CoreC++

Multiple Inheritance in C++

Tobias Nipkow

Technische Universität München

What is CoreC++?

Roughly speaking:

$$\text{Jinja} \subset \textit{CoreC++} \subset \text{C++}$$

Intention:

CoreC++ models multiple inheritance exactly as in C++.

Warning:

CoreC++ lacks many C++ features, incl. overloading.

Overview

- Multiple inheritance in C++
- A formal model of *subobjects*
- Examples
- Semantics and type system

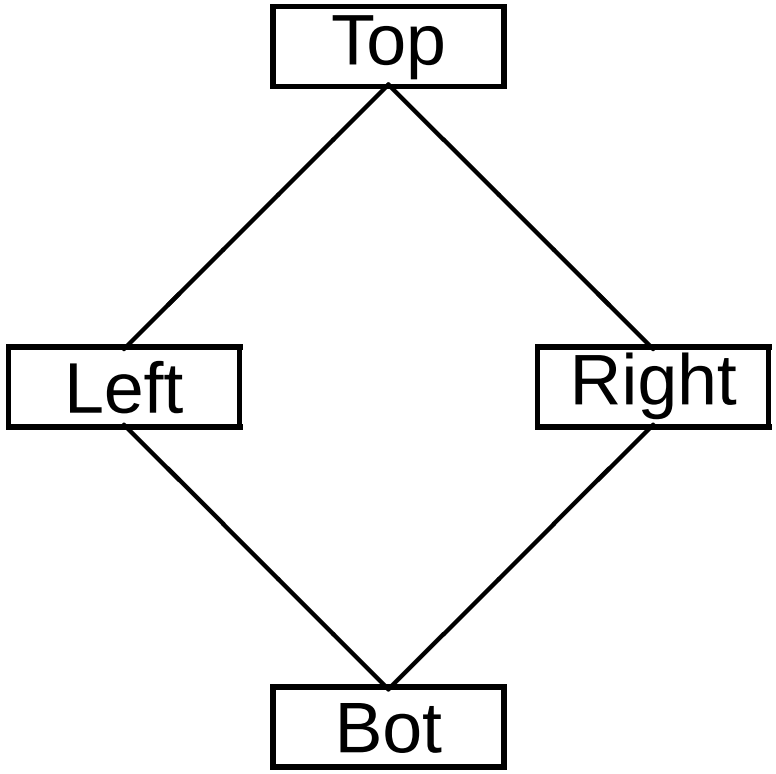
Why multiple inheritance and C++?

Not pretty but millions of lines of code out there:

- Want to understand them
- Want to port them automatically to safer languages

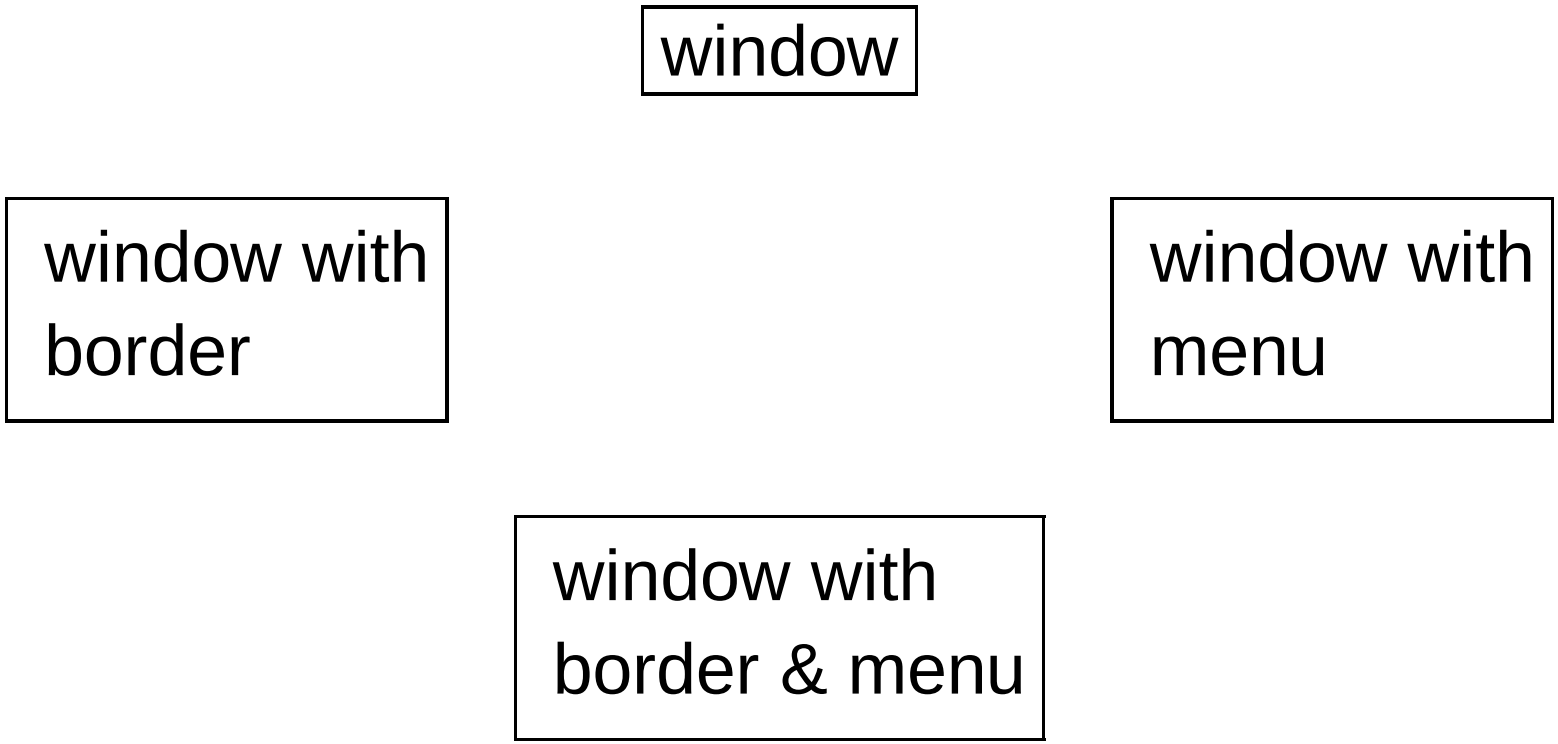
Informal definition of C++: pointers and tables (Stroustrup)
Challenge: abstract formal model

Multiple inheritance



1 or 2 instances of Top in Bot?

Shared inheritance

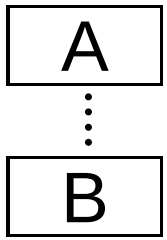
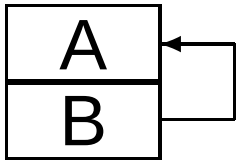
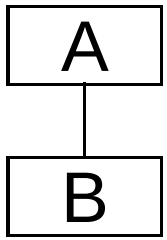
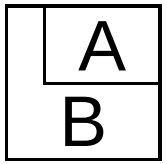


Repeated inheritance

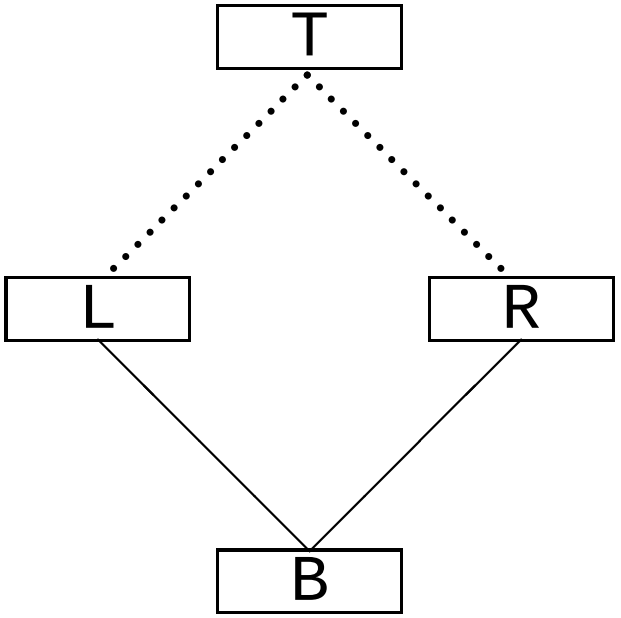
Motivation:

Modelling?
Efficiency?

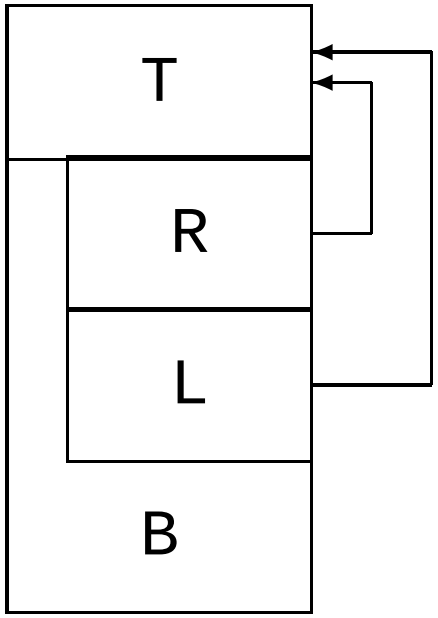
Multiple inheritance in C++

	C++ keyword	class diagram	object layout
shared:	`virtual`		
repeated:			

The shared diamond

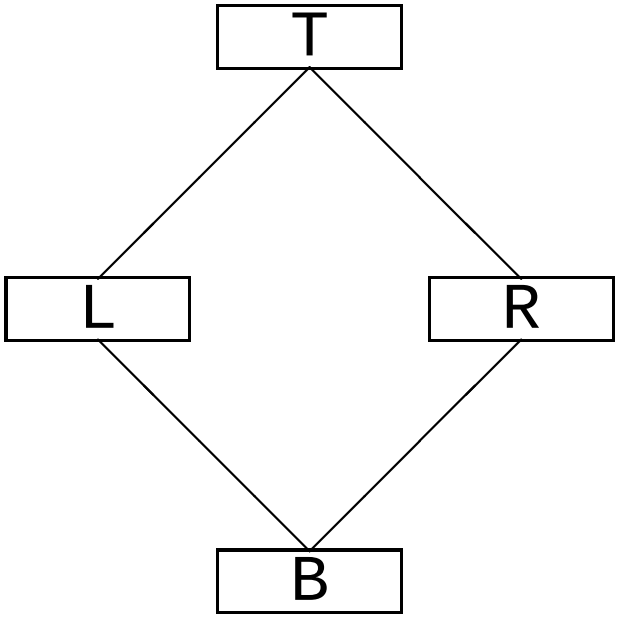


Class diagram

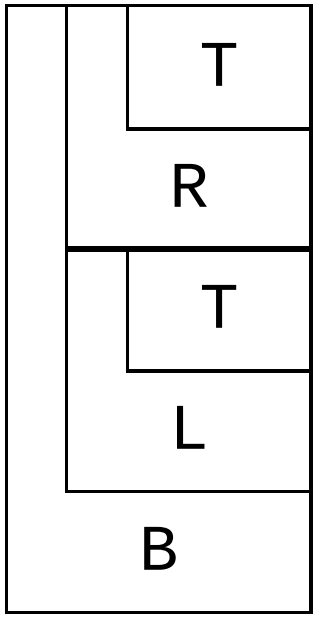


Object layout

The repeated diamond



Class diagram



Object layout

Alternative Java

Multiple inheritance only for *interfaces*

Outline

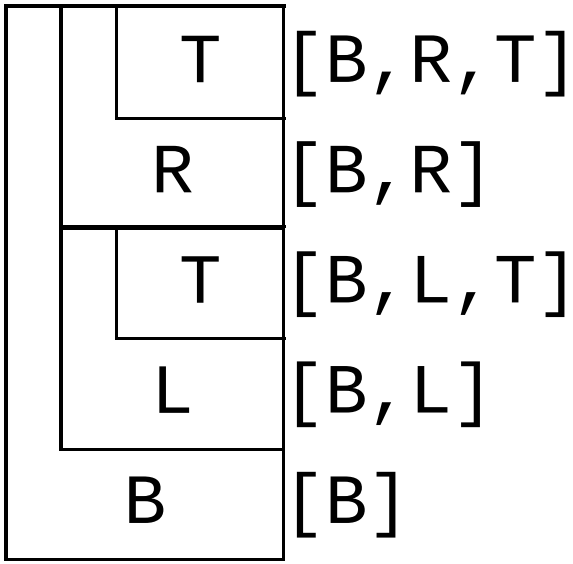
- Multiple inheritance in C++
- A formal model of *subobjects*
- Examples
- Semantics and type system

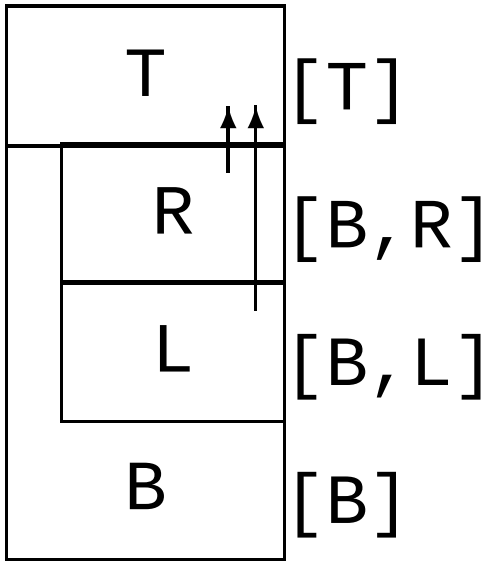
The Rossie-Friedman model of subobjects

©1995

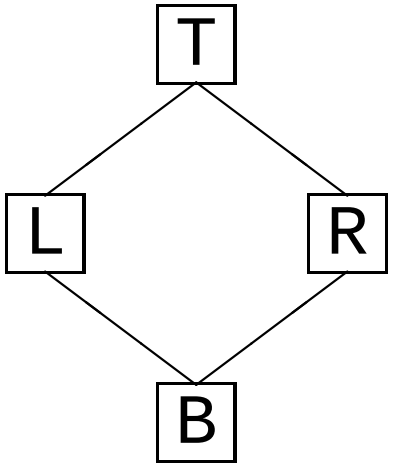
Paths

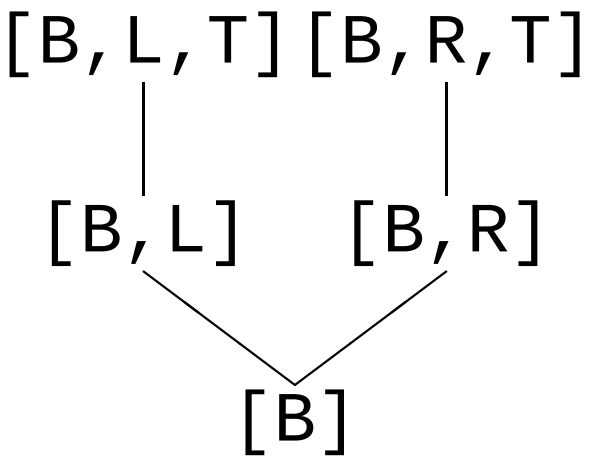
Identify (nested) subobjects by access path

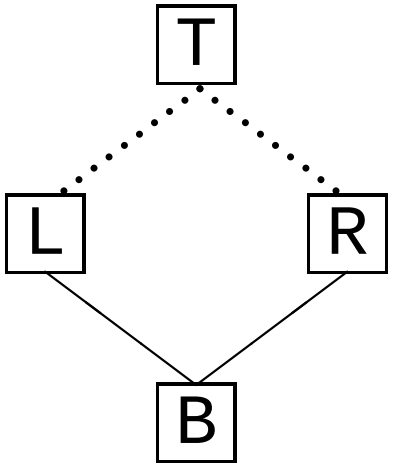


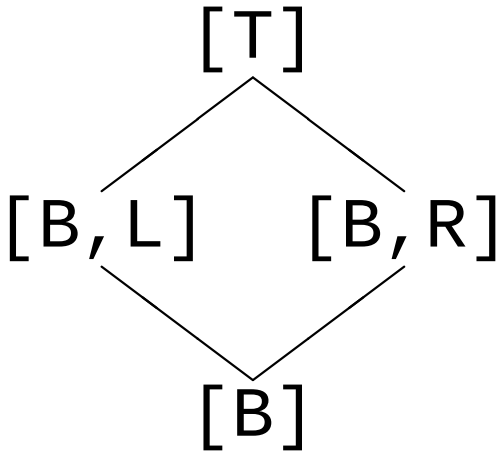


Unfolding repeated inheritance







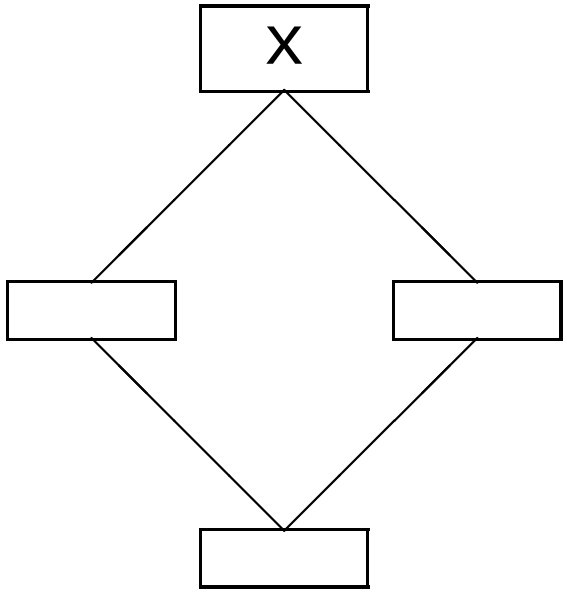


Paths are ordered

Outline

- Multiple inheritance in C++
- A formal model of *subobjects*
- Examples
- Semantics and type system

Ambiguous?



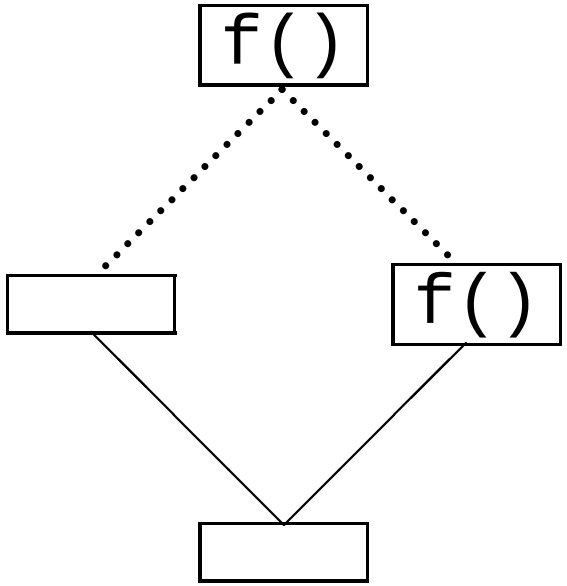

```
t:T;
t := (L) new B;
t.x
```

Cast adjusts pointer

```
l:L;
l := new B;
l.x
```

Assignment performs implicit cast

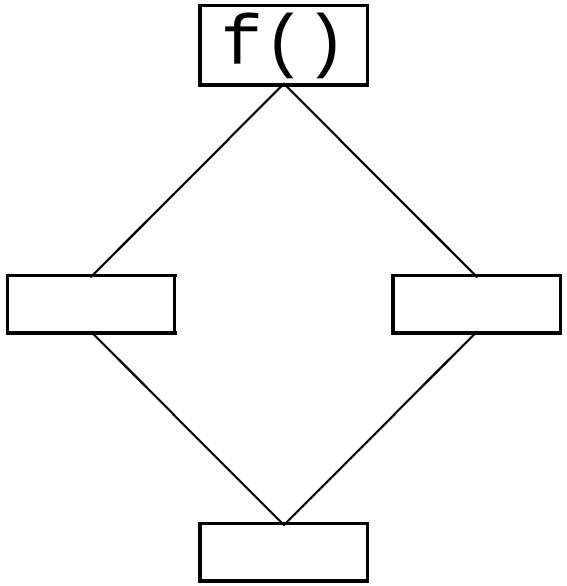
What is called?




```
l:L;
l := new B;
l.f()
```

[B,R] “dominates” [T]

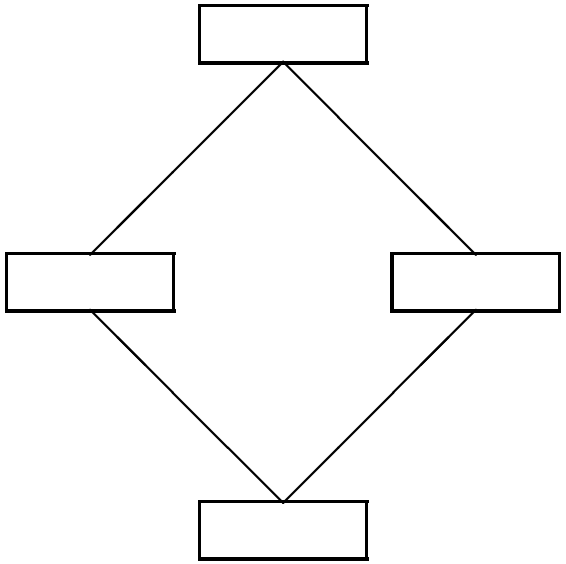
What is called?




```
l:L;
l := new B;
l.f()
```

Static type of object may disambiguate method call

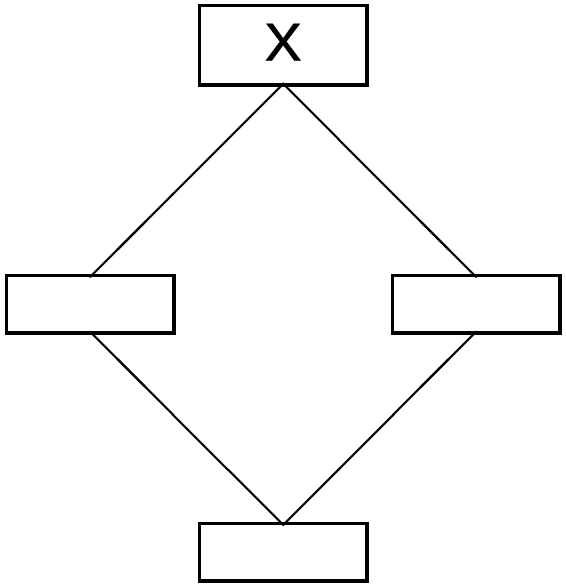
Legal?



```
b:B; b := new B;
l:L; l := b;
t:T; t := l;
(R)t
```

Statically allowed. But at run time ...

Legal?




```
class A: {f():Top = ...}
class B: A {f():Bot = ...}

a:A;
a := new B;
a.f().x
```

`f():Bot` is illegal.

Semantic domains

Paths

$[C_1,\ldots,C_n]$ is a *path* from C if

- $C_1 \prec_R \ldots \prec_R C_n$ and
- $C = C_1$ or $\exists\, C'.\ C \preceq^* C' \prec_S C_1$.

Objects

A B object in the shared diamond:

$$
\begin{array}{lll}
(B, & \{ & ([T], \quad [x \mapsto \ldots]), \\
 & & ([B,L], \quad \ldots), \\
 & & ([B,R], \quad \ldots), \\
 & & ([B], \quad \ldots) \\
 & \} & \\
) & &
\end{array}
$$

$$obj = cname \times subo\ set$$
$$subo = path \times (vname \rightharpoonup val)$$

Better: $obj = cname \times (path \times vname \rightharpoonup val)$

References

References must point to subobjects

Subobjects are identified by paths

$Addr\ a$ (Jinja) $\leadsto$ $Ref\ (a, Cs)$ (CoreC++)

$$ref\,(a,Cs) \equiv Val\,(Ref\,(a,Cs))$$

Intuition

If $e :: \mathit{Class}\ C$ and $e \Rightarrow \mathit{ref}\,(a,[C_1,\ldots,C_n])$
then $C = C_n$

Path functions

$P \vdash$ *path* C *to* D *via* $Cs \equiv P \vdash Cs$ *path from* $C \wedge$ *last* $Cs = D$

$P \vdash$ *path* C *to* D *unique* $\equiv$
$\exists!Cs.\ P \vdash Cs$ *path from* $C \wedge$ *last* $Cs = D$

$Cs \mathbin{@_p} Cs' \equiv$ *if* *last* $Cs =$ *hd* Cs' *then* $Cs \mathbin{@}$ *tl* Cs' *else* Cs'

Example:

In repeated diamond: $[B, L] \mathbin{@_p} [L, T] = [B, L, T]$
In shared diamond: $[B, L] \mathbin{@_p} [T] = [T]$

Semantics and type system

Assignment

Cast

Field access

Method call

Assignment: typing

$$\frac{E\ V = \lfloor T \rfloor \qquad P,E \vdash e :: T' \qquad P \vdash T' \leq T}{P,E \vdash V := e :: T}$$

where

$$\frac{P \vdash \mathit{path}\ C\ \mathit{to}\ D\ \mathit{unique}}{P \vdash \mathit{Class}\ C \leq \mathit{Class}\ D}$$

$$P \vdash T \leq T \qquad P \vdash NT \leq \mathit{Class}\ C$$

Assignment: semantics

$$\frac{P,E \vdash \langle e,s_0\rangle \Rightarrow \langle \mathtt{Val}\ v,(h,l)\rangle \qquad E\ V = \lfloor T \rfloor \qquad P \vdash T\ \textit{casts}\ v\ \textit{to}\ v' \qquad l' = l(V \mapsto v')}{P,E \vdash \langle V := e,s_0\rangle \Rightarrow \langle \mathtt{Val}\ v',(h,l')\rangle}$$

Examples:

$P \vdash \mathit{Integer}\ \mathit{casts}\ v\ \mathit{to}\ v$

$P \vdash \mathit{Class}\ T\ \mathit{casts}\ \mathit{Ref}\ (b,[B,R])\ \mathit{to}\ \mathit{Ref}\ (b,[B,R,T])$
(repeated diamond)

Assignment: casting

$$\frac{P \vdash \textit{path last } Cs \textit{ to } C \textit{ via } Cs'}{P \vdash \textit{Class } C \textit{ casts Ref } (a, Cs) \textit{ to Ref } (a, Cs \mathbin{@_p} Cs')}$$

$$\frac{\forall C.\ T \neq \textit{Class } C}{P \vdash T \textit{ casts } v \textit{ to } v}$$

$$P \vdash \textit{Class } C \textit{ casts Null to Null}$$

Semantics and type system

Assignment

Cast

Field access

Method call

Three casts!

- C-style cast - unsafe
- Static cast - unsafe
- Dynamic cast - safe but complicated

Dynamic cast: typing

$$\frac{P,E \vdash e :: \mathit{Class}\ D \qquad \mathit{is\text{-}class}\ P\ C}{P,E \vdash \texttt{dyn\_cast}\ C\ e :: \mathit{Class}\ C}$$

Why potentially ok even if neither $C \preceq^* D$ nor $D \preceq^* C$?

Dynamic up cast

Up cast extends path

	shared diamond	repeated diamond
$\texttt{dyn\_cast}\ T\ (\texttt{new}\ B)$	$\Rightarrow \mathit{ref}\ (b, [T])$	$\Rightarrow \mathit{null}$

$$\frac{\begin{array}{c} P,E \vdash \langle e,s_0\rangle \Rightarrow \langle \mathit{ref}\ (a, Cs),s_1\rangle \\ P \vdash \mathit{path}\ \mathit{last}\ Cs\ \mathit{to}\ C\ \mathit{unique} \\ P \vdash \mathit{path}\ \mathit{last}\ Cs\ \mathit{to}\ C\ \mathit{via}\ Cs' \end{array}}{P,E \vdash \langle \texttt{dyn\_cast}\ C\ e,s_0\rangle \Rightarrow \langle \mathit{ref}\ (a, Cs\ @_p\ Cs'),s_1\rangle}$$

Dynamic down cast (repeated)

Down cast shortens path

$$\texttt{dyn\_cast}\ R\ (\mathit{ref}\ (b, [B, R, T])) \Rightarrow \mathit{ref}\ (b, [B, R])$$

$$\frac{P,E \vdash \langle e,s_0\rangle \Rightarrow \langle \mathit{ref}\ (a, Cs\ @\ [C]\ @\ Cs'),s_1\rangle}{P,E \vdash \langle \texttt{dyn\_cast}\ C\ e,s_0\rangle \Rightarrow \langle \mathit{ref}\ (a, Cs\ @\ [C]),s_1\rangle}$$

Dynamic down cast (shared)

Wanted:

$\texttt{dyn\_cast}\ R\ (\mathit{ref}\ (b, [T])) \Rightarrow \mathit{ref}\ (b, [B, R])$

Need to consult dynamic class of object!

Dynamic cross cast

In an all-shared diamond:

$\texttt{dyn\_cast}\ R\ (\mathit{ref}\ (b, [L])) \Rightarrow \mathit{ref}\ (b, [R])$

Need to consult dynamic class of object!

Dynamic cast (general)

$$P,E \vdash \langle e,s_0 \rangle \Rightarrow \langle \mathit{ref}\ (a, Cs),(h, l) \rangle$$

$$h\ a = \lfloor (D, S) \rfloor$$

$$\frac{P \vdash \mathit{path}\ D\ \mathit{to}\ C\ \mathit{via}\ Cs' \qquad P \vdash \mathit{path}\ D\ \mathit{to}\ C\ \mathit{unique}}{P,E \vdash \langle \texttt{dyn\_cast}\ C\ e,s_0 \rangle \Rightarrow \langle \mathit{ref}\ (a, Cs'),(h, l) \rangle}$$

All 3 rules are required

$$\frac{P,E \vdash \langle e,s_0\rangle \Rightarrow \langle \mathit{ref}\ (a, Cs),(h, l)\rangle \qquad h\ a = \lfloor (D, S)\rfloor \qquad P \vdash \mathit{path}\ D\ \mathit{to}\ C\ \mathit{via}\ Cs' \qquad P \vdash \mathit{path}\ D\ \mathit{to}\ C\ \mathit{unique}}{P,E \vdash \langle \texttt{dyn\_cast}\ C\ e,s_0\rangle \Rightarrow \langle \mathit{ref}\ (a, Cs'),(h, l)\rangle}$$

$$\frac{P,E \vdash \langle e,s_0\rangle \Rightarrow \langle \mathit{ref}\ (a, Cs),s_1\rangle \qquad P \vdash \mathit{path}\ \mathit{last}\ Cs\ \mathit{to}\ C\ \mathit{unique} \qquad P \vdash \mathit{path}\ \mathit{last}\ Cs\ \mathit{to}\ C\ \mathit{via}\ Cs'}{P,E \vdash \langle \texttt{dyn\_cast}\ C\ e,s_0\rangle \Rightarrow \langle \mathit{ref}\ (a, Cs\ @_p\ Cs'),s_1\rangle}$$

$$\frac{P,E \vdash \langle e,s_0\rangle \Rightarrow \langle \mathit{ref}\ (a, Cs\ @\ [C]\ @\ Cs'),s_1\rangle}{P,E \vdash \langle \texttt{dyn\_cast}\ C\ e,s_0\rangle \Rightarrow \langle \mathit{ref}\ (a, Cs\ @\ [C]),s_1\rangle}$$

If all else fails

$$\frac{P,E \vdash \langle e,s_0 \rangle \Rightarrow \langle \mathit{ref}\ (a, Cs),(h, l) \rangle \qquad h\ a = \lfloor (D, S) \rfloor \qquad \neg\ P \vdash \mathit{path}\ D\ \mathit{to}\ C\ \mathit{unique} \qquad \neg\ P \vdash \mathit{path}\ \mathit{last}\ Cs\ \mathit{to}\ C\ \mathit{unique} \qquad C \notin \mathit{set}\ Cs}{P,E \vdash \langle \texttt{dyn\_cast}\ C\ e,s_0 \rangle \Rightarrow \langle \mathit{null},(h, l) \rangle}$$

Type system and semantics

Assignment

Cast

Field access

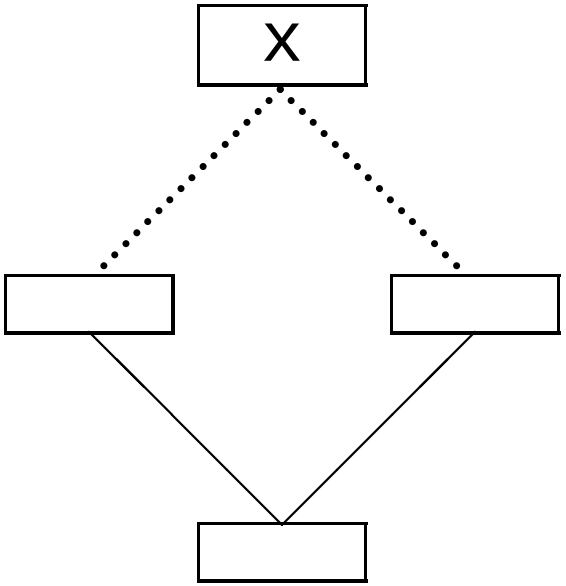
Method call

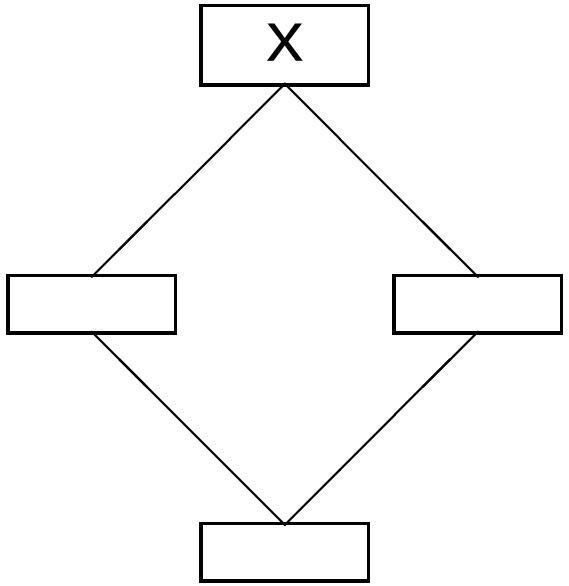
Field access: syntax

$$e.F\{Cs\}$$

Cs is path from static class of e to class declaring F

Typing examples





new $B.x\{[T]\}$ legal

new $B.x\{[\_]\}$ illegal

Field access: typing

$$\frac{P,E \vdash e :: Class\ C \qquad P \vdash C\ has\ least\ F : T\ via\ Cs}{P,E \vdash e.F\{Cs\} :: T}$$

- Path Cs leads from C to unique lowest declaration of F.

Field access: semantics

$$\frac{P,E \vdash \langle e,s_0 \rangle \Rightarrow \langle \mathit{ref}\ (a, Cs'),(h, l) \rangle \quad h\ a = \lfloor (D, S) \rfloor \quad (Cs' @_p Cs, fs) \in S \quad fs\ F = \lfloor v \rfloor}{P,E \vdash \langle e.F\{Cs\},s_0 \rangle \Rightarrow \langle \mathtt{Val}\ v,(h, l) \rangle}$$

Semantics and type system

Assignment

Cast

Field access

Method call

Method call: typing

$$\frac{P,E \vdash e :: Class\ C \qquad P \vdash C\ has\ least\ M = (Ts, T, \_, \_)\ via\ \_ \qquad P,E \vdash es\ [::]\ Ts' \qquad P \vdash Ts'\ [\leq]\ Ts}{P,E \vdash e.M(es) :: T}$$

- Must have unique lowest definition of M

Method call: semantics

$$\frac{\begin{array}{c} P,E \vdash \langle e,s_0\rangle \Rightarrow \langle \mathit{ref}\ (a, Cs),s_1\rangle \\ P,E \vdash \langle ps,s_1\rangle\ [\Rightarrow]\ \langle \mathit{map}\ \mathtt{Val}\ vs,(h_2, l_2)\rangle \\ h_2\ a = \lfloor (C, \_)\rfloor \\ P \vdash \mathit{last}\ Cs\ \mathit{has\ least}\ M = (Ts', T', pns', body')\ \mathit{via}\ Ds \\ P \vdash (C, Cs\ @_p\ Ds)\ \mathit{selects}\ M = (Ts, T, pns, body)\ \mathit{via}\ Cs' \\ \vdots \end{array}}{P,E \vdash \langle e.M(ps),s_0\rangle \Rightarrow \langle e',(h_3, l_2)\rangle}$$

Method selection

$$\frac{P \vdash C \text{ has least } M = mthd \text{ via } Cs'}{P \vdash (C, Cs) \text{ selects } M = mthd \text{ via } Cs'}$$

$$\frac{\forall mthd\ Cs'.\ \neg\, P \vdash C \text{ has least } M = mthd \text{ via } Cs' \qquad P \vdash (C, Cs) \text{ has overrider } M = mthd \text{ via } Cs'}{P \vdash (C, Cs) \text{ selects } M = mthd \text{ via } Cs'}$$

Method overriding: unique covariance

Wellformedness:

$$
\begin{array}{l}
\text{if} \quad M : Ts \to Class\ A \text{ in } C \\
\text{and } M : Us \to Class\ B \text{ in } D \\
\text{and } P \vdash D \preceq^* C \\
\text{then } P \vdash \textit{path } B \textit{ to } A \textit{ unique}
\end{array}
$$

Type Safety

CoreC++ is type safe

Proof similar to Jinja