Spécification et Implémentation des Langages à Objets

Roland Ducournau

LIRMM / INFO / DOC
CNRS – Université Montpellier II

Journée LIRMM/INFO

LIRMM

Plan

Plan

LIRMM

Motivation

La plus grande réussite du dernier millénaire ?

Les objets (programmation, modélisation, etc.)

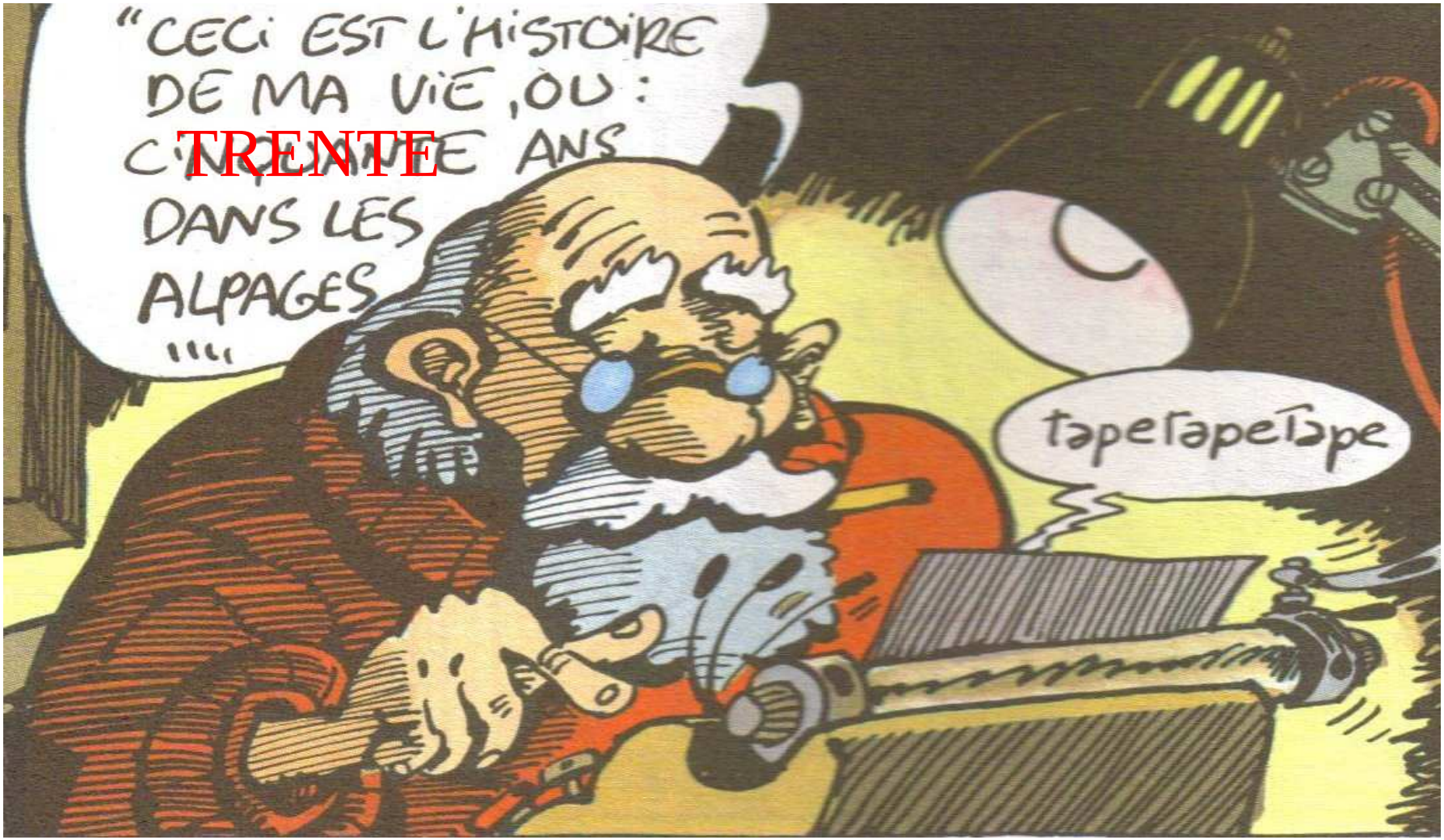
Motivation

La plus grande réussite du dernier millénaire ?

Les objets (programmation, modélisation, etc.)

Le plus grand échec du dernier millénaire ?

C++ !


"CECI EST L'HISTOIRE DE MA VIE, OU : TRENTE ANS DANS LES ALPAGES"
tapetapetape

Plan

Les trois mousquetaires (© A. Dumas)

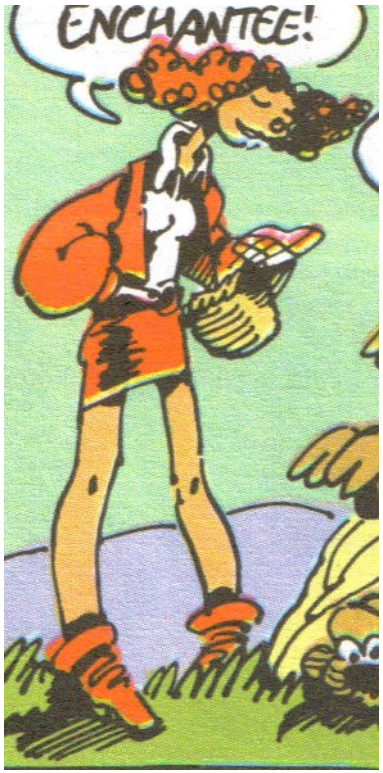
1986-1994

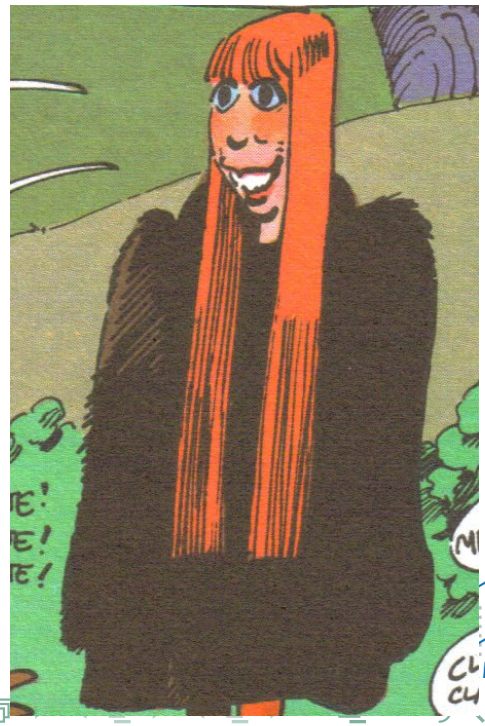
- Michel Habib
- Marianne Huchard
- Marie-Laure Mugnier

Les trois mousquetaires (© A. Dumas)

1986-1994

- Michel Habib
- Marianne Huchard
- Marie-Laure Mugnier





Spécification et implémentation ...

Héritage multiple

- Formalisation des techniques de linéarisation
- Utilisées par CLOS, DYLAN, PYTHON, C++, ...

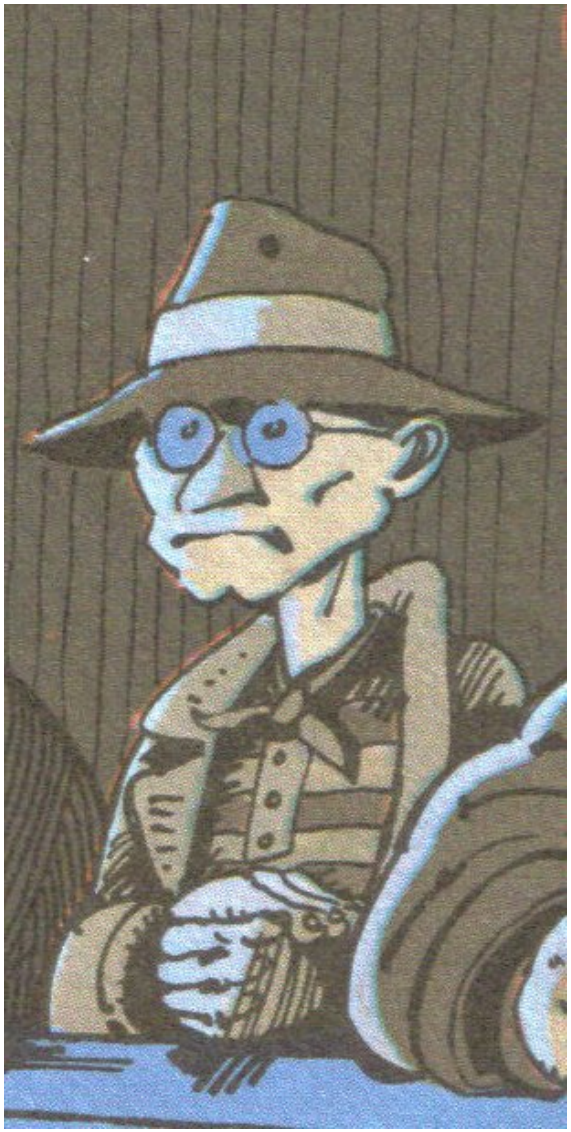
YAFOOL

- le plus beau langage objet du monde ...
- ... passé dans les poubelles de l'histoire

Contexte de typage dynamique

LISP, CLOS, YAFOOL, ...

LIRMM



avec Joël Quinqueton (le barde)

Plan

Vingt ans après (© A. Dumas)

2001-2006

- Jean Privat (soutenance en juillet 2006, post-doc Purdue U.)
- Floréal Morandat (thèse commencée en septembre 2006)
- stagiaires de DEA/M2R (JP, P. Takhedmit, N. Desnos, FM, ...)
- étudiants (IUP2, maîtrise/M1)

Implémentation efficace de l'héritage multiple

- étude de techniques particulières
- simulation sur gros benchmarks
- schéma de compilation global/séparé
- spécification du langage PRM
- compilateur PRM autogène

Passage au typage statique

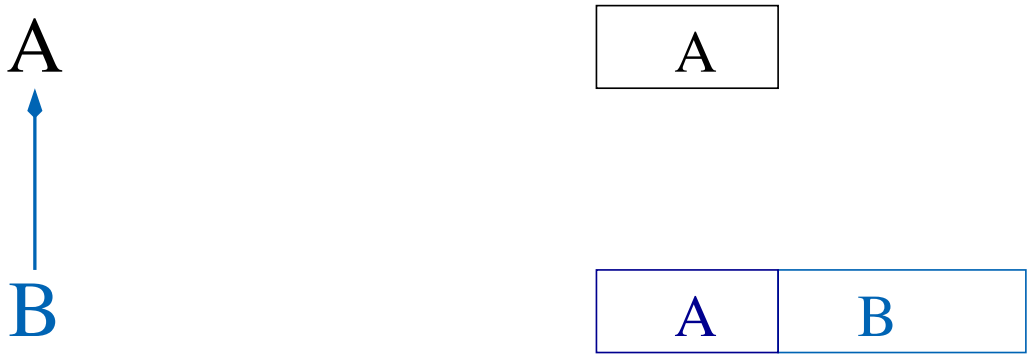
Implémentation des objets

3 mécanismes originaux

- envoi de message (ou liaison tardive)
  le polymorphisme ne permet pas l'appel statique
- accès aux attributs
  le polymorphisme ne garantit pas la position de l'attribut
- test de sous-typage
  le typage n'est pas si sûr que ça !
  (vieille spécialité du LIRMM :
  M. Habib, L. Nourine, E. Thierry, O. Raynaud)

Héritage simple, typage statique

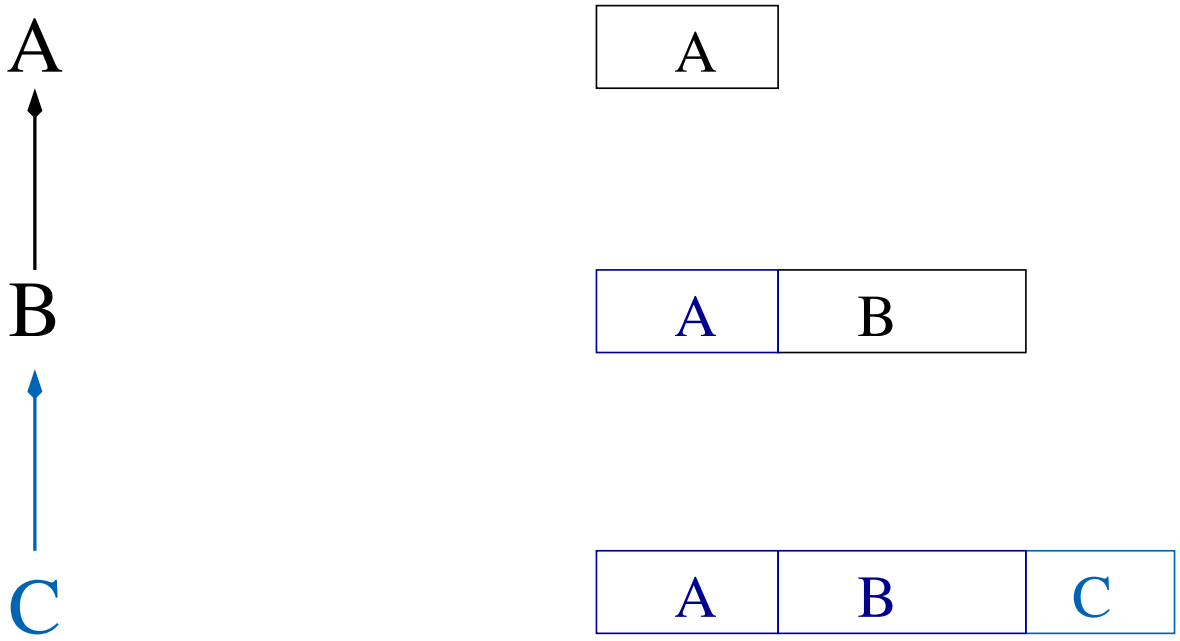
Héritage simple, typage statique



Problème : héritage multiple impossible

- Invariants de référence et de position
- Espace linéaire dans la taille de la relation de spécialisation
- Implémentation idéale

Héritage simple, typage statique



Problème : héritage multiple impossible

- Invariants de référence et de position
- Espace linéaire dans la taille de la relation de spécialisation
- Implémentation idéale

Alternatives pour l'héritage multiple

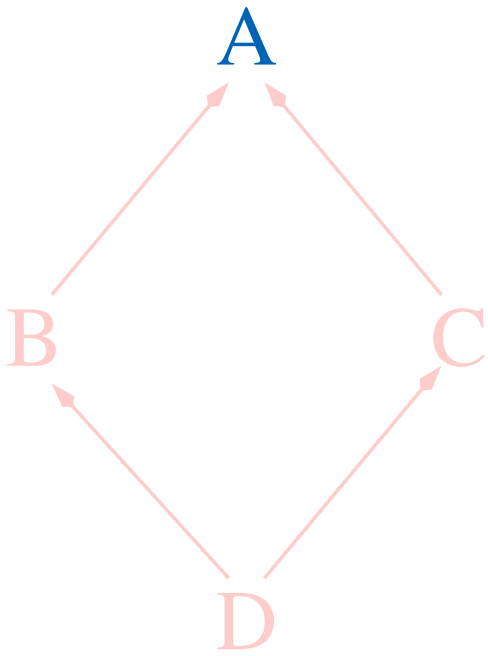
Coloration (1990-2006)

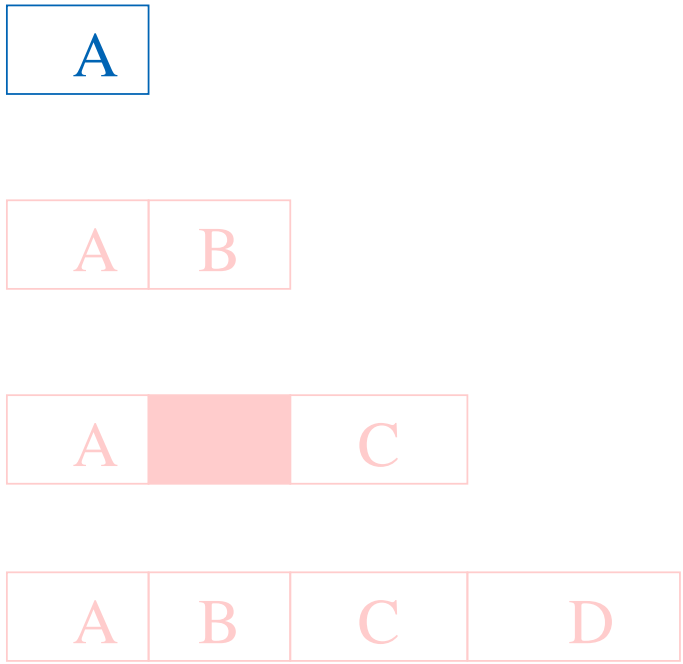
idéale mais incompatible avec le chargement dynamique

Hachage parfait (2005-2006)

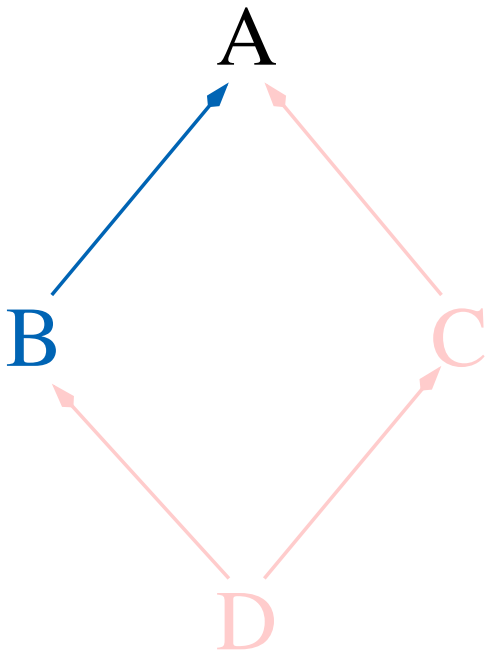
parfaitement incrémental mais efficacité moindre

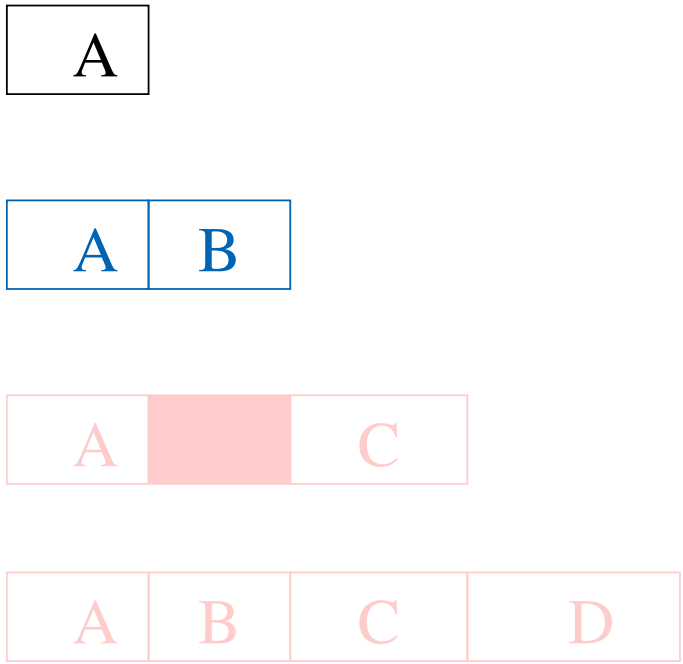
Coloration





Coloration

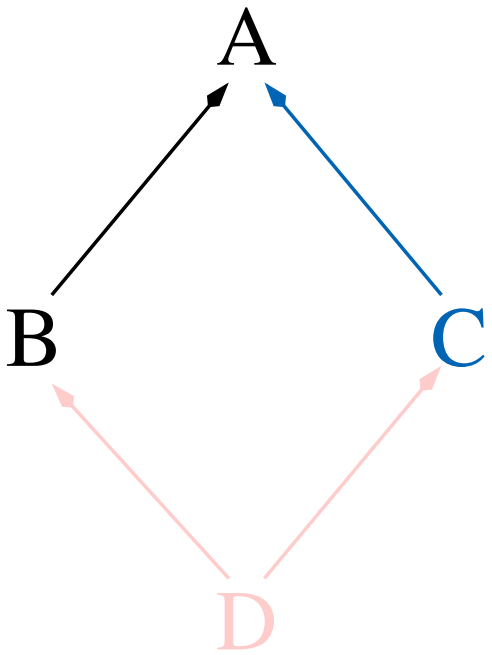


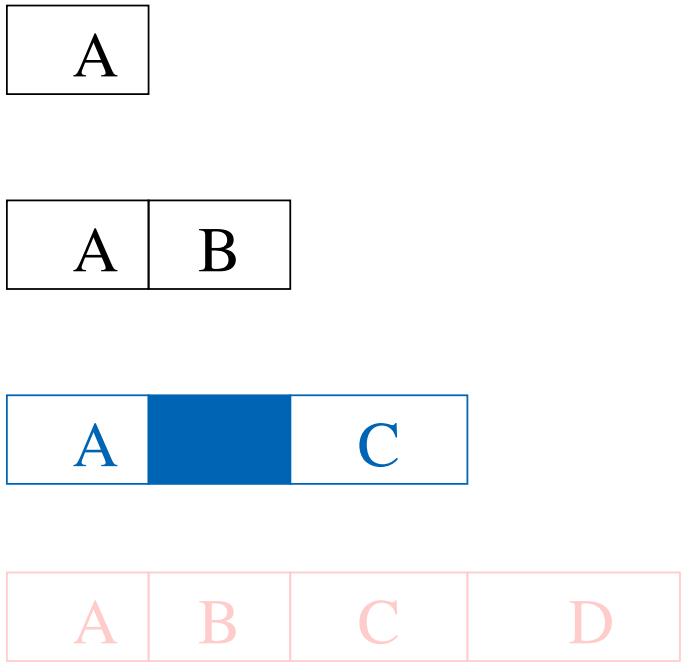


Problème : chargement dynamique impossible

- La coloration est une optimisation globale
- Invariants maintenus au prix de quelques trous

Coloration

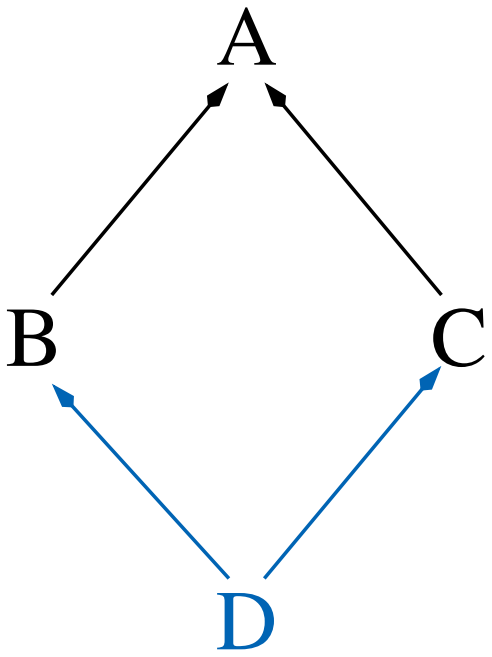


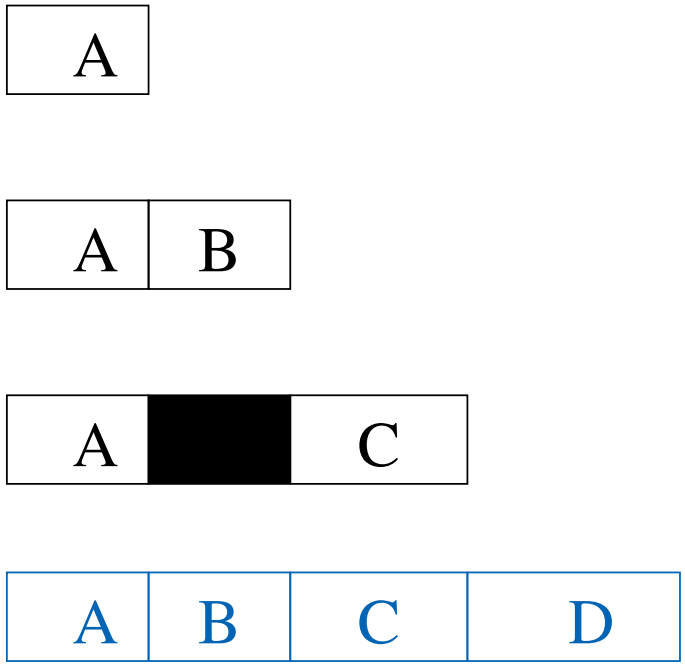


Problème : chargement dynamique impossible

- La coloration est une optimisation globale
- Invariants maintenus au prix de quelques trous

Coloration





Problème : chargement dynamique impossible

- La coloration est une optimisation globale
- Invariants maintenus au prix de quelques trous

Coloration

Quelques résultats

- Complexité (P. Takhedmit, Ch. Paul)
  problème de coloration de graphe, mais pas minimum
  NP-difficile dans la plupart des cas
- Heuristiques efficaces
- Simulation sur les benchmarks précités
- Schéma d'utilisation en compilation séparée
- Synthèse sur la technique
  3 "inventeurs" indépendants et complémentaires :
  Dixon 89, Pugh & Weddell 90, Ducournau 91, Vitek 97

LIRMM

Hachage Parfait

Sous-typage par table de hachage

Soit une classe C et un objet o instance de D :

- $o \in C \iff id_C$ est trouvé dans la table de o

Hachage Parfait

Sous-typage par table de hachage

Soit une classe C et un objet o instance de D :

- $o \in C \iff id_C$ est trouvé dans la table de o

Particularité

Au chargement de la classe D, toutes ses super-classes sont connues.

- plus besoin d'ajout, ni de retrait
- $\Rightarrow$ La table de D peut être optimisée statiquement

Hachage Parfait

Sous-typage par table de hachage

Soit une classe C et un objet o instance de D :

- $o \in C \iff id_C$ est trouvé dans la table de o

Particularité

Au chargement de la classe D, toutes ses super-classes sont connues.

- plus besoin d'ajout, ni de retrait
- $\Rightarrow$ La table de D peut être optimisée statiquement

Solution : le "hachage parfait" [Sprugnoli 77]

- hachage sans collision
- temps constant, espace à déterminer par l'expérience

Application à JAVA (1 pierre, 2 coups)

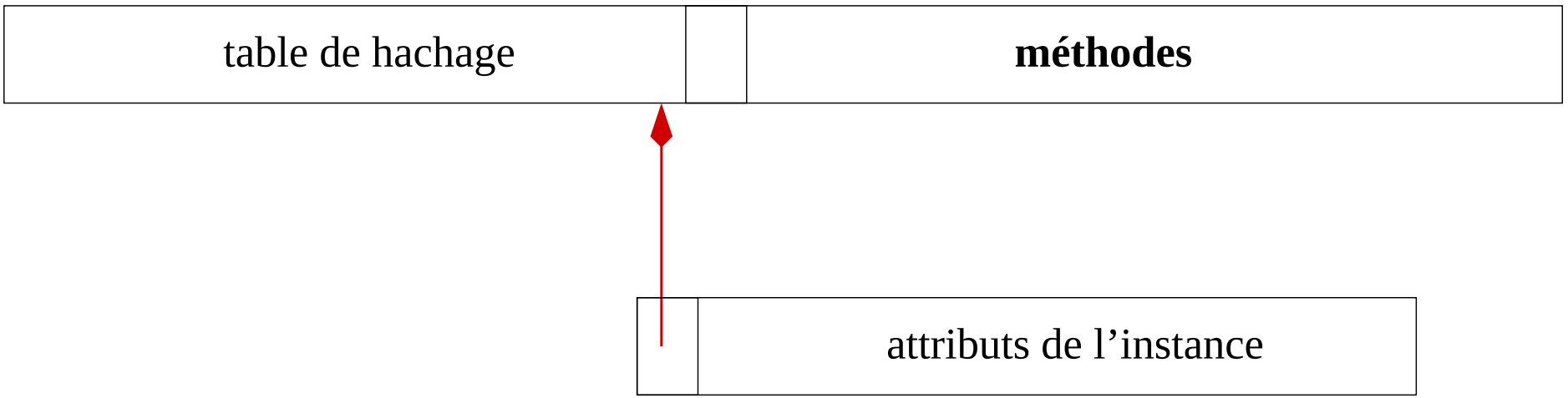


Table bidirectionnelle de la classe *C*

- Partie positive pour les classes, comme en héritage simple
- Partie négative pour les interfaces, avec hachage parfait

Application à JAVA (1 pierre, 2 coups)

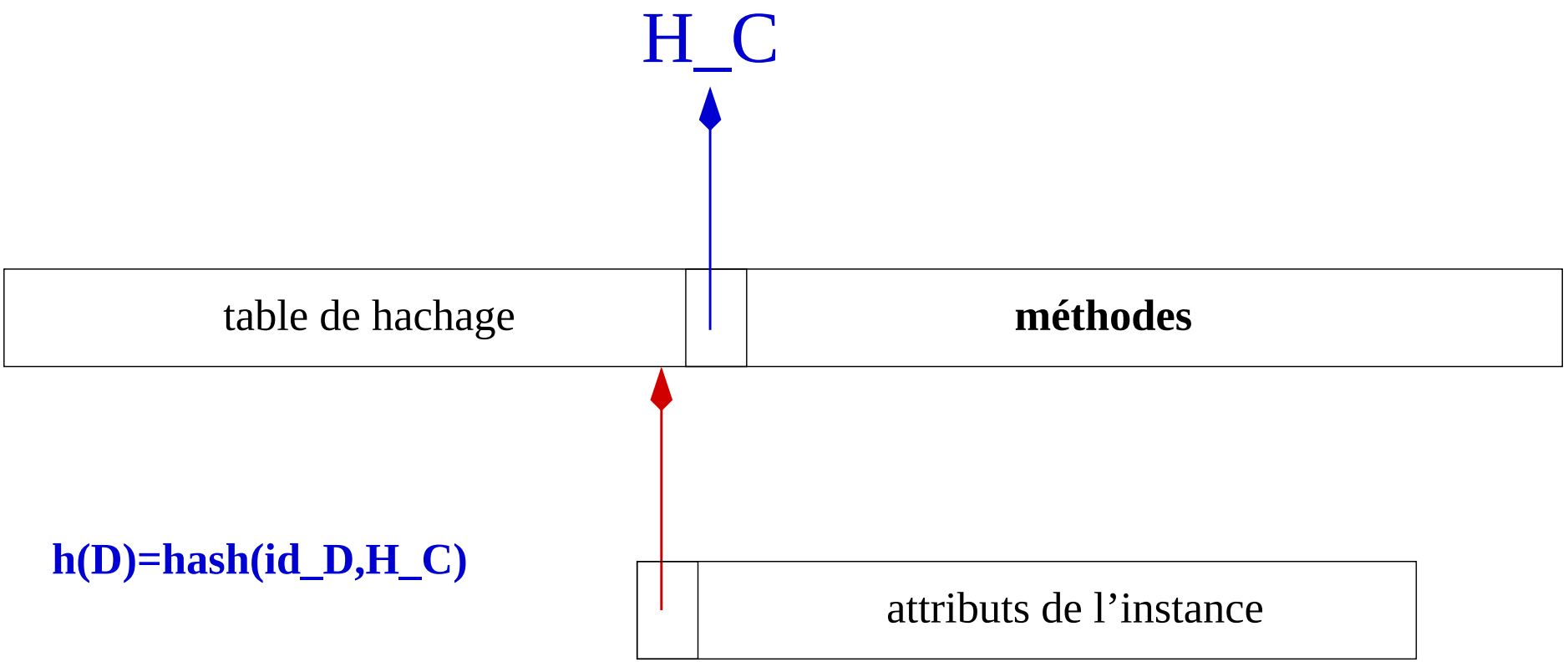


Table bidirectionnelle de la classe *C*

- Partie positive pour les classes, comme en héritage simple
- Partie négative pour les interfaces, avec hachage parfait

Application à JAVA (1 pierre, 2 coups)

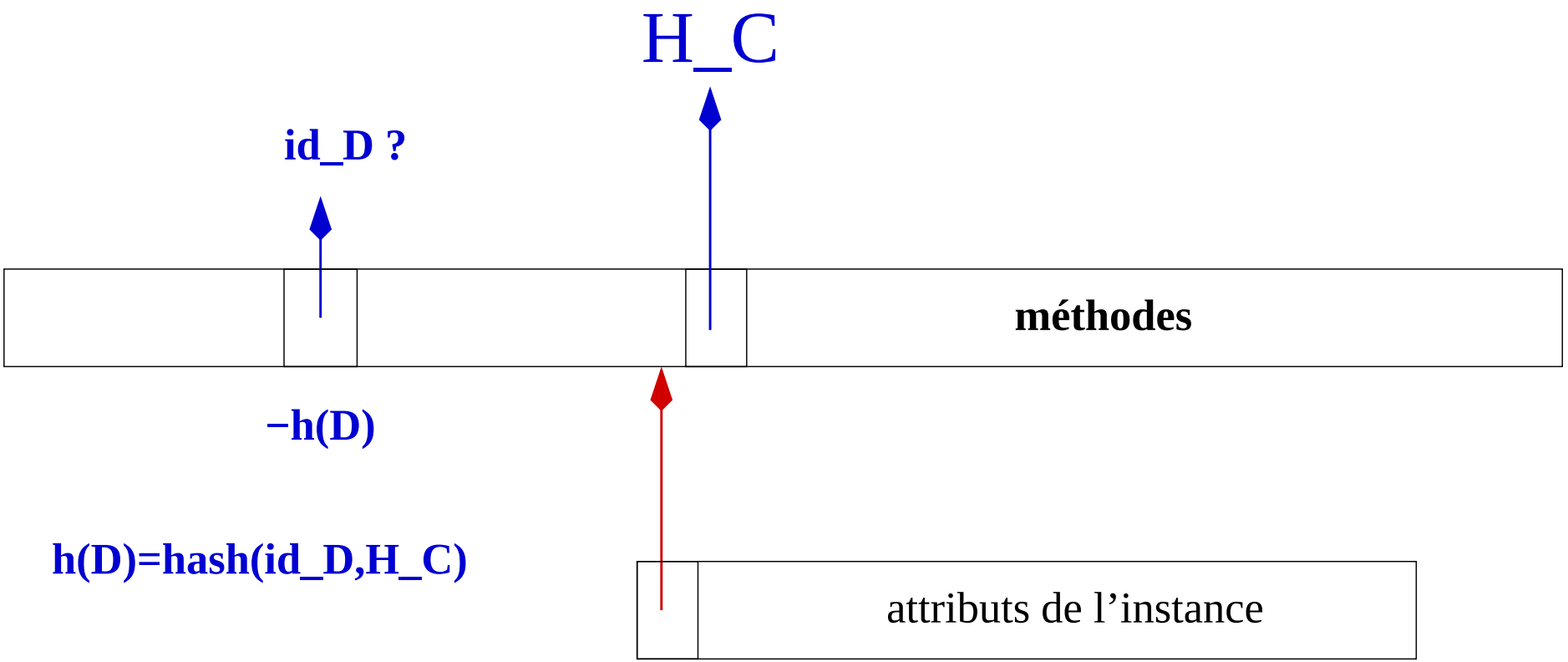


Table bidirectionnelle de la classe C

- Test de sous-typage pour interfaces
- D est une interface : $o \in D$?

Application à JAVA (1 pierre, 2 coups)

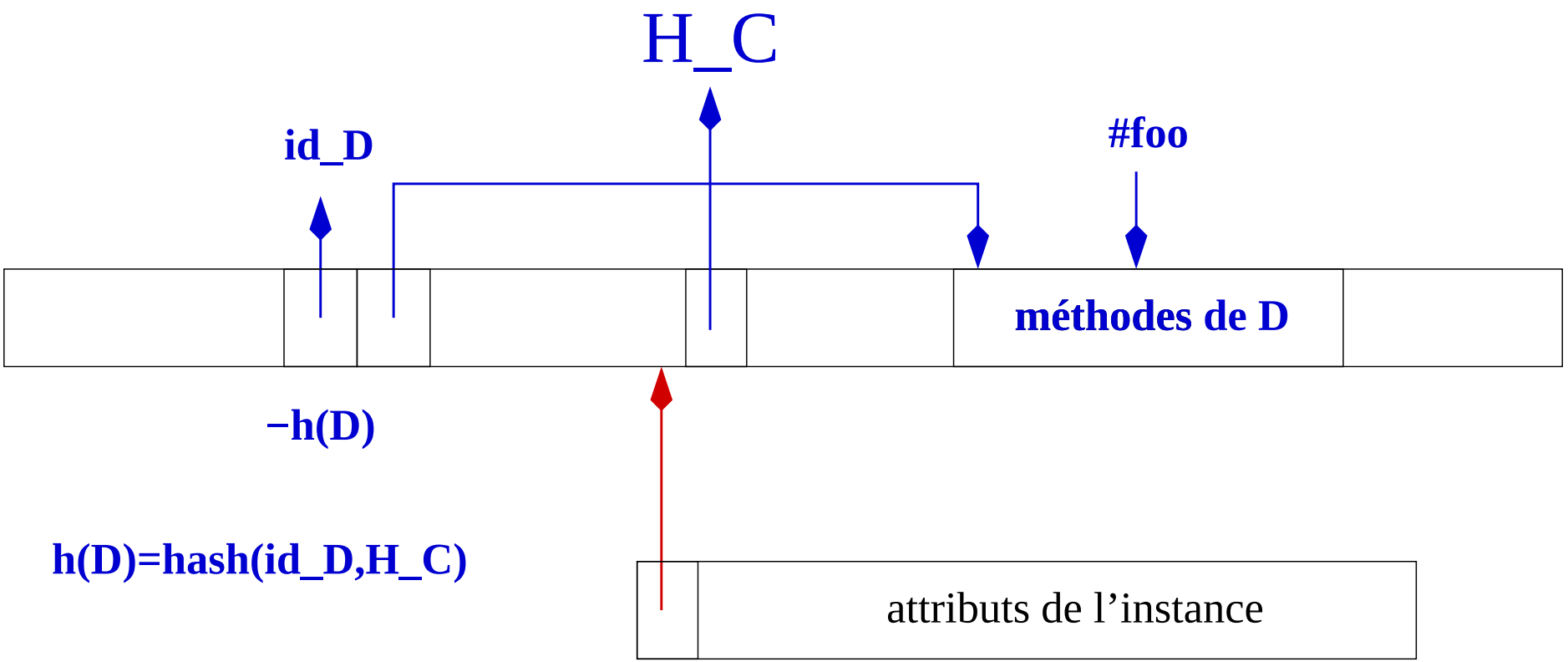


Table bidirectionnelle de la classe *C*

- Appel de méthode pour interfaces (`invokeinterface`)
- `D x; x.foo(arg)`

Hachage Parfait

Expérimentation avec des fonctions de hachage simples

- and et modulo
- complexité faible et algorithmes exacts simples
- Simulation sur les benchmarks précités
- Intrinsèquement incrémental

Conclusions mitigées

- trop grande tables avec and
- la division entière est trop lente !

Cherche fonction de hachage ...

- 2 opérations à 1 cycle,
- garantissant des tables compactes.

Compilation séparée et optimisation globale

Quadrature du cercle

- Compilation séparée : génération de code C, compilé séparément
- Optimisations globales à l'édition de liens

Compilation séparée et optimisation globale

Quadrature du cercle

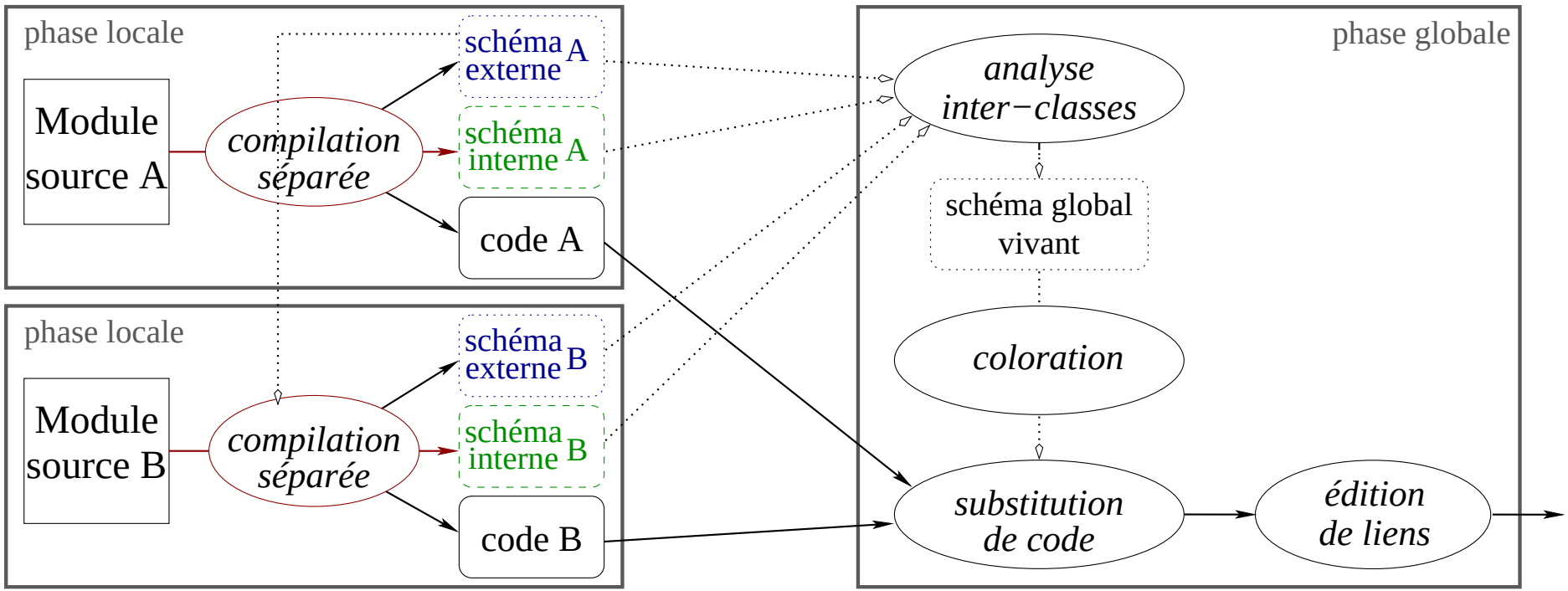
- Compilation séparée : génération de code C, compilé séparément
- Optimisations globales à l'édition de liens

Optimisations

- Analyse de types
- Invocation de méthodes :
  - Coloration pour les appels mégamorphes
  - Arbres de décision binaires pour les appels oligomorphes
  - Appel statique dans les cas monomorphes
- Elimination du code mort.

LIRMM

Schéma de Compilation



Le langage PRM

Un langage objet normal (donc unique !)

- statiquement typé
- avec héritage multiple basé ...
- ... sur un méta-modèle simple.

Le langage PRM

Un langage objet normal (donc unique !)

- statiquement typé
- avec héritage multiple basé ...
- ... sur un méta-modèle simple.

Une innovation majeure

- des modules
- avec du raffinement de classes

Le langage PRM

Un langage objet normal (donc unique !)

- statiquement typé
- avec héritage multiple basé ...
- ... sur un méta-modèle simple.

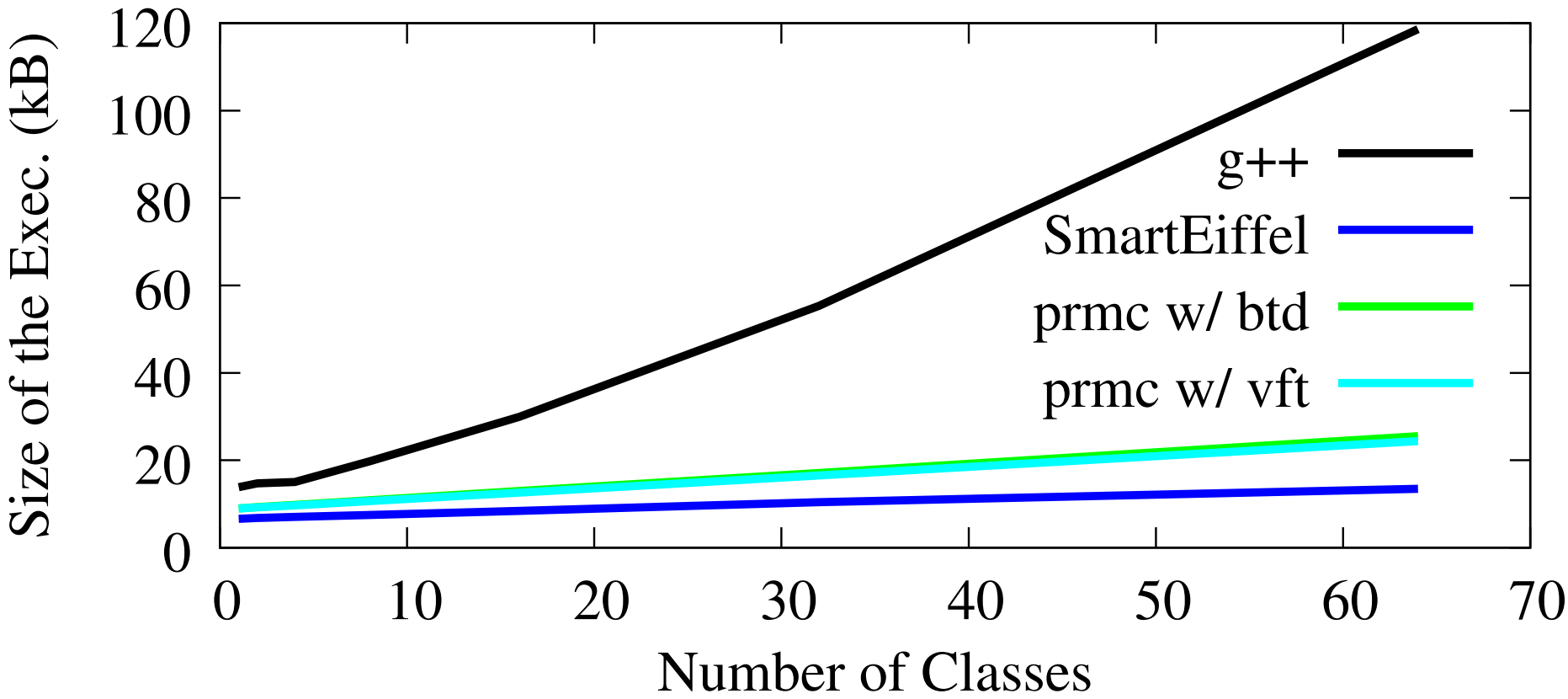
Une innovation majeure

- des modules
- avec du raffinement de classes

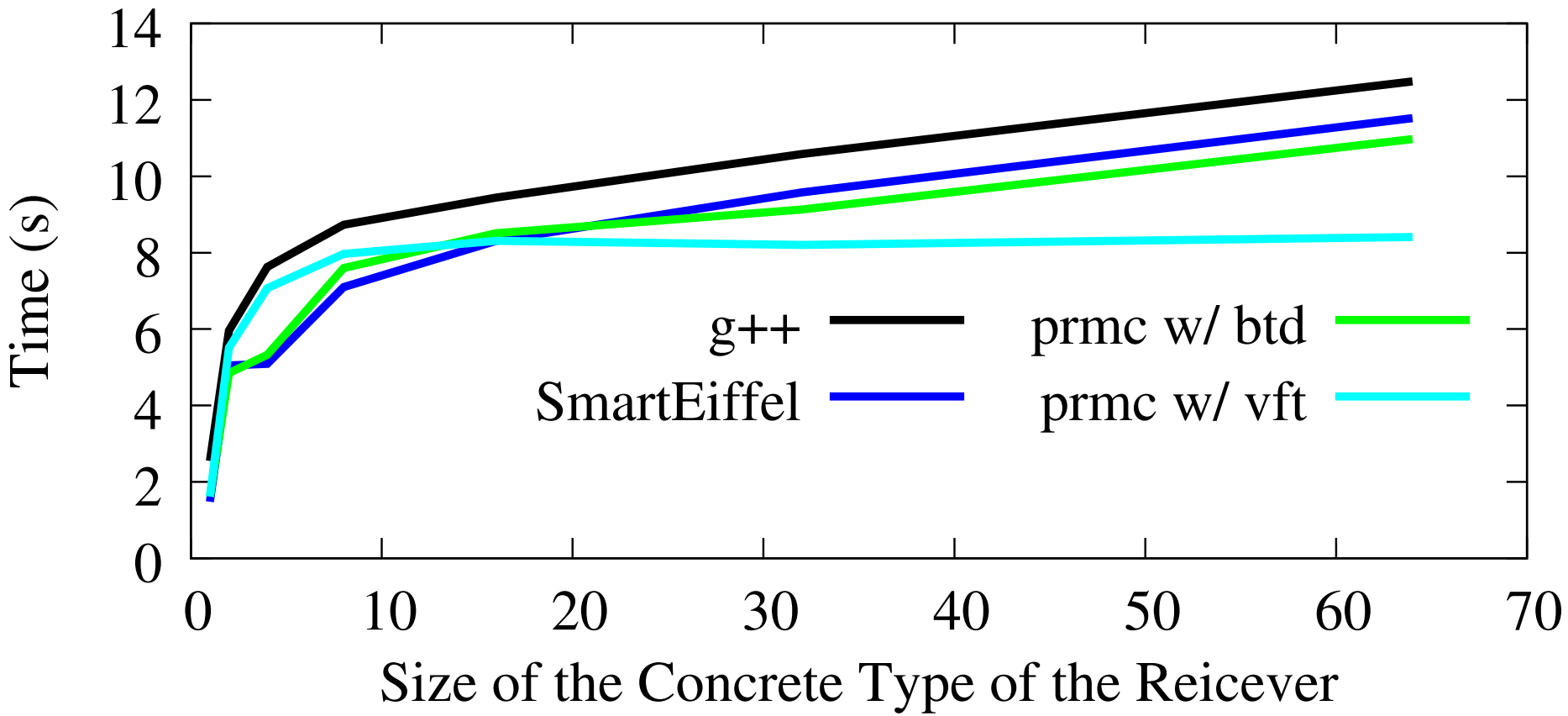
Un compilateur modulaire, PRMc

- architecture basée sur les modules
- et le raffinement de classes

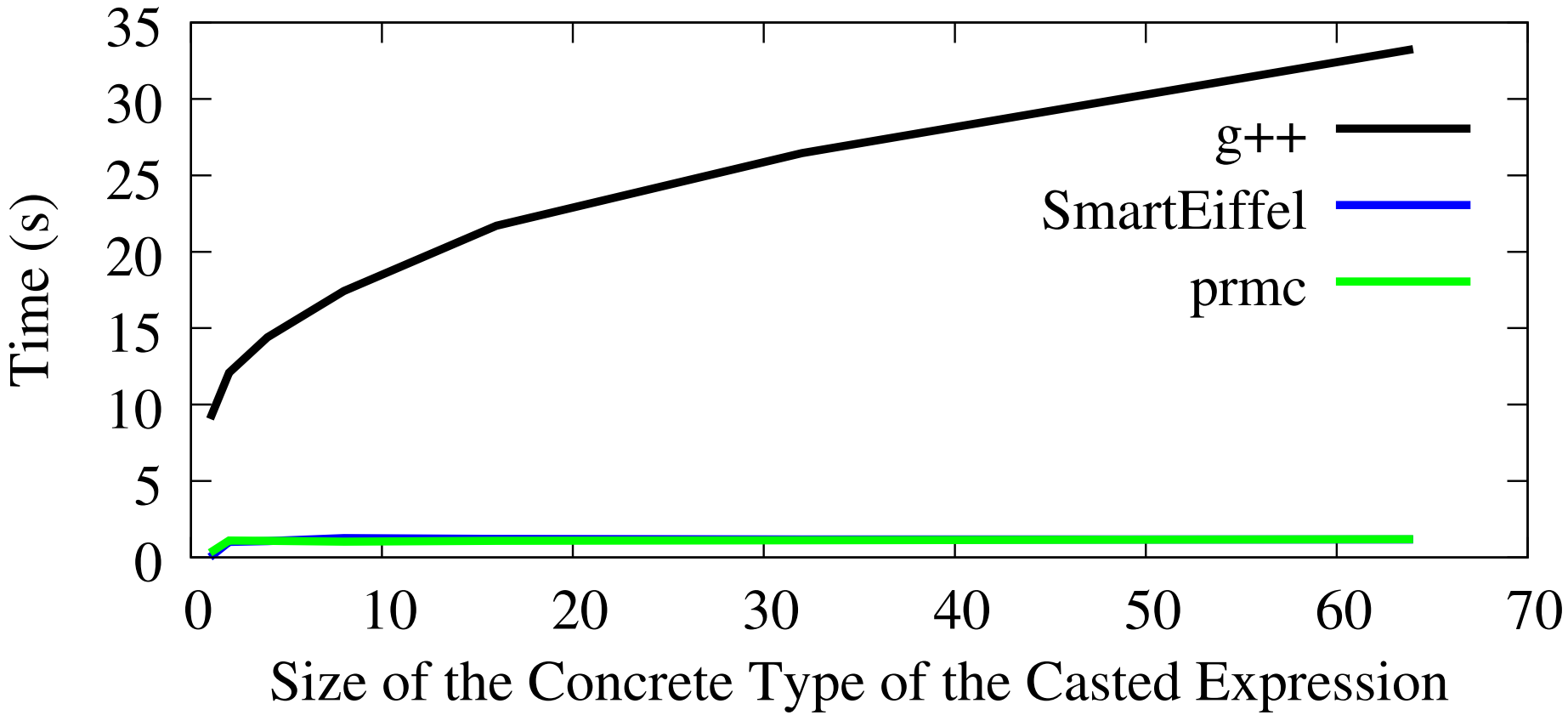
Benchmarks


Size of the Exec. (kB)
120
100
80
60
40
20
0
0
10
20
30
40
50
60
70
Number of Classes
g++
SmartEiffel
prmc w/ btd
prmc w/ vft

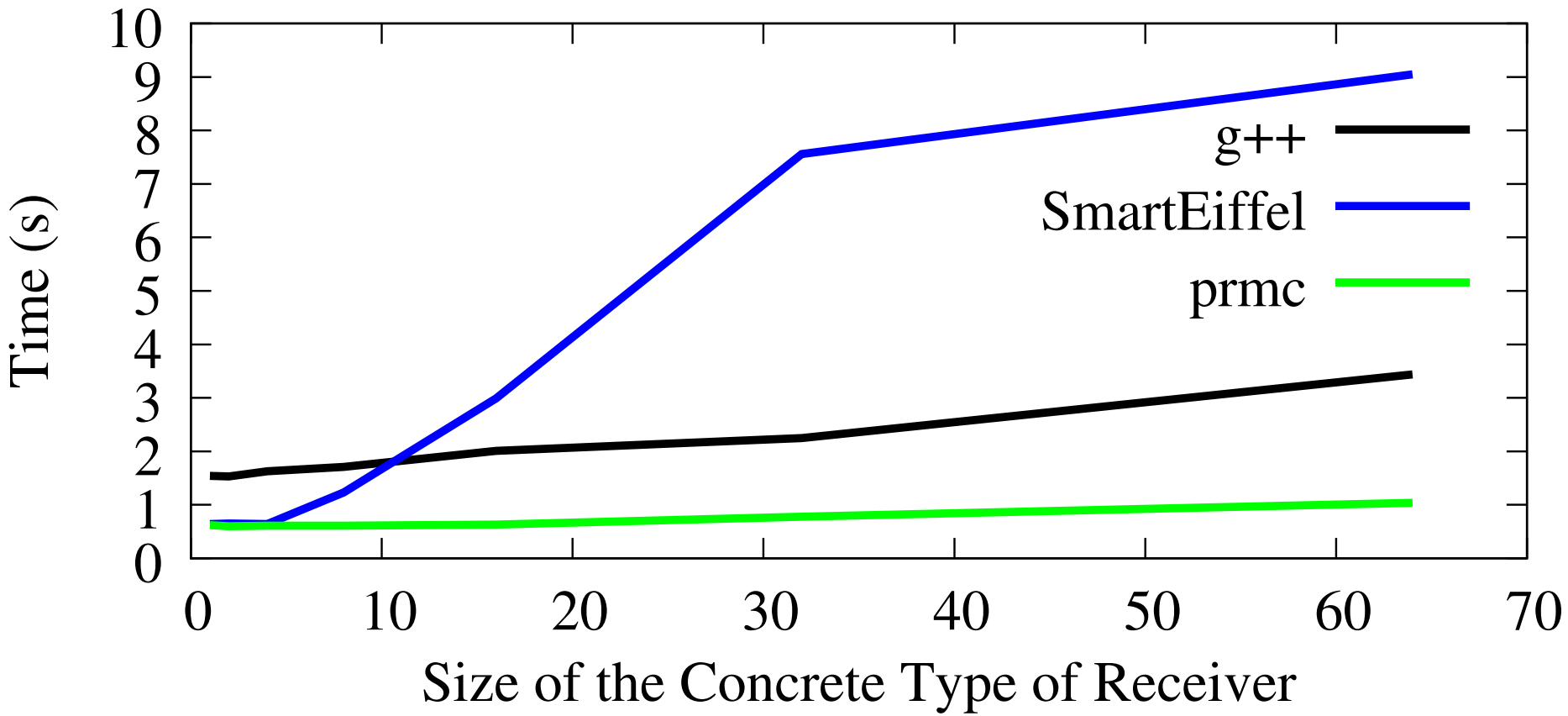
Benchmarks


Time (s)
0
2
4
6
8
10
12
14
0
10
20
30
40
50
60
70
Size of the Concrete Type of the Reicever
g++
SmartEiffel
prmc w/ btd
prmc w/ vft

Benchmarks


Time (s)
35
30
25
20
15
10
5
0
0
10
20
30
40
50
60
70
Size of the Concrete Type of the Casted Expression
g++
SmartEiffel
prmc

Benchmarks


Time (s)
0
1
2
3
4
5
6
7
8
9
10
0
10
20
30
40
50
60
70
Size of the Concrete Type of Receiver
g++
SmartEiffel
prmc

Perspectives

A court terme : Benchmarks sur PRMc

- autant de versions du compilateur que de techniques à tester ;
- autant de versions du compilateur que de versions du compilateur pour le compiler ;
- le vertige !

Perspectives

A court terme : Benchmarks sur PRMc

- autant de versions du compilateur que de techniques à tester ;
- autant de versions du compilateur que de versions du compilateur pour le compiler ;
- le vertige !

A moyen terme : Machine virtuelle PRM

- la coloration n'est pas incrémentale,
- le raffinement de classes non plus.
- Enfin un problème intéressant !

LIRMM

Ne pas céder aux sirènes de la spécialisation !

Besoin d'étudiants très polyvalents

- programmation, de C à YAFOOL et PRM, impérative, fonctionnelle, objet
- algorithmique pointue
- analyse syntaxique, compilation, interprétation
- architecture des machines
- proba-stats
- un zest d'Aristote et de Spinoza

Ne pas céder aux sirènes de la spécialisation !

Besoin d'étudiants très polyvalents

- programmation, de C à YAFOOL et PRM, impérative, fonctionnelle, objet
- algorithmique pointue
- analyse syntaxique, compilation, interprétation
- architecture des machines
- proba-stats
- un zest d'Aristote et de Spinoza

Durée de thèse

Sur un tel sujet, en partant de rien, 4 ans c'est extrêmement court !

Publications

Difficile et long !

- 180 pages actuellement soumises :
  - ACM Computing Surveys (soumis 2002, révisé 2005, ...)
  - ACM Trans. on Progr. Lang. Syst. (soumis 2005, révisé 2006)
- 4 refus successifs à ECOOP et OOPSLA !
- 1 workshop (conférence) ACM PASTE
- 1 workshop (workshop) à ECOOP
- 4 LMO (francophone)

Commentaires

- revues : liberté de soumission mais délais et longueurs
  papier jamais assez autosuffisant
- ECOOP/OOPSLA : trop de sélection tue

Plan

Le vicomte de Bragelonne (© A. Dumas)

Le règne de PRM

- tout le monde programme en PRM
- la hot-line sonne sans arrêt
- l'âge de la retraite est repoussé tous les siècles
- on s'ennuie ...

```
import string
import file
println("That's all folks")
```

d'après A. Dumas, dessins F'murr.

```
import string
import file
println("That's all folks")
```

d'après A. Dumas, dessins F'murr.