Définition d'un préordre sur les programmes Featherweight Java

Notes sur mon stage au LACL

Samy Avrillon

ENS de Lyon

été 2022

1 Featherweight Java

ENS DE LYON

ENS DE LYON

ENS DE LYON

ENS DE LYON

ENS DE LYON

1. Featherweight Java
   - Sa grammaire
   - Ses structures
   - Sa réduction
   - Son typage

ENS DE LYON

Sa grammaire

$e :=$

Sa grammaire

$$e := \quad x \qquad\qquad \text{E-Var}$$

Sa grammaire

$$
\begin{array}{rcll}
e & := & \mathtt{x} & \textsc{E-Var} \\
 & | & \mathtt{new}\ \mathtt{C}(\overline{e}) & \textsc{E-Cstr}
\end{array}
$$

ENS DE LYON

Sa grammaire

$$
\begin{array}{lll}
e := & \texttt{x} & \textsc{E-Var} \\
\mid & \texttt{new C}(\overline{e}) & \textsc{E-Cstr} \\
\mid & e\texttt{.f} & \textsc{E-Field}
\end{array}
$$

ENS DE LYON

Sa grammaire

$$
\begin{array}{rll}
e := & \texttt{x} & \textsc{E-Var} \\
\mid & \textbf{new } \texttt{C}(\overline{e}) & \textsc{E-Cstr} \\
\mid & e.\texttt{f} & \textsc{E-Field} \\
\mid & e.\texttt{m}(\overline{e}) & \textsc{E-Meth}
\end{array}
$$

ENS DE LYON

Sa grammaire

$e :=$	`x`	E-Var
$\mid$	**`new`** `C(`$\overline{e}$`)`	E-Cstr
$\mid$	e`.f`	E-Field
$\mid$	e`.m(`$\overline{e}$`)`	E-Meth
$\mid$	`(C)`e	E-Cast

ENS DE LYON

Sa grammaire

$$
\begin{array}{rll}
e := & \texttt{x} & \textsc{E-Var} \\
\mid & \textbf{new } \texttt{C}(\overline{e}) & \textsc{E-Cstr} \\
\mid & e.\texttt{f} & \textsc{E-Field} \\
\mid & e.\texttt{m}(\overline{e}) & \textsc{E-Meth} \\
\mid & (\texttt{C})e & \textsc{E-Cast}
\end{array}
$$

```
new Paire((D)new B().get(), new C(new A())).snd.apply()
```

ENS DE LYON

Ses structures

Ses structures

Class table

```
class A extends Object {...}
class B extends A {...}
class Paire extends Object {
    A fst;
    B snd;
    Paire(A fst, B snd){...}
    B getBFst(){
        return (B)this.fst;
    }
}
```

Ses structures

Class table + Expression = Programme

```
class A extends Object {...}
class B extends A {...}
class Paire extends Object {
    A fst;
    B snd;
    Paire(A fst, B snd){...}
    B getBFst(){
        return (B)this.fst;
    }
}
 new Paire(new A(), new B())
```

Sa réduction

Sa réduction

- **new** `C(e1,e2,...,en).field` $\rightarrow$ `ek` (R-Field)

Sa réduction

- **new** C(e1,e2,...,en).field $\rightarrow$ ek (R-Field)
- **new** C(e1,e2,...,en).meth(f1,f2,...,fn) $\rightarrow$ $e_{\texttt{C.meth}}$[**new** C(e1,e2,...,en)/**this**, f1/x1, f2/x2, ..., fn/xn] (R-Invk)

ENS DE LYON

Sa réduction

- **new** C(e1,e2,...,en).field $\rightarrow$ ek (R-Field)
- **new** C(e1,e2,...,en).meth(f1,f2,...,fn) $\rightarrow$ $e_{\texttt{C.meth}}$[**new** C(e1,e2,...,en)/**this**, f1/x1, f2/x2, ..., fn/xn] (R-Invk)
- (C) **new** D(...) $\rightarrow$ **new** D(...) si D $<:$ C (R-Cast)

ENS DE LYON

Sa réduction

- **new** C(e1,e2,...,en).field $\to$ ek (R-Field)
- **new** C(e1,e2,...,en).meth(f1,f2,...,fn) $\to$ $e_{\texttt{C.meth}}$[**new** C(e1,e2,...,en)/**this**, f1/x1, f2/x2, ..., fn/xn] (R-Invk)
- (C) **new** D(...) $\to$ **new** D(...) si D <: C (R-Cast)
- $e \to^* f \implies h[e] \to^* h[f]$

Son typage

Son typage

$$\frac{\mathtt{x} \in \Gamma}{\mathcal{A}, \Gamma \vdash \mathtt{x} : \Gamma(\mathtt{x})} \qquad \frac{\mathcal{A}, \Gamma \vdash e : \mathtt{C}c \quad \mathtt{C}\,\mathtt{f} \in \mathsf{fields}(\mathtt{C}c)}{\mathcal{A}, \Gamma \vdash e.\mathtt{f} : \mathtt{C}}$$

ENS DE LYON

Son typage

$$\frac{\mathtt{x} \in \Gamma}{\mathcal{A}, \Gamma \vdash \mathtt{x} : \Gamma(\mathtt{x})} \qquad \frac{\mathcal{A}, \Gamma \vdash e : \mathtt{C}c \quad \mathtt{C}\,\mathtt{f} \in \mathsf{fields}(\mathtt{C}c)}{\mathcal{A}, \Gamma \vdash e.\mathtt{f} : \mathtt{C}}$$

$$\frac{\begin{array}{c}\mathcal{A}, \Gamma \vdash e : \mathtt{C}c \quad \mathcal{A}, \Gamma \vdash \overline{e_x} : \overline{\mathtt{C}\mathtt{X}} \\ \overline{\mathtt{C}\mathtt{X}} <: \overline{\mathtt{D}} \quad \mathsf{mtype}(\mathtt{m}, \mathtt{C}c) = \overline{\mathtt{D}} \to \mathtt{C}\end{array}}{\mathcal{A}, \Gamma \vdash e.\mathtt{m}(\overline{e_x}) : \mathtt{C}}$$

$$\frac{\mathcal{A}, \Gamma \vdash \overline{e_x} : \overline{\mathtt{C}\mathtt{X}} \quad \overline{\mathtt{C}\mathtt{X}} <: \overline{\mathtt{D}} \quad \mathsf{constructor}(\mathtt{C}) = \overline{\mathtt{D}}}{\mathcal{A}, \Gamma \vdash \mathbf{new}\ \mathtt{C}(\overline{e_x}) : \mathtt{C}}$$

$$\frac{\mathcal{A}, \Gamma \vdash e : \mathtt{D} \quad \mathtt{C} <: \mathtt{D} \quad \mathtt{C} \neq \mathtt{D}}{\mathcal{A}, \Gamma \vdash (\mathtt{C})\,e : \mathtt{C}} \qquad \frac{\mathcal{A}, \Gamma \vdash e : \mathtt{D} \quad \mathtt{D} <: \mathtt{C}}{\mathcal{A}, \Gamma \vdash (\mathtt{C})\,e : \mathtt{C}}$$

ENS DE LYON

2 Notre problème

- Qu'est-ce qu'une équivalence
- La méthode classique

Qu'est-ce qu'une équivalence

Sens

Qu'est-ce qu'une équivalence

Sens

- Deux programmes «font la même chose»

ENS DE LYON

Qu'est-ce qu'une équivalence

Sens

- Deux programmes «font la même chose»
- Assez stricte

ENS DE LYON

Qu'est-ce qu'une équivalence

Sens

- Deux programmes «font la même chose»
- Assez stricte
- Assez large

ENS DE LYON

Qu'est-ce qu'une équivalence

Sens

- Deux programmes «font la même chose»
- Assez stricte
- Assez large

Utilisations

- Utiliser un autre algorithme (complexités)
- Utiliser une version plus sécurisée
- Effectuer une optimisation

ENS DE LYON

La méthode classique

Contexte : C, typiquement $\boxed{[\cdot].\texttt{meth(}\textbf{new}\ \texttt{A())}}$

La méthode classique

Contexte : C, typiquement $\boxed{[\cdot]\texttt{.meth(new A())}}$

Contextualisation : $C[P]$, exemple $\boxed{(\mathcal{A}, \texttt{e.meth(new A())})}$

ENS DE LYON

La méthode classique

Contexte : C, typiquement $\boxed{[\cdot].\texttt{meth(}\textbf{new}\ \texttt{A())}}$

Contextualisation : $C[P]$, exemple $\boxed{(\mathcal{A}, \texttt{e.meth(}\textbf{new}\ \texttt{A()))}}$

Préordre : $P \prec Q \iff \forall C \quad \exists v \quad C[P] \to^* v \implies C[Q] \to^* v$

ENS DE LYON

La méthode classique

Contexte : C, typiquement $\boxed{[\cdot].\texttt{meth(}\textbf{new}\ \texttt{A())}}$

Contextualisation : $C[P]$, exemple $\boxed{(\mathcal{A}, \texttt{e.meth(}\textbf{new}\ \texttt{A())})}$

Préordre : $P \prec Q \iff \forall C \quad \exists v \quad C[P] \to^* v \implies C[Q] \to^* v$

Équivalence : $P \equiv Q \iff P \prec Q \wedge Q \prec P$

ENS DE LYON

3 Une nouvelle structure

- Comparer uniquement les CT
- Idée d'une structure
- Exemple de *test interface*
- Théorème de cohérence
- Définition du préordre

ENS DE LYON

Comparer uniquement les CT

Comparer uniquement les CT

```
class XXX extends XXX {}
⋮
expr
```

Comparer uniquement les CT

```
class XXX extends XXX {}
⋮
expr
```

```
class XXX extends XXX {
}
⋮
class Main {
    Object main(){
        return expr;
    }
}
```

ENS DE LYON

Comparer uniquement les CT

```
class XXX extends XXX {}
⋮
expr
```

```
class XXX extends XXX {
}
⋮
class Main {
    Object main(){
        return expr;
    }
}
```

new Main().main() $\rightarrow$ expr

Idée d'une structure

- Les simples expressions à trou

Idée d'une structure

- Les simples expressions à trou
  - Pas assez puissant

ENS DE LYON

Idée d'une structure

- Les simples expressions à trou
  - Pas assez puissant
  - Bloque sur les différences syntaxiques

ENS DE LYON

Idée d'une structure

- Les simples expressions à trou
  - Pas assez puissant
  - Bloque sur les différences syntaxiques
- Inclure une *class table* de tests

ENS DE LYON

Idée d'une structure

- Les simples expressions à trou
  - Pas assez puissant
  - Bloque sur les différences syntaxiques
- Inclure une *class table* de tests
- Empêcher au test d'accéder à tout

ENS DE LYON

Idée d'une structure

- Les simples expressions à trou
  - Pas assez puissant
  - Bloque sur les différences syntaxiques
- Inclure une *class table* de tests
- Empêcher au test d'accéder à tout
- Utiliser un mécanisme déjà là

ENS DE LYON

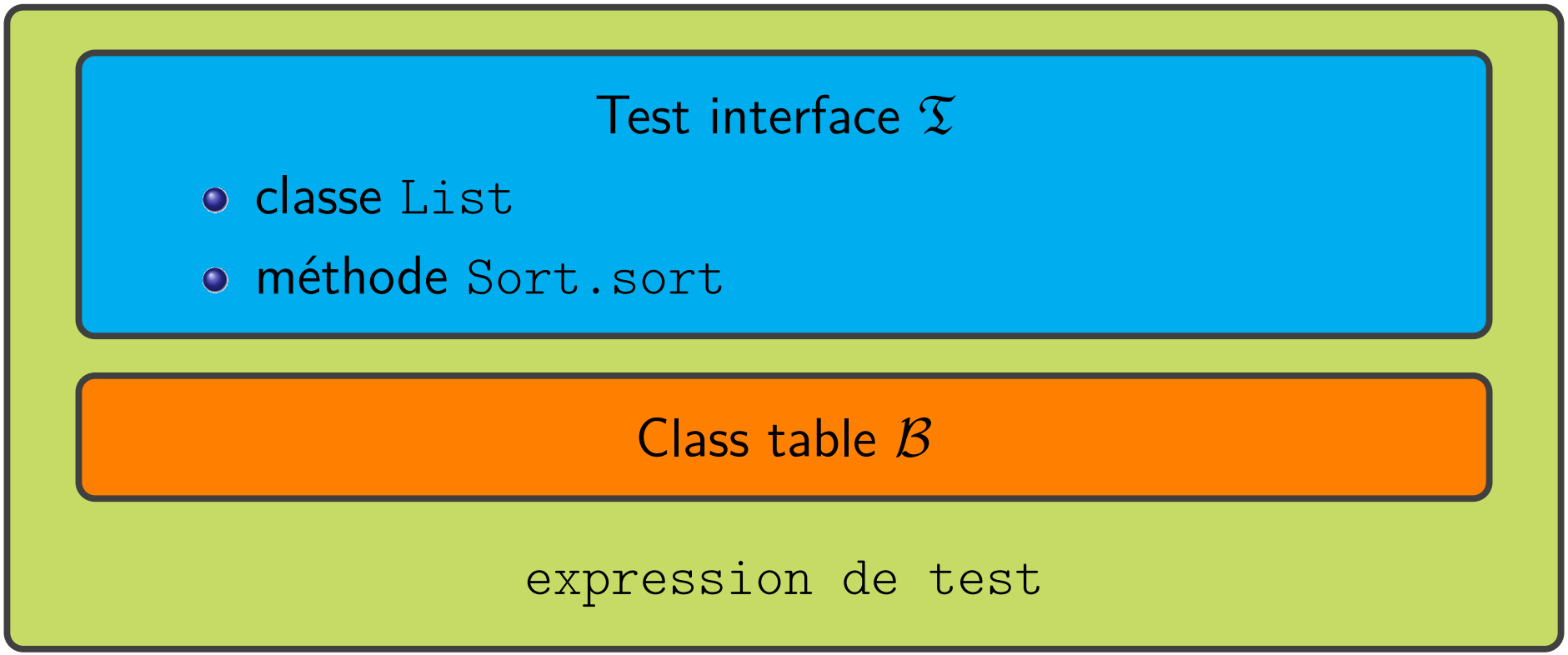
Idée d'une structure

- Les simples expressions à trou
  - Pas assez puissant
  - Bloque sur les différences syntaxiques
- Inclure une *class table* de tests
- Empêcher au test d'accéder à tout
- Utiliser un mécanisme déjà là
- Inspiration : *header file* en C

ENS DE LYON

Idée d'une structure

Compilation (typage)



Idée d'une structure

Execution (réduction)

Class table $\mathcal{A}$

- classe `List`
- méthode `Sort.sort`
- classe `Map`

Class table $\mathcal{B}$

`expression de test`

Exemple de *test interface*

```
Number {}
```

Exemple de *test interface*

```
Number {}

Int {}
Int : Int add(Int)
Int <: Number
```

Exemple de *test interface*

```
Number {}

Int {}
Int : Int add(Int)
Int <: Number

Frac (Int numerateur, Int)
RichInt {Int value}
```

ENS DE LYON

Exemple de *test interface*

Relations à créer

Exemple de *test interface*

Relations à créer

- Opérateur d'implémentation ($\triangleright$)

ENS DE LYON

Exemple de *test interface*

Relations à créer

- Opérateur d'implémentation ($\triangleright$)
- Typage avec la TI ($\Vdash$)

Exemple de *test interface*

Relations à créer

- Opérateur d'implémentation ($\triangleright$)
- Typage avec la TI ($\Vdash$)
- Validation d'une CT dans une TI

ENS DE LYON

Théorème de cohérence

Théorème de cohérence

Theorem

Soit une test interface $\mathfrak{T}$*, une class table* $\mathcal{A}$*, un couple* $(\mathcal{B}, e)$ *class table* $\times$ *expression fermée appelée « test », et un environnement de typage* Γ *vérifiant*

- $\mathcal{A} \triangleright \mathfrak{T}$
- $\mathcal{B}$ OK IN $\mathfrak{T}$
- $\mathfrak{T}, \mathcal{B}, \Gamma \Vdash e : \mathtt{D}$

Alors il existe $\mathtt{C}$ *tel que* $\mathcal{A} \oplus \mathcal{B}, \Gamma \vdash e : \mathtt{C}$ *et* $\mathtt{C} <: \mathtt{D}$.

ENS DE LYON

Définition du préordre

Définition du préordre

Definition

Soit une *test interface* $\mathfrak{T}$.
Soient deux *class tables* $\mathcal{A}$ et $\mathcal{B}$ qui implémentent toutes deux $\mathfrak{T}$
$\mathcal{A} \prec_{\mathfrak{T}} \mathcal{A}'$ si et seulement si
Pour tout test $(\mathcal{B}, e)$ tel que $\mathcal{B}$ OK IN $\mathfrak{T}$ et $\mathfrak{T}, \mathcal{B} \Vdash e : \mathtt{D}$

$$\exists v \quad \mathcal{A} \oplus \mathcal{B} \to^* v \implies \mathcal{A}' \oplus \mathcal{B} \to^* v$$

ENS DE LYON

4 Les valeurs infinies

- Qu'est-ce qu'un context lemma
- Pourquoi est-il faux ici
- Idée de valeurs infinies
- Explication de la formalisation
- Redéfinition des préordres
- Propriétés

ENS DE LYON

Qu'est-ce qu'un context lemma

ENS DE LYON

Qu'est-ce qu'un context lemma

$$(\mathcal{A}, e) \prec (\mathcal{B}, f) \implies (\mathcal{A}, h[e]) \prec (\mathcal{A}, h[f])$$

ENS DE LYON

Pourquoi est-il faux ici

```
public Static {
    IList intList(Int i){
        return new IList(i, intList(i+1));
    }
    IList zeroList(){
        return new IList(0, zeroList());
    }
}
```

$$e_i = \texttt{intList(0)} \rightarrow^*$$

$$\textbf{new}\ \texttt{IList(0,}\textbf{new}\ \texttt{IList(1,}\textbf{new}\ \texttt{IList(2,intList(3)))))}$$

$$e_0 = \texttt{zeroList()} \rightarrow^*$$

$$\textbf{new}\ \texttt{IList(0,}\textbf{new}\ \texttt{IList(0,}\textbf{new}\ \texttt{IList(0,zeroList())))}$$

Pourquoi est-il faux ici

$$
\begin{array}{rcl}
h[e_i] & = & \texttt{intList(0).next.obj} \\
& \to & \texttt{\textbf{new} IList(0, intList(0+1)).next.obj} \\
& \to & \texttt{intList(0+1).obj} \\
& \to & \texttt{\textbf{new} IList(0+1, intList(0+1+1)).obj} \\
& \to & \texttt{0+1} \\
& \to^* & \texttt{1} \\
h[e_0] & = & \texttt{zeroList().next.obj} \\
& \to & \texttt{\textbf{new} IList(0, zeroList()).next.obj} \\
& \to & \texttt{zeroList().obj} \\
& \to & \texttt{\textbf{new} IList(0, zeroList()).obj} \\
& \to & \texttt{0}
\end{array}
$$

ENS DE LYON

Explication de la formalisation

Comment capturer ces « valeurs infinies » ?

ENS DE LYON

Explication de la formalisation

Comment capturer ces « valeurs infinies » ?

Valeurs à variable : $\hbar$

```
new C(x,new S(new S(y,new G()),z))
```

ENS DE LYON

Explication de la formalisation

Comment capturer ces « valeurs infinies » ?

Valeurs à variable : $\hbar$

```
new C(x,new S(new S(y,new G()),z))
```

$$\boxed{e \to^* v \implies f \to^* v} \Rrightarrow \boxed{\forall \hbar \forall \alpha \ (e \to^*_{\mathcal{A}} \hbar[\alpha] \implies \exists \beta \quad f \to^*_{\mathcal{B}} \hbar[\beta])}$$

Explication de la formalisation

Comment capturer ces « valeurs infinies » ?

Valeurs à variable : $\hbar$

```
new C(x,new S(new S(y,new G()),z))
```

$$\boxed{e \to^* v \implies f \to^* v} \Rrightarrow \boxed{\forall \hbar \forall \alpha \ (e \to^*_{\mathcal{A}} \hbar[\alpha] \implies \exists \beta \quad f \to^*_{\mathcal{B}} \hbar[\beta])}$$

$$e \to^* \texttt{new C(new D(...))} \implies f \to^* \texttt{new C(new D(...))}$$

```
new IList(0,new IList(1,...))
```

$$\neq$$

```
new IList(0,new IList(0,...))
```

Redéfinition des préordres

$$\boxed{\mathcal{A} \prec_{\mathfrak{T}} \mathcal{A}'} \qquad \forall(\mathcal{B}, e) \left| \begin{array}{l} \mathcal{B} \text{ OK in } \mathfrak{T} \\ \mathfrak{T}, \mathcal{B} \Vdash e : \mathtt{D} \end{array} \right.$$

$$\mathcal{A} \oplus \mathcal{B} \Downarrow v \implies \mathcal{A}' \oplus \mathcal{B} \Downarrow v$$

$$\boxed{\mathcal{A} \preccurlyeq_{\mathfrak{T}} \mathcal{A}'} \qquad \forall(\mathcal{B}, e) \left| \begin{array}{l} \mathcal{B} \text{ OK in } \mathfrak{T} \\ \mathfrak{T}, \mathcal{B} \Vdash e : \mathtt{D} \end{array} \right.$$

$$\forall \hbar, \alpha \quad \mathcal{A} \oplus \mathcal{B} \to^* \hbar[\alpha] \implies \exists \beta \quad \mathcal{A}' \oplus \mathcal{B} \to^* \hbar[\beta]$$

ENS DE LYON

Propriétés

ENS DE LYON

Propriétés

Propriété

$$\mathcal{A} \preccurlyeq \mathcal{B} \implies \mathcal{A} \prec \mathcal{B}$$

ENS DE LYON

Propriétés

Propriété

$$\mathcal{A} \preccurlyeq \mathcal{B} \implies \mathcal{A} \prec \mathcal{B}$$

Propriété (Context Lemma)

$$\forall \mathcal{A} \quad \forall e, f \quad \forall (h, \mathcal{B})$$
$$(e, \mathcal{A}) \preccurlyeq (f, \mathcal{B}) \implies (h[e], \mathcal{A} \oplus \mathcal{B}) \preccurlyeq (h[f], \mathcal{A} \oplus \mathcal{B})$$

ENS DE LYON

Conclusion

Résumé

- On a créé un préordre

Futur

ENS DE LYON

Conclusion

Résumé

- On a créé un préordre
- Assez puissant (sémantique)

Futur

ENS DE LYON

Conclusion

Résumé

- On a créé un préordre
- Assez puissant (sémantique)
- Assez tolérant

Futur

ENS DE LYON

Conclusion

Résumé

- On a créé un préordre
- Assez puissant (sémantique)
- Assez tolérant
- Context Lemma

Futur

ENS DE LYON

Conclusion

Résumé

- On a créé un préordre
- Assez puissant (sémantique)
- Assez tolérant
- Context Lemma
- Concepts réutilisables

Futur

ENS DE LYON

Conclusion

Résumé

- On a créé un préordre
- Assez puissant (sémantique)
- Assez tolérant
- Context Lemma
- Concepts réutilisables

Futur

- Formalisation dans un assistant de preuve

ENS DE LYON

Conclusion

Résumé

- On a créé un préordre
- Assez puissant (sémantique)
- Assez tolérant
- Context Lemma
- Concepts réutilisables

Futur

- Formalisation dans un assistant de preuve
- Comparer uniquement certains types de retour

ENS DE LYON

Conclusion

Résumé

- On a créé un préordre
- Assez puissant (sémantique)
- Assez tolérant
- Context Lemma
- Concepts réutilisables

Futur

- Formalisation dans un assistant de preuve
- Comparer uniquement certains types de retour
- Théorèmes pour simplifier l'équivalence

ENS DE LYON

Conclusion

Résumé

- On a créé un préordre
- Assez puissant (sémantique)
- Assez tolérant
- Context Lemma
- Concepts réutilisables

Futur

- Formalisation dans un assistant de preuve
- Comparer uniquement certains types de retour
- Théorèmes pour simplifier l'équivalence
- Obtenir des retours

ENS DE LYON

Conclusion

FIN

Leur grammaire

$$
\begin{array}{rll}
\mathsf{TR} := & \mathsf{C} : \mathsf{C}\ \mathsf{m}(\overline{\mathsf{C}}) & \textsc{TIG-Meth} \\
\mid & \mathsf{C}\ \{\overline{\mathsf{C}}\ \overline{\mathtt{f}}\} & \textsc{TIG-Attr} \\
\mid & \mathsf{C}\ (\overline{\mathsf{C}}\ \overline{\mathtt{?f}}) & \textsc{TIG-Cstr} \\
\mid & \mathsf{C} <: \mathsf{C} & \textsc{TIG-Cast}
\end{array}
$$

ENS DE LYON

L'opérateur d'implémentation

$$\frac{\overline{\mathtt{F}}\ \overline{\mathtt{f}} \subset \mathsf{fields}_{\mathcal{A}}(C) \quad \overline{\mathtt{F}} <:_{\mathcal{A}} \overline{\mathtt{E}}}{\mathcal{A} \triangleright [\mathtt{C}\ \ \{\overline{\mathtt{E}}\ \overline{\mathtt{f}}\}]}$$

$$\frac{\begin{array}{l}\exists \overline{\mathtt{f'}} \quad \overline{\mathtt{f0}} = \overline{\mathtt{f}} \boxplus \overline{\mathtt{f'}} \\ \overline{\mathtt{F}} = \overline{\mathtt{E}} \quad \text{où } \mathtt{f} \text{ est défini} \\ \overline{\mathtt{E}} <: \overline{\mathtt{F}} \quad \wedge \quad \overline{\mathtt{F}}\ \overline{\mathtt{f0}} = \mathsf{fields}_{\mathcal{A}}(C)\end{array}}{\mathcal{A} \triangleright [\mathtt{C}\ \ (\overline{\mathtt{E}}\ \overline{\mathtt{f}})]}$$

$$\frac{\mathsf{mtype}_{\mathcal{A}}(\mathtt{m}, \mathtt{C}) = \overline{\mathtt{E}} \to \mathtt{H} \quad \overline{\mathtt{F}} <:_{\mathcal{A}} \overline{\mathtt{E}} \quad \mathtt{H} <:_{\mathcal{A}} \mathtt{G}}{\mathcal{A} \triangleright [\mathtt{C}\ :\ \mathtt{G}\ \ \mathtt{m}(\overline{\mathtt{F}})]}$$

$$\frac{\mathtt{C} <:_{\mathcal{A}} \mathtt{D}}{\mathcal{A} \triangleright [\mathtt{C} <: \mathtt{D}]}$$

ENS DE LYON

Nouvelles applications de *lookup*

$$\frac{[\mathtt{C}\,(\overline{\mathtt{D}}\;\mathtt{f})] \in \mathfrak{T}}{\mathsf{construct}(\mathtt{C}) = \overline{\mathtt{D}}} \qquad \frac{[\mathtt{C}\;\;(\overline{\mathtt{D}}\;\mathtt{f})] \in \mathfrak{T}}{\mathsf{fields}(\mathtt{C}) = \underbrace{\overline{\mathtt{D}}\;\mathtt{f}}_{\text{pour f défini}} + \bigcup_{C<:E} \mathsf{fields}(\mathtt{E})}$$

$$\frac{\mathtt{class\ C}\{\mathtt{C}\,(\overline{\mathtt{D}}\;\mathtt{f})\{\ldots\}\;\ldots\;\} \in \mathcal{B}}{\mathsf{construct}(\mathtt{C}) = \overline{\mathtt{D}}} \qquad \frac{[\mathtt{C}\;\;\{\overline{\mathtt{D}}\;\mathtt{f}\}] \in \mathfrak{T}}{\mathsf{fields}(\mathtt{C}) = \overline{\mathtt{D}}\;\mathtt{f} + \bigcup_{C<:E} \mathsf{fields}(\mathtt{E})}$$

$$\frac{\mathtt{class\ C}\;\{\overline{\mathtt{D}}\;\mathtt{f};\;\;\ldots\} \in \mathcal{B}}{\mathsf{fields}(\mathtt{C}) = \overline{\mathtt{D}}\;\mathtt{f} + \bigcup_{C<:E} \mathsf{fields}(\mathtt{E})} \qquad \frac{[\mathtt{C}\;:\;\mathtt{E}\;\mathtt{m}(\overline{\mathtt{D}})] \in \mathfrak{T}}{\mathsf{mtype}(\mathtt{m},\mathtt{C}) = \overline{\mathtt{D}} \to \mathtt{E}}$$

$$\frac{\mathtt{class\ C}\;\{\ldots;\;\mathtt{E}\;\mathtt{m}(\overline{\mathtt{D}}\;\overline{\mathtt{x}})\} \in \mathcal{B}}{\mathsf{mtype}(\mathtt{m},\mathtt{C}) = \overline{\mathtt{D}} \to \mathtt{E}} \qquad \frac{\mathsf{mtype}(\mathtt{m},\mathtt{C'}) = \overline{\mathtt{D}} \to \mathtt{E} \quad \mathtt{C} <: \mathtt{C'}}{\mathsf{mtype}(\mathtt{m},\mathtt{C}) = \overline{\mathtt{D}} \to \mathtt{E}}$$

Nouveau typage

$$\frac{\mathfrak{T}, \mathcal{B}, \Gamma \Vdash e : \mathtt{C}c \quad \mathtt{C}\,\mathtt{f} \in \mathsf{fields}(\mathtt{C}c)}{\mathfrak{T}, \mathcal{B}, \Gamma \Vdash e.\mathtt{f} : \mathtt{C}}$$

$$\frac{\mathtt{x} \in \Gamma}{\mathfrak{T}, \mathcal{B}, \Gamma \Vdash \mathtt{x} : \Gamma(\mathtt{x})} \qquad \frac{\mathfrak{T}, \mathcal{B}, \Gamma \Vdash e : \mathtt{C}c \quad \mathfrak{T}, \mathcal{B}, \Gamma \Vdash \overline{e_x} : \overline{\mathtt{C}\mathtt{X}} \quad \overline{\mathtt{C}\mathtt{X}} <: \overline{\mathtt{D}} \quad \mathsf{mtype}(\mathtt{m}, \mathtt{C}c) = \overline{\mathtt{D}} \to \mathtt{C}}{\mathfrak{T}, \mathcal{B}, \Gamma \Vdash e.\mathtt{m}(\overline{e_x}) : \mathtt{C}}$$

$$\frac{\mathfrak{T}, \mathcal{B}, \Gamma \Vdash \overline{e_x} : \overline{\mathtt{C}\mathtt{X}} \quad \overline{\mathtt{C}\mathtt{X}} <: \overline{\mathtt{D}} \quad \mathsf{constructor}(\mathtt{C}) = \overline{\mathtt{D}}}{\mathfrak{T}, \mathcal{B}, \Gamma \Vdash \mathbf{new}\ \ \mathtt{C}(\overline{e_x}) : \mathtt{C}}$$

$$\frac{\mathfrak{T}, \mathcal{B}, \Gamma \Vdash e : \mathtt{D} \quad \mathtt{C} <: \mathtt{D} \quad \mathtt{C} \neq \mathtt{D}}{\mathfrak{T}, \mathcal{B}, \Gamma \Vdash (\mathtt{C})\, e : \mathtt{C}} \qquad \frac{\mathfrak{T}, \mathcal{B}, \Gamma \Vdash e : \mathtt{D} \quad \mathtt{D} <: \mathtt{C}}{\mathfrak{T}, \mathcal{B}, \Gamma \Vdash (\mathtt{C})\, e : \mathtt{C}}$$

ENS DE LYON