An Overview of Aspectualized Component Models

Hakim Hannousse

ECOLE DES MINES DE NANTES
DEPARTMENT OF COMPUTER SCIENCE

Outline

- Aspectualized Component Models
  - FuseJ Component Model [2 Papers]
  - CaeserJ [2 Paper]

FuseJ Component Model : Concepts & Features

- Aspects are ordinary Components
- FuseJ introduces the concept of Service specification
  - provided services
  - expected services
- Gates are used to access to component services : Incoming gates, Outgoing gates, ordinary gates
- The interaction between gates is made using connectors : regular and aspect-oriented interactions

FuseJ Component Model : Language - Gates (1)

```
interface BookingService for BookingServiceComponent {

  gate BookHotel {
    binds:
      Float bookHotel(String hotelname);
    exposes:
      String inputHotelName = hotelname;
      Float outputPrice = returnvalue;
  }

  outputgate ChargeForHotel {
    binds:
      void fireChargeRequest(ChargeEvent event);
    exposes:
      ChargeEvent chargeEvent = event;
  }

}
```

FuseJ Component Model : Language - Gates (2)

```
interface PaymentService for PaymentServiceComponent {

  gate ChargeAmount {
    binds:
      void chargeAmount(String ccnumber, Float amount);
    exposes:
      String inputCCNumber = ccnumber;
      Float inputAmount = amount;
  }

  gate ReserveAmount {
    binds:
      void reserveAmount(String ccnumber, Float amount);
    exposes:
      String inputCCNumber = ccnumber;
      Float inputAmount = amount;
  }

  outputgate BillingActions {
    binds:
      void *Amount(String ccnumber, Float amount);
    exposes:
      String inputCCNumber = ccnumber;
      Float inputAmount = amount;
  }

}
```

FuseJ Component Model : Language - Gates (3)

```
interface DiscountService for DiscountServiceComponent {

  inputgate Discount {
    binds:
      Float getDiscountPrice(Float price, Float percent);
    exposes:
      Float inputPrice = price;
      Float discountPercentage = percent;
      Float outputPrice = returnvalue;
  }

}
```

FuseJ Component Model : Language - Regular Interaction Connectors

```
connector BookingPayment {

  execute:
    PaymentService.ChargeAmount;
  for:
    BookingService.ChargeForHotel;
  where:
    PaymentService.ChargeAmount.inputCCNumber =
      BookingService.ChargeForHotel.chargeEvent.visaNumber;
    PaymentService.ChargeAmount.inputAmount =
      BookingService.ChargeForHotel.chargeEvent.amount;

}
```

FuseJ Component Model : Language - Aspect-Oriented Interaction Connectors

```
connector BookingDiscount {

  execute:
    DiscountService.Discount;
  around:
    BookingService.BookHotel;
  where:
    DiscountService.Discount.inputPrice =
      BookingService.BookHotel.outputPrice;
    DiscountService.Discount.discountPercentage = 15;
  when:
    DateService.ChristmasHolidayDate.outputValue;

}
```

FuseJ Component Model : Implementation Model

FuseJ introduces two implementation mechanisms :

1. Gate Interface Preprocessor : Translate each gate to a Java 1.5 annotations and bind these annotations to the corresponding components at compile time
2. Execution Environment : Generates for each component a container based on the Java annotations and generate a JavaBean for each connector, which is the responsible to manage the interaction between the involved gates.

FuseJ Component Model : Evaluation

1. Comprehensibility : Aspects and Components are both from the same dimension
2. Evolvability: Connectors can be attached and detached at run-tim, aspects and components can be evolved independently.
3. Semantic interaction: No semantic interaction between components are supported by FuseJ
4. Predictability

CaesarJ Component Model: Features

1. Supports a large-sclae units of modularity by introducing the concept of a class family
2. Supports virtual classes
3. introduces Family polymorphism
4. Provides a hierarchical composition mechanism : Mixin Composition
5. Supports abstract classes and collaboration interfaces
6. Supports binding (i.e. Wraper) : A class family which implements a facet by adapting external classes
7. Aspects are components inheriting their pointcuts and advices from its component binding.
8. Aspects can be statically or dynamically deployed, it can be also Remotely deployed using Caesar RMI API extension to Java

CaesarJ Component Model: Syntax

Grammar of *vc*

CL	::=	**class** C **extends** $\overline{\mathsf{C}}$ { K $\overline{\mathsf{CL}}$; $\overline{\mathsf{T}}$ $\overline{\mathsf{f}}$; $\overline{\mathsf{T}}$ $\overline{\mathsf{v}}$ }
K	::=	T C($\overline{\mathsf{T}}$ $\overline{\mathsf{f}}$) { e; }
T	::=	path.C
path	::=	spine.$\overline{\mathsf{f}}$
spine	::=	**this**.$\overline{\textbf{out}}$
e	::=	**null** | e ; e | path | path.v | path.v = e | **new** path.C($\overline{\mathsf{e}}$)

Identifiers

class names	C	
field names	f	
variable names	v	
members	m	= f ∪ v

(C, f, and v are pairwise disjoint)

CaesarJ Component Model: Definitions

Metavariable

$$\text{static paths} \quad \mathsf{p} \quad ::= \quad \overline{\mathsf{C}}$$

Class table

$$CT(\mathsf{p}) = CT2(\mathsf{p}, \overline{\mathsf{CL}}_{root})$$

$$\frac{\mathsf{CL}_i = \textbf{class } \mathsf{C} \textbf{ extends } \overline{\mathsf{C}} \{ \ldots \}}{CT2(\mathsf{C}, \overline{\mathsf{CL}}) = \mathsf{CL}_i}$$

$$\frac{\mathsf{CL}_i = \textbf{class } \mathsf{C} \textbf{ extends } \overline{\mathsf{C}} \{ \mathsf{K}\ \overline{\mathsf{CL}}';\ \ldots \}}{CT2(\mathsf{C.p}, \overline{\mathsf{CL}}) = CT(\mathsf{p}, \overline{\mathsf{CL}}')}$$

All members

$$\mathcal{M}embers(\mathsf{nil_p}) = \mathsf{nil}_{\mathsf{T\,f}}, \mathsf{nil}_{\mathsf{T\,v}}$$

$$\frac{\mathcal{M}embers(\overline{\mathsf{p}}) = \overline{\mathsf{T}}\ \overline{\mathsf{f}}, \overline{\mathsf{T}}'\ \overline{\mathsf{v}} \qquad CT(\mathsf{p}) = \textbf{class } \mathsf{C} \textbf{ extends } \overline{\mathsf{C}} \{ \mathsf{K}\ \overline{\mathsf{CL}};\ \overline{\mathsf{T}}''\ \overline{\mathsf{f}}';\overline{\mathsf{T}}'''\ \overline{\mathsf{v}}' \}}{\mathcal{M}embers(\mathsf{p}\ \overline{\mathsf{p}}) = \overline{\mathsf{T}}''\ \overline{\mathsf{f}}'\ \overline{\mathsf{T}}\ \overline{\mathsf{f}}, \overline{\mathsf{T}}'''\ \overline{\mathsf{v}}'\ \overline{\mathsf{T}}'\ \overline{\mathsf{v}}}$$

Constructor

$$\frac{CT(\mathsf{p}) = \textbf{class } \mathsf{C} \textbf{ extends } \overline{\mathsf{C}} \{ \mathsf{K}\ \overline{\mathsf{CL}};\ \overline{\mathsf{T}}''\ \overline{\mathsf{f}}';\overline{\mathsf{T}}'''\ \overline{\mathsf{v}}' \}}{\mathcal{C}onstr(\mathsf{p}) = \mathsf{K}}$$

CaesarJ Component Model: Semantics (1)

Objects and the Heap:

$$\begin{array}{ll}
\textbf{Address} = \text{natural numbers} & \iota \\
\textbf{Object} = \{ [\![\ \iota \parallel \mathsf{C} \parallel \overline{\mathsf{f}} : \overline{\mathsf{val}} \quad \overline{\mathsf{v}} : \overline{\mathsf{val}}'\ ]\!] \} & [\![\ \ldots\ ]\!] \\
\textbf{Heap} = \textbf{Address} \xrightarrow{\text{fin}} \textbf{Object} & \mathsf{H} \\
\textbf{Value} = \textbf{Address} \cup \{\textbf{null}\} & \mathsf{val}
\end{array}$$

Evaluation rules:

$\leadsto$: $\mathsf{e} \times \textbf{Heap} \times \textbf{Address} \rightarrow \textbf{Value} \cup \{\mathsf{TypeErr}, \mathsf{NullErr}\} \times \textbf{Heap}$

$$\textbf{null}, \mathsf{H}, \iota \leadsto \textbf{null}, \mathsf{H} \quad \text{(R1)}$$

$$\frac{\begin{array}{c}\mathsf{e}, \mathsf{H}, \iota \leadsto \mathsf{val}, \mathsf{H}' \\ \mathsf{e}', \mathsf{H}', \iota \leadsto \mathsf{val}', \mathsf{H}''\end{array}}{\mathsf{e}\,;\mathsf{e}', \mathsf{H}, \iota \leadsto \mathsf{val}', \mathsf{H}''} \quad \text{(R2)}$$

$$\frac{\mathit{Walk}(\mathsf{H}, \iota, \mathsf{path}) = \mathsf{val}}{\mathsf{path}, \mathsf{H}, \iota \leadsto \mathsf{val}, \mathsf{H}} \quad \text{(R3)}$$

$$\frac{\begin{array}{c}\mathsf{path}, \mathsf{H}, \iota \leadsto \iota', \mathsf{H} \\ \mathsf{H}(\iota')(\mathsf{v}) = \mathsf{val}\end{array}}{\mathsf{path.v}, \mathsf{H}, \iota \leadsto \mathsf{val}, \mathsf{H}} \quad \text{(R4)}$$

$$\frac{\begin{array}{c}\mathsf{path}, \mathsf{H}, \iota \leadsto \iota', \mathsf{H} \qquad \mathsf{e}, \mathsf{H}, \iota \leadsto \mathsf{val}, \mathsf{H}' \\ \mathsf{H}'(\iota')(\mathsf{v}) \neq \perp \qquad \mathsf{H}'' = \mathsf{H}'[\iota' \mapsto \mathsf{H}'(\iota')[\mathsf{v} \mapsto \mathsf{val}]]\end{array}}{\mathsf{path.v} = \mathsf{e}, \mathsf{H}, \iota \leadsto \mathsf{val}, \mathsf{H}''} \quad \text{(R5)}$$

$$\frac{\begin{array}{c}\mathsf{path}, \mathsf{H}, \iota \leadsto \iota', \mathsf{H} \qquad \mathsf{H} = \mathsf{H}_1 \\ \mathsf{e}_i, \mathsf{H}_i, \iota \leadsto \mathsf{val}_i, \mathsf{H}_{i+1} \ \text{ for } i \in \{1 \ldots |\overline{\mathsf{e}}|\} \\ \mathsf{H}' = \mathsf{H}_{|\overline{\mathsf{e}}|+1} \qquad \overline{\mathsf{p}} = \mathit{Assemble}(\mathit{Mix}(\mathsf{H}', \iota'), \mathsf{C}) \\ \mathit{Members}(\overline{\mathsf{p}}) = \overline{\mathsf{T}}\,\overline{\mathsf{f}}, \overline{\mathsf{T}}'\,\overline{\mathsf{v}} \qquad |\overline{\mathsf{f}}| = |\overline{\mathsf{val}}| \\ \iota'' \text{ is new in } \mathsf{H}' \qquad \mathit{Constr}(\mathsf{p}_{|\overline{\mathsf{p}}|}) = \mathsf{T}\ \mathsf{C}(\_)\{\mathsf{e}';\} \\ \mathsf{H}'' = \mathsf{H}'[\ \iota'' \mapsto [\![\ \iota' \parallel \mathsf{C} \parallel \overline{\mathsf{f}} : \overline{\mathsf{val}} \quad \overline{\mathsf{v}} : \overline{\textbf{null}}\ ]\!]\ ] \\ \mathsf{e}', \mathsf{H}'', \iota'' \leadsto \mathsf{val}, \mathsf{H}'''\end{array}}{\textbf{new}\ \mathsf{path.C}(\overline{\mathsf{e}}), \mathsf{H}, \iota \leadsto \mathsf{val}, \mathsf{H}'''} \quad \text{(R6)}$$

Enclosing object:

$\mathcal{E}ncl([\![\ \iota \parallel \_ \parallel \ldots\ ]\!]) = \iota$

Evaluation functions:

$$\begin{array}{ll}
\mathit{Walk}(\mathsf{H}, \iota, \textbf{this}) = \iota & \\
\mathit{Walk}(\mathsf{H}, \iota, \mathsf{spine}.\textbf{out}) = \mathcal{E}ncl(\mathsf{H}(\iota')) & \text{if } \mathit{Walk}(\mathsf{H}, \iota, \mathsf{spine}) = \iota' \neq \iota_{root} \\
\mathit{Walk}(\mathsf{H}, \iota, \mathsf{path.f}) = \mathsf{val} & \text{if } \mathsf{H}(\mathit{Walk}(\mathsf{H}, \iota, \mathsf{path}))(\mathsf{f}) = \mathsf{val} \\
\mathit{Walk}(\mathsf{H}, \iota, \mathsf{path.f}) = \mathsf{NullErr} & \text{if } \mathit{Walk}(\mathsf{H}, \iota, \mathsf{path}) = \textbf{null} \\
\mathit{Walk}(\mathsf{H}, \iota, \mathsf{path.f}) = \mathsf{TypeErr} & \text{if } \mathsf{H}(\mathit{Walk}(\mathsf{H}, \iota, \mathsf{path}))(\mathsf{f}) = \perp \\
\mathit{Walk}(\mathsf{H}, \iota, \mathsf{spine}.\textbf{out}) = \mathsf{TypeErr} & \text{if } \mathit{Walk}(\mathsf{H}, \iota, \mathsf{spine}) = \iota_{root}
\end{array}$$

Error handling:

$$\frac{\mathsf{path}, \mathsf{H}, \iota \leadsto \textbf{null}, \mathsf{H}}{\begin{array}{c}\mathsf{path.v}, \mathsf{H}, \iota \leadsto \mathsf{NullErr}, \mathsf{H} \\ \mathsf{path.v} = \mathsf{e}, \mathsf{H}, \iota \leadsto \mathsf{NullErr}, \mathsf{H} \\ \textbf{new}\ \mathsf{path.C}(\overline{\mathsf{e}}), \mathsf{H}, \iota \leadsto \mathsf{NullErr}, \mathsf{H}\end{array}} \quad \text{(ER1)}$$

$$\frac{\mathsf{path}, \mathsf{H}, \iota \leadsto \iota', \mathsf{H} \qquad \mathsf{H}(\iota')(\mathsf{v}) = \perp}{\begin{array}{c}\mathsf{path.v}, \mathsf{H}, \iota \leadsto \mathsf{TypeErr}, \mathsf{H} \\ \mathsf{path.v} = \mathsf{e}, \mathsf{H}, \iota \leadsto \mathsf{TypeErr}, \mathsf{H}\end{array}} \quad \text{(ER2)}$$

$$\frac{\begin{array}{c}\mathsf{path}, \mathsf{H}, \iota \leadsto \iota', \mathsf{H} \\ \mathit{Assemble}(\mathit{Mix}(\mathsf{H}, \iota'), \mathsf{C}) = \perp\end{array}}{\textbf{new}\ \mathsf{path.C}(\overline{\mathsf{e}}), \mathsf{H}, \iota \leadsto \mathsf{TypeErr}, \mathsf{H}} \quad \text{(ER3)}$$

$$\frac{\begin{array}{c}\mathsf{path}, \mathsf{H}, \iota \leadsto \iota', \mathsf{H} \\ \mathit{Assemble}(\mathit{Mix}(\mathsf{H}, \iota'), \mathsf{C}) = \overline{\mathsf{p}} \\ \mathit{Members}(\overline{\mathsf{p}}) = \overline{\mathsf{T}}\,\overline{\mathsf{f}}, \_ \qquad |\overline{\mathsf{e}}| \neq |\overline{\mathsf{f}}|\end{array}}{\textbf{new}\ \mathsf{path.C}(\overline{\mathsf{e}}), \mathsf{H}, \iota \leadsto \mathsf{TypeErr}, \mathsf{H}} \quad \text{(ER4)}$$

Mixin Computation:

$$\begin{array}{l}
\mathit{Mix}(\mathsf{H}, \iota_{root}) = [\mathsf{nil_c}] \\
\mathit{Mix}(\mathsf{H}, \iota) = \mathit{Assemble}(\mathit{Mix}(\mathsf{H}, \iota'), \mathsf{C}) \\
\quad \text{where } \mathsf{H}(\iota) = [\![\ \iota' \parallel \mathsf{C} \parallel \ldots\ ]\!]
\end{array}$$

$$\mathit{Assemble}(\overline{\mathsf{p}}, \mathsf{C}) = \mathit{Linearize}[\,\mathit{Expand}(\overline{\mathsf{p}}, \mathsf{p}) \mid \mathsf{p} \leftarrow \mathit{Defs}(\overline{\mathsf{p}}, \mathsf{C})\,]$$

$$\begin{array}{l}
\mathit{Defs}(\overline{\mathsf{p}}, \mathsf{C}) = \mathit{check}[\,\mathsf{p.C} \mid \mathsf{p} \leftarrow \overline{\mathsf{p}}, \mathit{CT}(\mathsf{p.C}) \neq \perp\,] \\
\quad \text{where } \ \mathit{check}(\overline{\mathsf{p}}) = \begin{cases} \perp & |\overline{\mathsf{p}}| = 0 \\ \overline{\mathsf{p}} & \mathit{otherwise} \end{cases}
\end{array}$$

$$\begin{array}{l}
\mathit{Expand}(\overline{\mathsf{p}}, \mathsf{p}) = \mathit{Linearize}([\,\mathit{Assemble}(\overline{\mathsf{p}}, \mathsf{C}) \mid \mathsf{C} \leftarrow \overline{\mathsf{C}}\,]\ \mathsf{p}) \\
\quad \text{where } \mathit{CT}(\mathsf{p}) = \textbf{class}\ \mathsf{C}'\ \textbf{extends}\ \overline{\mathsf{C}}\ \{\ \ldots\ \}
\end{array}$$

$$\begin{array}{lcl}
\mathit{Linearize}(\mathsf{nil}_{\overline{\mathsf{p}}}) = \mathsf{nil_p} & & \\
\mathit{Linearize}(\overline{\overline{\mathsf{p}}}\ \overline{\mathsf{p}}) = \mathit{Lin2}(\mathit{Linearize}(\overline{\overline{\mathsf{p}}}), \overline{\mathsf{p}}) & & \\
\mathit{Lin2}(\mathsf{nil_p}, \mathsf{nil_p}) & = & \mathsf{nil_p} \\
\mathit{Lin2}(\overline{\mathsf{p}}\ \mathsf{p}, \overline{\mathsf{p}}'\ \mathsf{p}) & = & \mathit{Lin2}(\overline{\mathsf{p}}, \overline{\mathsf{p}}')\ \mathsf{p} \\
\mathit{Lin2}(\overline{\mathsf{p}}, \overline{\mathsf{p}}'\ \mathsf{p}') & = & \mathit{Lin2}(\overline{\mathsf{p}}, \overline{\mathsf{p}}')\ \mathsf{p}', \ \text{if } \mathsf{p}' \notin \overline{\mathsf{p}} \\
\mathit{Lin2}(\overline{\mathsf{p}}\ \mathsf{p}, \overline{\mathsf{p}}') & = & \mathit{Lin2}(\overline{\mathsf{p}}, \overline{\mathsf{p}}')\ \mathsf{p}, \ \text{if } \mathsf{p} \notin \overline{\mathsf{p}}' \\
\mathit{Lin2}(\overline{\mathsf{p}}\,\mathsf{p}'\,\overline{\mathsf{p}}''\,\mathsf{p}, \overline{\mathsf{p}}'\,\mathsf{p}') & = & \mathit{Lin2}(\overline{\mathsf{p}}\,\overline{\mathsf{p}}''\,\mathsf{p}, \overline{\mathsf{p}}')\ \mathsf{p}'
\end{array}$$

CaesarJ Component Model: Semantics (2)

Typing domains:

$$\begin{array}{lll} u & ::= & \langle p\rangle.\overline{f} \\ s & ::= & \langle p\rangle.\overline{f}.C \\ t & ::= & u \mid s \end{array} \qquad \begin{array}{lll} q & ::= & \mathbf{this} \mid \mathbf{out} \mid f \\ Q & ::= & \overline{q} \mid \overline{q}.C \end{array}$$

Expression Typing:

$$\frac{\mathcal{M}(t) \neq \bot}{p \vdash \mathbf{null} : t} \ \text{(T1)} \qquad \frac{\mathcal{W}(\langle p\rangle, path) = u}{p \vdash path : u} \ \text{(T3)}$$

$$\frac{p \vdash e : t \quad p \vdash e' : t'}{p \vdash e; e' : t'} \ \text{(T2)} \qquad \frac{p \vdash path : u \quad \mathcal{W}(u, \mathit{DclType}(u,v)) = s}{p \vdash path.v : s} \ \text{(T4)}$$

$$\frac{p \vdash path.v : s \qquad p \vdash e : t \qquad \mathcal{C}(t) <: s}{p \vdash path.v = e : t} \ \text{(T5)}$$

$$\frac{\begin{array}{c} p \vdash path : u \quad p' \in \mathcal{M}(u.C) \quad p \vdash \overline{e} : \overline{t} \\ \mathit{Constr}(p') = T_0\ C(\overline{T}\,\overline{f}) \ldots \quad |\overline{T}| = |\overline{t}| \\ s_i = \begin{cases} \mathcal{W}(u, \mathbf{this}.Q) & \text{if } T_i = \mathbf{this}.\mathbf{out}.Q \\ \mathcal{W}(u_j, \mathbf{this}.Q) & \text{if } T_i = \mathbf{this}.f_j.Q \wedge t_j = u_j \end{cases} \\ \text{for } i = 0 \ldots |\overline{t}| \\ \mathcal{C}(t_i) <: s_i \text{ for } i = 1 \ldots |\overline{t}| \end{array}}{p \vdash \mathbf{new}\ path.C(\overline{e}) : s_0} \ \text{(T6)}$$

Conversion to class types:

$$\begin{array}{lll} \mathcal{C} & :: & t \to s \\ \mathcal{C}(\langle p.C\rangle) & = & \langle p\rangle.C \\ \mathcal{C}(u.f) & = & \mathcal{W}(u, \mathit{DclType}(u,f)) \\ \mathcal{C}(s) & = & s \end{array}$$

Mixins:

$$\begin{array}{lll} \mathcal{M} & :: & t \to \overline{p} \\ \mathcal{M}(\langle\rangle) & = & [nil_c] \\ \mathcal{M}(u.C) & = & \mathit{Assemble}(\mathcal{M}(u), C) \\ \mathcal{M}(u) & = & \mathcal{M}(\mathcal{C}(u)) \end{array}$$

Enclosing object type:

$$\begin{array}{ll} \mathcal{E} & :: t \to u \\ \mathcal{E}(u.C) & = u \\ \mathcal{E}(u) & = \mathcal{E}(\mathcal{C}(u)) \end{array}$$

Static lookup:

$$\begin{array}{ll} \mathcal{W} & :: u \times (path \cup T) \to t \\ \mathcal{W}(u, \mathbf{this}) & = u \\ \mathcal{W}(u, spine.\mathbf{out}) & = \mathcal{E}(\mathcal{W}(u, spine)) \\ \mathcal{W}(u, path.f) & = \mathcal{W}(u, path).f \ \text{ if } \mathit{Exists}(\mathcal{W}(u, path), f) \\ \mathcal{W}(u, path.C) & = \mathcal{W}(u, path).C \ \text{ if } \mathit{Exists}(\mathcal{W}(u, path), C) \end{array}$$

Program Typing:

$$\frac{\mathcal{M}(\langle p\rangle.C) \neq \bot}{p \vdash C\ \mathrm{OK}} \ \text{(WF1)} \qquad \frac{\mathcal{W}(\langle p\rangle, T) \neq \bot}{p \vdash T\ \mathrm{OK}} \ \text{(WF2)}$$

$$\frac{C = C' \Rightarrow T = T', \overline{T}\,\overline{f} = \overline{T}'\,\overline{f}'}{T\ C(\overline{T}\,\overline{f})\ \{e;\}\ \text{overrides}\ T'\ C'(\overline{T}'\,\overline{f}')\ \{e';\}\ \mathrm{OK}} \ \text{(WF3)}$$

$$\frac{\begin{array}{c} K = T\ C(\overline{T}''\,\overline{f}')\ \{e;\} \qquad \mathcal{M}(\langle p\rangle.C) = \overline{p} \\ \mathit{Members}(\overline{p}) = \overline{T}''\,\overline{f}', \_ \\ p \vdash \overline{C}\ \mathrm{OK} \quad p.C \vdash \overline{T}\ \mathrm{OK} \quad p.C \vdash \overline{T}'\ \mathrm{OK} \quad p.C \vdash T\ \mathrm{OK} \\ p.C \vdash e : t \qquad \mathcal{C}(t) <: \mathcal{W}(\langle p.C\rangle, T) \\ K' = \mathit{Constr}(p_j) \Rightarrow K\ \text{overrides}\ K'\ \mathrm{OK} \end{array}}{p \vdash \mathbf{class}\ C\ \mathbf{extends}\ \overline{C}\ \{K\ \overline{CL};\ \overline{T}\,\overline{f};\ \overline{T}'\,\overline{v}\}\ \mathrm{OK}} \ \text{(WF4)}$$

$$\frac{\begin{array}{c} \text{There is a strict partial order } \sqsubset_f \text{ on field names such that} \\ \forall p, f. \quad spine.\overline{f}.C\ f \in \mathit{Members}(p) \Rightarrow \forall i.\ f_i \sqsubset_f f \\ \text{There is a strict partial order } \sqsubset_c \text{ on class names such that} \\ \forall p.\ \mathit{CT}(p) = \mathbf{class}\ C\ \mathbf{extends}\ \overline{C}\ \{\ldots\} \Rightarrow \forall i.\ C_i \sqsubset_c C \end{array}}{\mathit{CT} \text{ is acyclic}} \ \text{(WF5)}$$

$$\frac{\begin{array}{c} \mathit{CT} \text{ is acyclic} \\ \forall p, p', C : \mathit{CT}(p.C) \neq \bot, \mathit{CT}(p'.C) \neq \bot \Rightarrow \\ p''.C \in \mathcal{M}(\langle p\rangle.C) \cap \mathcal{M}(\langle p'\rangle.C) \\ \forall p \neq p' : \mathit{CT}(p) = \mathbf{class}\ C \ldots \{K\ \overline{CL};\ \overline{T}\,\overline{f};\ \overline{T}'\,\overline{v}\} \\ \mathit{CT}(p') = \mathbf{class}\ C' \ldots \{K'\ \overline{CL}';\ \overline{T}''\,\overline{f}';\ \overline{T}'''\,\overline{v}'\} \\ \Rightarrow \overline{f} \cap \overline{f}' = \emptyset, \overline{v} \cap \overline{v}' = \emptyset \\ \forall p, C : \mathit{CT}(p.C) \neq \bot \Rightarrow p \vdash \mathit{CT}(p.C)\ \mathrm{OK} \end{array}}{\mathit{CT}\ \mathrm{OK}} \ \text{(WF6)}$$

Subtyping:

$$s <: s \ \text{(S-Refl)} \qquad \frac{s <: s' \quad s' <: s''}{s <: s''} \ \text{(S-Trans)}$$

$$\frac{\mathcal{M}(u) = \overline{p} \qquad \mathit{CT}(p_j.C) = \mathbf{class}\ C\ \mathbf{extends} \ldots C' \ldots}{u.C <: u.C'} \ \text{(S-Decl)}$$

Declared type of member:

$$\begin{array}{lll} \mathit{DclType}(t, m) & = & T \quad \text{where } T\ m \in \mathit{Members}(\mathcal{M}(t)) \\ \mathit{Exists}(t, m) & = & (\mathit{DclType}(t, m) \neq \bot) \\ \mathit{Exists}(u, C) & = & (\mathcal{M}(u.C) \neq \bot) \end{array}$$

CaesarJ Component Model: Semantics (3)

Well-formedness:

$$\frac{\mathsf{H}(\iota)(\mathsf{m}) = \textbf{null}}{\iota.\mathsf{m} : \mathsf{T}\ \ \mathsf{OK\ in\ H}} \qquad \text{(WF-NULL)}$$

$$\frac{\begin{array}{c}\mathsf{H}(\iota)(\mathsf{m}) = \iota' \\ \mathcal{Walk}(\mathsf{H}, \iota', \textbf{out}) = \mathcal{Walk}(\mathsf{H}, \iota, \mathsf{path}) \quad \mathsf{p.C} \in \mathcal{Mix}(\mathsf{H}, \iota')\end{array}}{\iota.\mathsf{m} : \mathsf{path.C}\ \ \mathsf{OK\ in\ H}} \qquad \text{(WF-MEM)}$$

$$\frac{\mathsf{T\ m} \in \mathit{Members}(\mathcal{Mix}(\mathsf{H}, \iota)) \Rightarrow \iota.\mathsf{m} : \mathsf{T}\ \ \mathsf{OK\ in\ H}}{\iota\ \ \mathsf{OK\ in\ H}} \qquad \text{(WF-OBJ)}$$

$$\frac{\mathsf{H}(\iota_{root}) = [\![\ \bot \ \|\ \mathsf{C}_{root}\ \|\ \ ]\!]}{\iota_{root}\ \ \mathsf{OK\ in\ H}} \qquad \text{(WF-ROOT)}$$

$$\frac{\forall \iota.\ \iota\ \ \mathsf{OK\ in\ H}}{\mathsf{H\ OK}} \qquad \text{(WF-HEAP)}$$

Agreement:

$$\mathsf{H}, \iota_0 \vdash \textbf{null} \triangleright \mathsf{val}_s \qquad \text{(A-NULL)}$$

$$\mathsf{H}, \iota_0 \vdash \iota_{root} \triangleright \langle\rangle \qquad \text{(A-ROOT)}$$

$$\frac{\begin{array}{c} j = \mathcal{Depth}(\mathsf{H}, \iota_0) - |\mathsf{p}| \\ \mathcal{Walk}(\mathsf{H}, \iota_0, \textbf{this}.\textbf{out}^j.\overline{\mathsf{f}}) = \iota \quad \mathsf{H}, \iota_0 \vdash \iota \triangleright \mathcal{C}(\langle \mathsf{p} \rangle.\overline{\mathsf{f}})\end{array}}{\mathsf{H}, \iota_0 \vdash \iota \triangleright \langle \mathsf{p} \rangle.\overline{\mathsf{f}}} \qquad \text{(A-OTYPE)}$$

$$\frac{\mathsf{p}'.\mathsf{C} \in \mathcal{Mix}(\mathsf{H}, \iota) \quad \mathsf{H}, \iota_0 \vdash \mathcal{Encl}(\mathsf{H}(\iota)) \triangleright \mathcal{E}(\mathsf{u.C})}{\mathsf{H}, \iota_0 \vdash \iota \triangleright \mathsf{u.C}} \qquad \text{(A-CTYPE)}$$

Auxiliary definitions:

$$\mathcal{Depth}(\mathsf{H}, \iota) = \begin{cases} 0, & \text{if } \iota = \iota_{root} \\ 1 + \mathcal{Depth}(\mathsf{H}, \mathcal{Encl}(\mathsf{H}(\iota))) \end{cases}$$

CaesarJ Component Model: Semantics (4)

THEOREM 1 (Preservation).

$$\left[\begin{array}{l} CT \ \mathsf{OK} \\ \mathsf{H}\ \mathsf{OK} \\ \mathsf{p} \vdash \mathsf{e} : \mathsf{t} \\ \mathsf{H}, \iota \vdash \iota \triangleright \langle \mathsf{p} \rangle \\ \mathsf{e}, \mathsf{H}, \iota \leadsto \mathsf{r}, \mathsf{H}' \end{array}\right] \Rightarrow \left[\begin{array}{l} \mathsf{H}'\ \mathsf{OK} \\ \mathsf{H}', \iota \vdash \iota \triangleright \langle \mathsf{p} \rangle \\ \quad \mathsf{r} = \mathsf{val} \ \wedge \ \mathsf{H}', \iota \vdash \mathsf{val} \triangleright \mathsf{t} \\ \vee \\ \quad \mathsf{r} = \mathsf{NullErr} \end{array}\right]$$

DEFINITION 1 (Finite Evaluation). *Define an evaluation relation $\leadsto_k$ as a copy of the rules for $\leadsto$. Replace each occurrence of $\leadsto$ in a premise by $\leadsto_{k-1}$. Replace $\leadsto$ in the conclusion of each rule and axiom with $\leadsto_k$. Note that the copied axioms are defined for all k. Add the following axiom:*

$$\mathsf{e}, \mathsf{H}, \iota \leadsto_0 \mathsf{KillErr}, \mathsf{H} \qquad (\textsc{Kill})$$

LEMMA 1 (Coverage). *For all natural numbers n and* e, H, ι, *there exists* r, H$'$ *such that*

$$\mathsf{e}, \mathsf{H}, \iota \leadsto_n \mathsf{r}, \mathsf{H}'$$