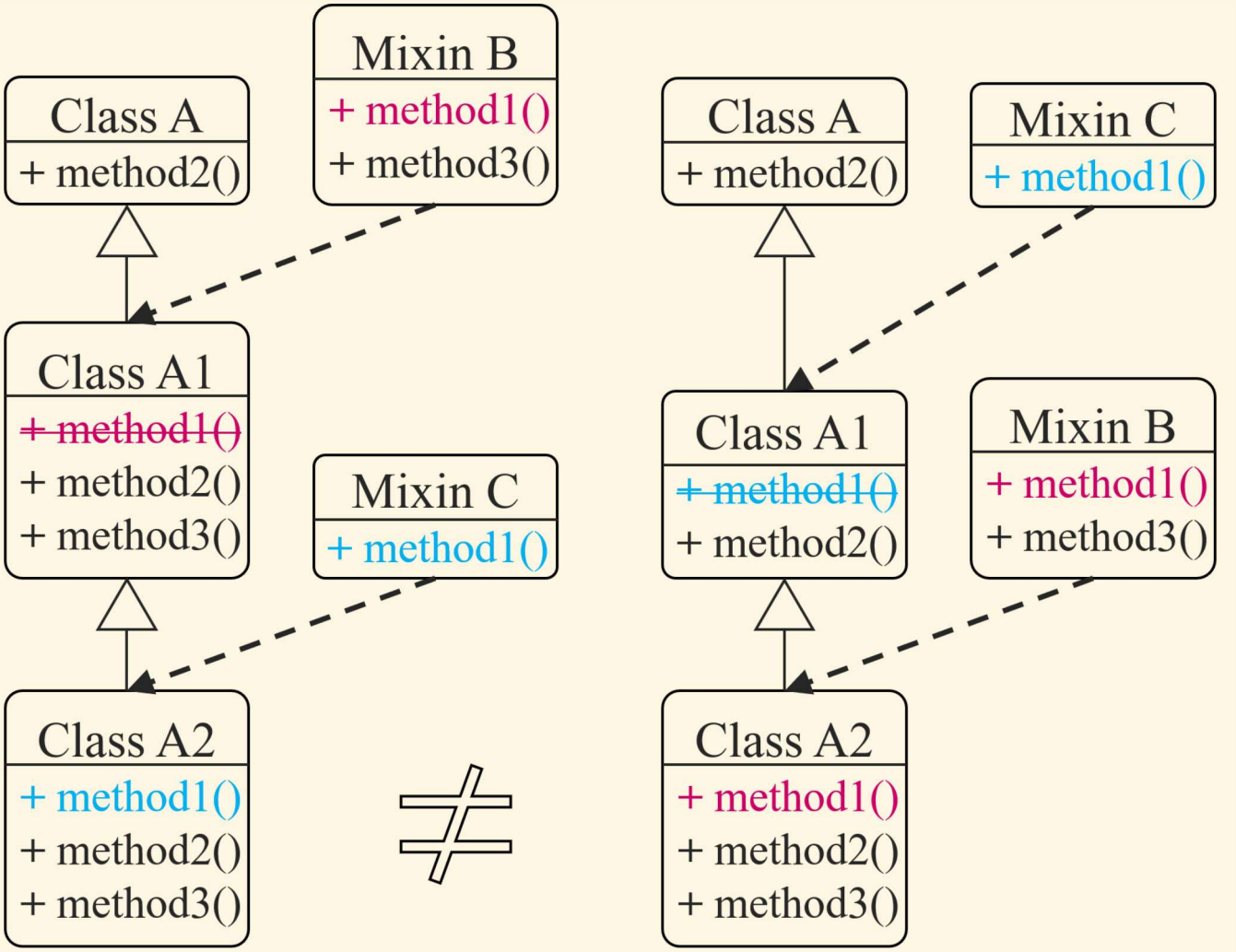


Run	Run	Run	Moveable
Walk	Walk	Walk	
Bark	Talk	Photon Power Beam	

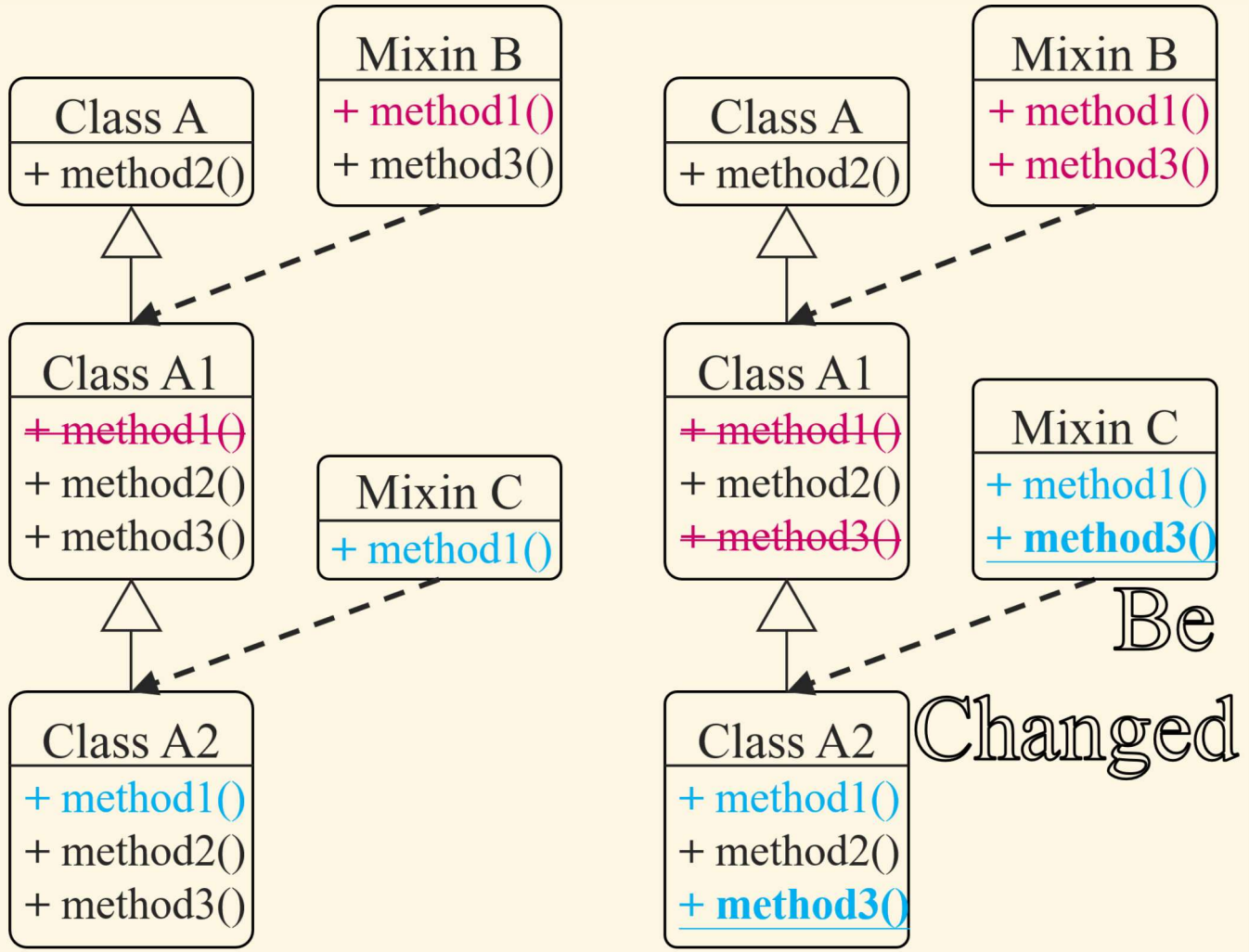
Issue: conflicting features



Issue: total ordering



Issue: fragile hierarchies



Issue (cont.)

- Limited compositional power.
- Conflicting features.
- Accessing overridden features.
- Total ordering.
- Dispersal of glue code.
- Fragile hierarchies.

Issue (cont.)

- Single Inheritance
  Not expressive enough to factor out all the common features.
- Multiple Inheritance
  Conflicting features, accessing overridden features, limited compositional power.

Issue (cont.)

- Mixin Inheritance
  Related to mixing order.
- Interface
  Not provide any help with the problem of code duplication.

Traits

$$\begin{aligned} Class &= \mathbf{Traits} + State + Glue \\ &= \mathbf{Traits} + Class' \end{aligned}$$

Traits (cont.)

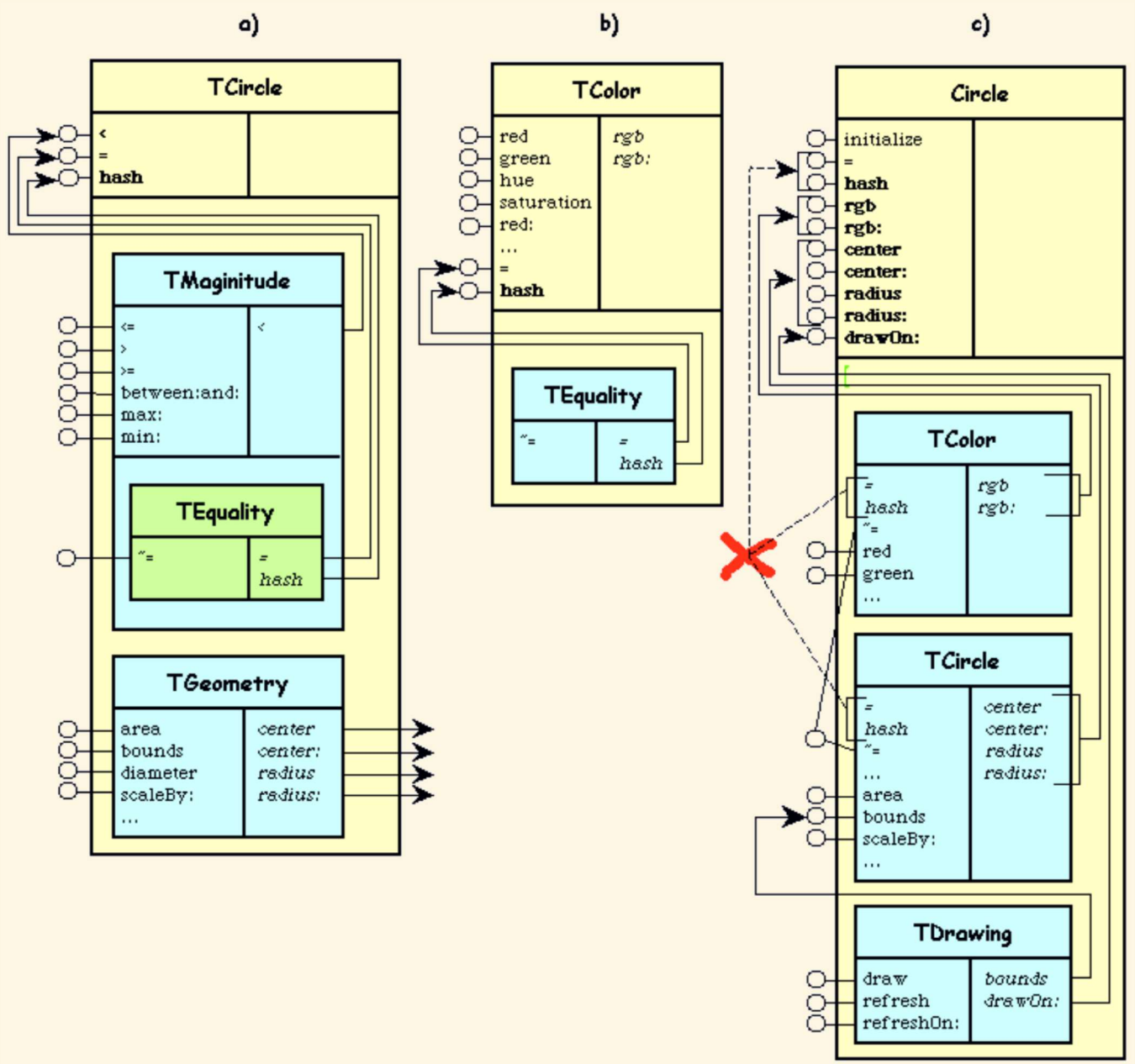
Properties

- Grouped by functionality.
- A set of methods that implement behaviour.
- Non state.
- Flattening. (Order independent)
- Can be nested, but nested traits are equivalent to flattened traits.

Traits (cont.)

Flattening

1. $T = T_1 + D,\ D = T_2 + T_3$
   $\Rightarrow T = T_1 + T_2 + T_3$
2. $Class > Trait_1 = Trait_2 = \cdots$
   $= Trait_n > Super\ Class$
3. Alias: If there is a conflict between traits, you can alias these methods or exclude them from composition.


a)
b)
c)
TCircle
<
=
hash
TMaginitude
<=
>
>=
between:and:
max:
min:
<
TEquality
~=
=
hash
TGeometry
area
bounds
diameter
scaleBy:
...
center
center:
radius
radius:
TColor
red
green
hue
saturation
red:
...
=
hash
rgb
rgb:
TEquality
~=
=
hash
Circle
initialize
=
hash
rgb
rgb:
center
center:
radius
radius:
drawOn:
TColor
=
hash
~=
red
green
...
rgb
rgb:
TCircle
=
hash
~=
...
area
bounds
scaleBy:
...
center
center:
radius
radius:
TDrawing
draw
refresh
refreshOn:
bounds
drawOn:

Traits (cont.)

$$
\begin{aligned}
T ::= \; & D \\
| \; & D \; with \; T \\
| \; & T_1 + T_2 \\
| \; & T - x \\
| \; & T[x \to y]
\end{aligned}
$$

Conclusion

- More lightweight than classes
- Non define state, so the diamond problem does not arise.
- Use method aliasing to avoid both tangled class references in the source and code.

Conclusion (cont.)

- Not impose total ordering
- Can be combined with inheritance, which allows a wide variety of partially ordered compositions.
- Separates the glue code from the code that implements the different aspects.

Thanks for your attention.

Q&A