La coloration : une technique universelle d'implémentation des objets

Roland Ducournau
LIRMM – Université Montpellier 2

29 janvier 2002

Plan

1. Le paradis : l'héritage simple
2. L'enfer : l'héritage multiple
3. Le purgatoire : la coloration
   - coloration de méthodes
   - coloration de classes
   - coloration d'attributs ?
   - problème et variations
   - coloration en compilation séparée
4. Historique et comparaisons
5. Perspectives : une vie nouvelle

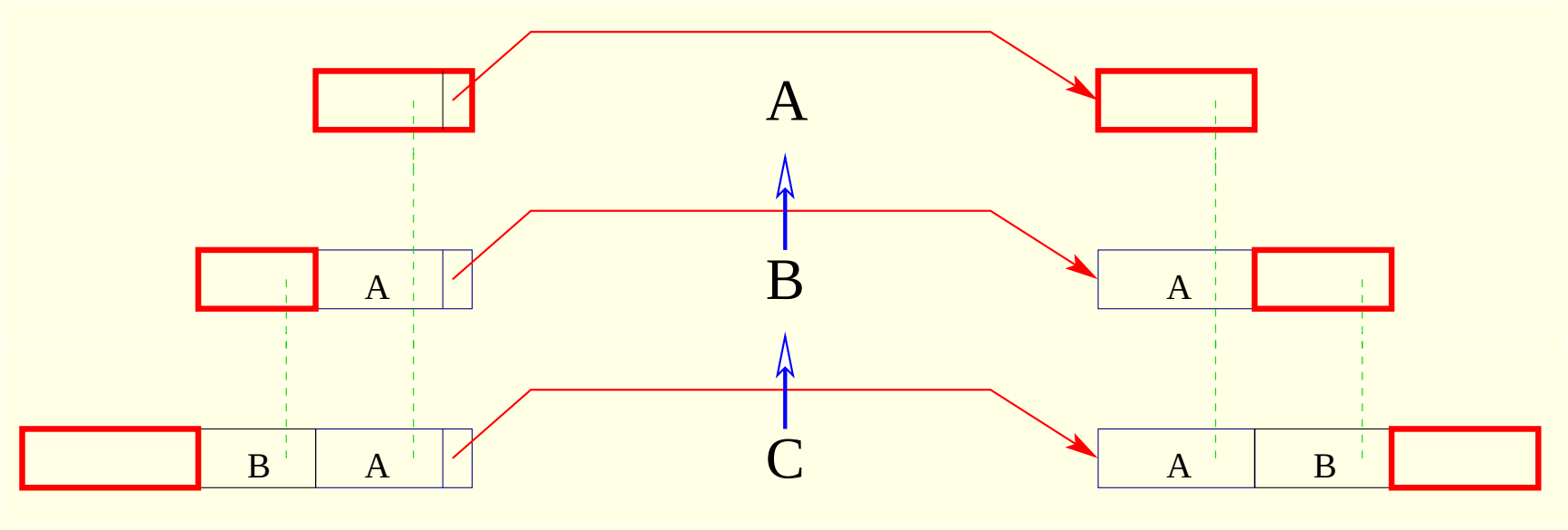
Le paradis : l'héritage simple

En typage statique,
double invariant par rapport aux types statiques :

- des indices des attributs ou méthodes
  dans les tables implémentant les objets ;
- des références aux objets elles-mêmes.

mémoire dynamique

mémoire statique



instances

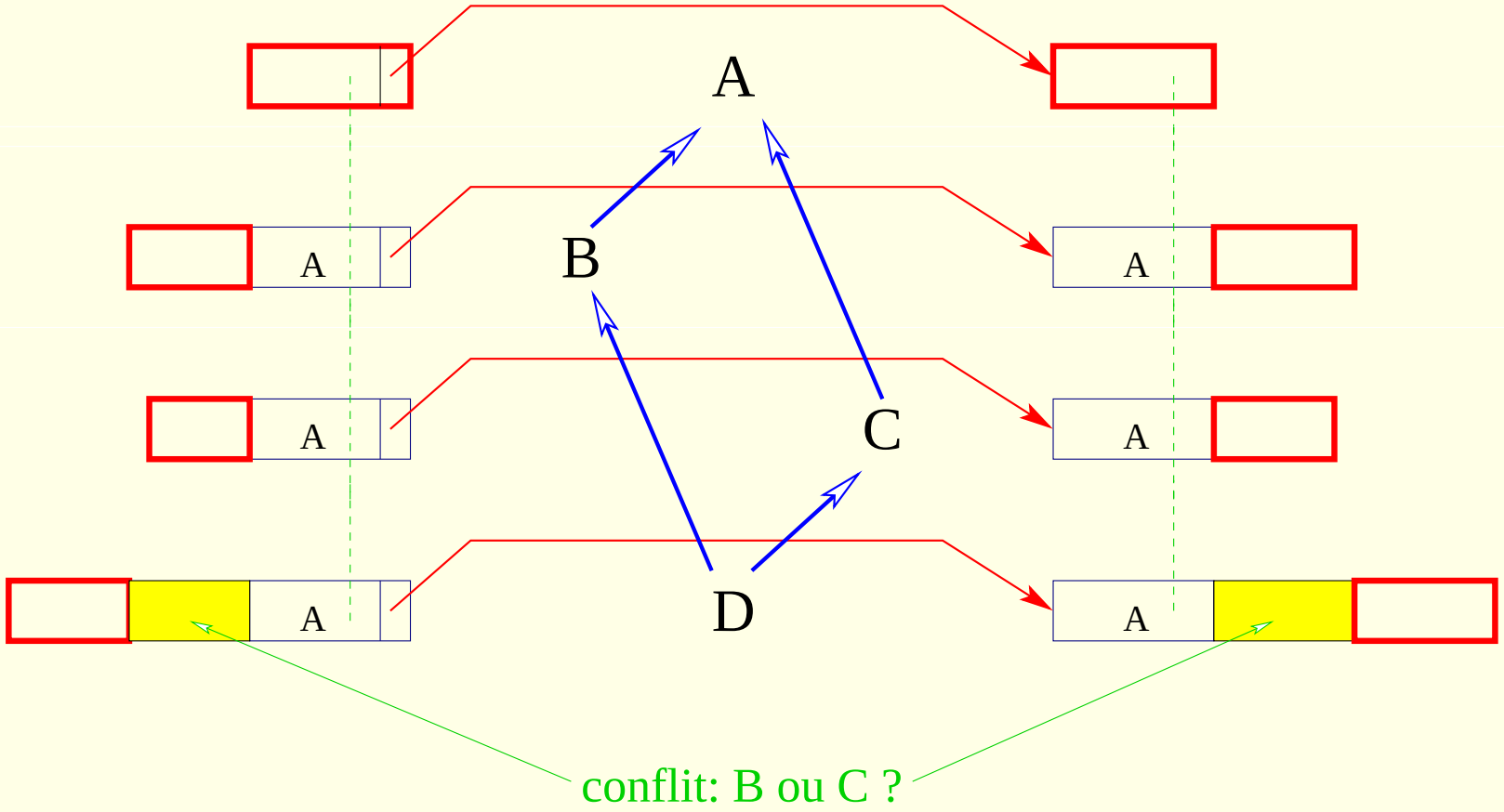
tables de méthodes

L'enfer : l'héritage multiple

En compilation séparée,
double anti-invariant par rapport aux types statiques :

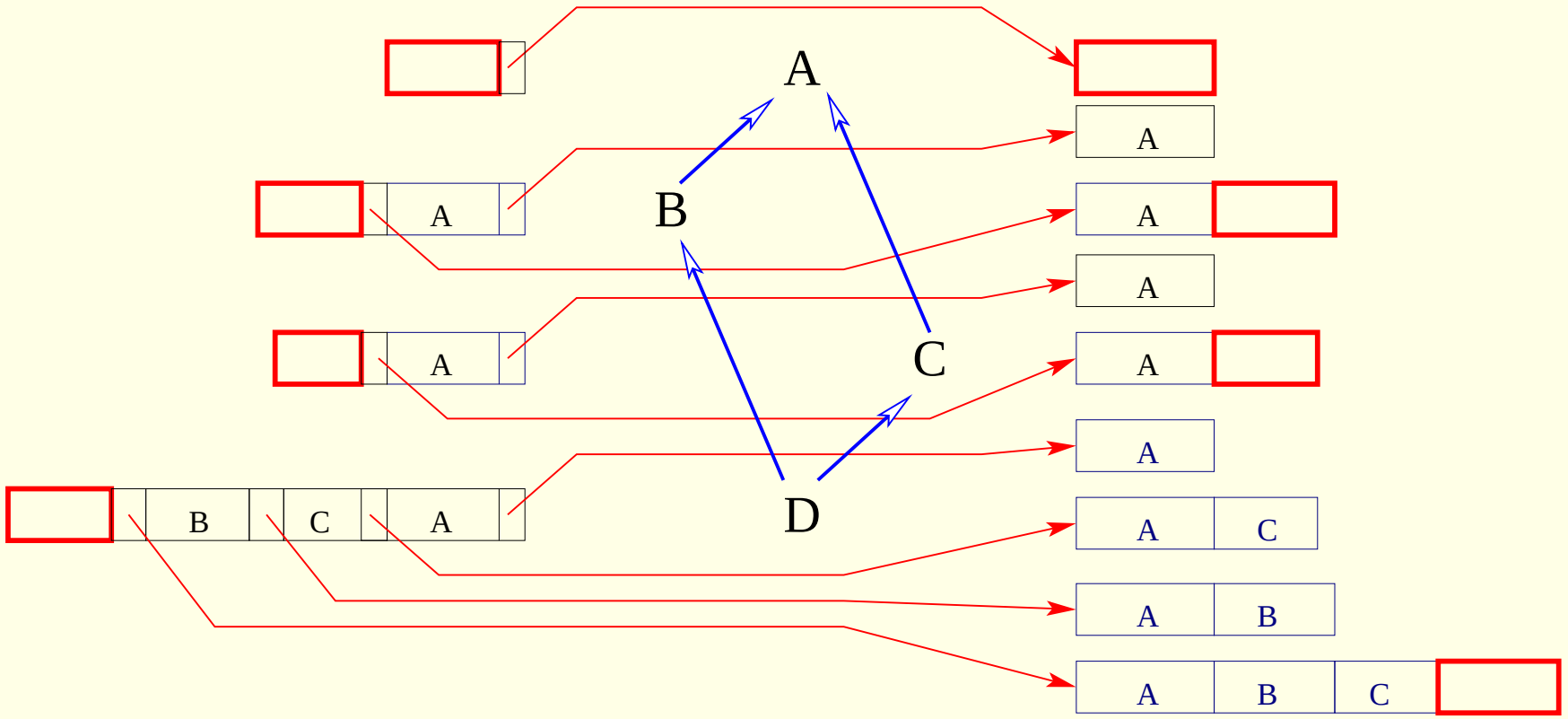
- les indices des attributs ou méthodes dépendent des types statiques ;
- les références aux objets aussi.

mémoire dynamique mémoire statique



mémoire dynamique

mémoire statique



Inconvénients :

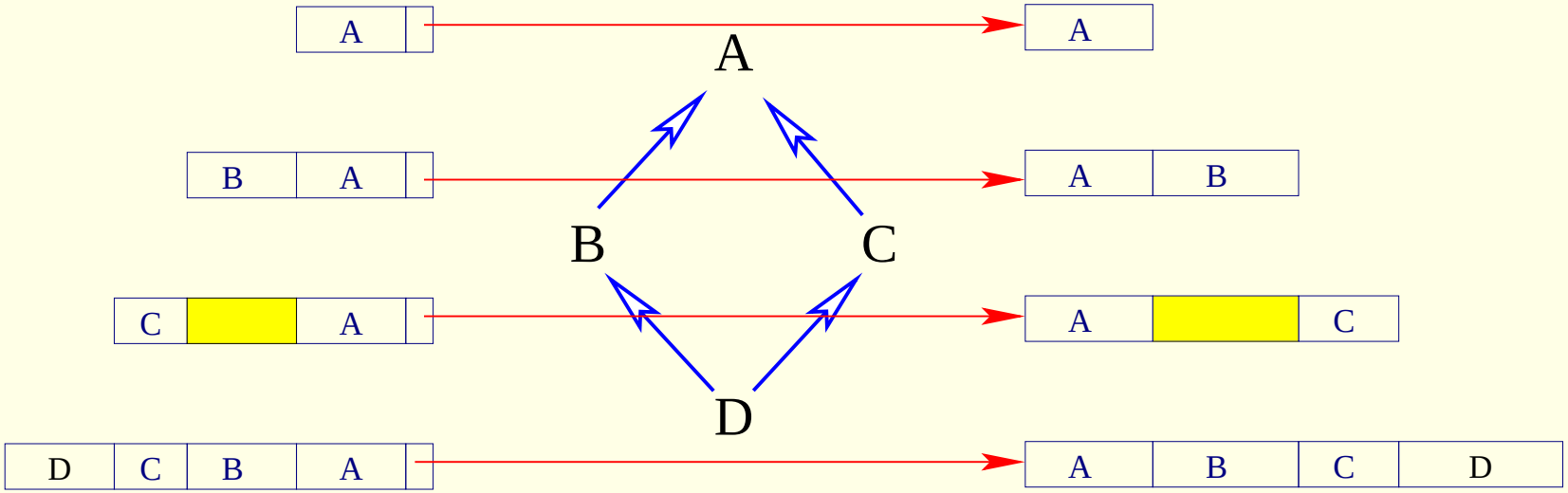
- **temporels :** indirections perpétuelles
  - accès aux attributs,
  - affectations et passages de paramètres qui ne sont pas à type statique constant,
  - ajustement du receveur dans les envois de messages.
- **mémoire statique :**
  - nombre de tables quadratique dans le nombre de classes,
  - taille des tables cubique, doublée par les ajustements,
  - taille du code doublée par les indirections,
- **mémoire dynamique :**
  - taille des objets augmentée des pointeurs sur les tables,
- **sémantique dégradée :** héritage non virtuel de C++ ;
- **performance dégradée** même en héritage simple.

Le purgatoire : la coloration

La coloration est le problème du maintien des invariants
de l'héritage simple en héritage multiple.

- avantages :
  - efficacité et simplicité de l'héritage simple ;
  - technique « universelle » : méthodes, classes, attributs ;
- inconvénients :
  - technique globale ;
  - existence de trous dans les tables ;
  - algorithmique difficile.

Le principe de la coloration



La coloration de classes

- réduction du problème de coloration de méthodes
  au cas où chaque classe introduit une méthode unique ;
- chaque classe C a
  - une couleur $k(C)$,
  - un identifiant $id(C)$,
  - une table de couleurs tab_C ;
- test de sous-type :

$$D \preceq C \Leftrightarrow tab_D[k(C)] = id(C)$$

La coloration d'attributs ?

Faut-il l'utiliser ?

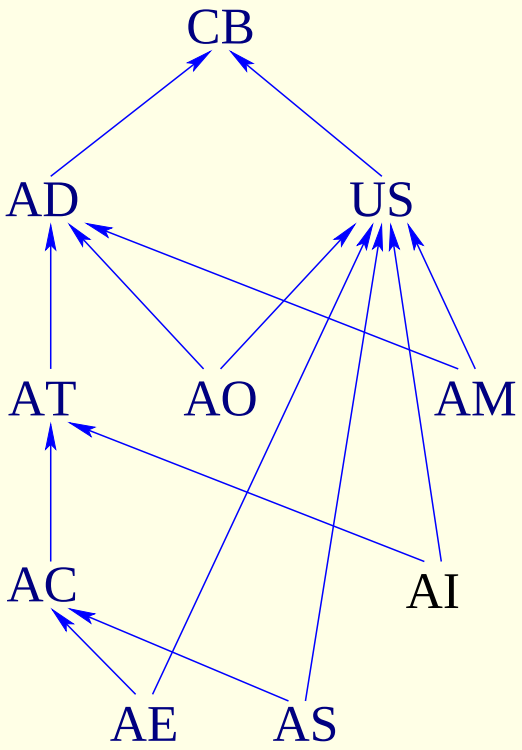
- inconvénient majeur :
  - trous en mémoire dynamique, donc en millions d'exemplaires ;
- solution alternative :
  - implémentation par décalages, avec coloration des décalages ;
- nouvel inconvénient :
  - décalages systématiques, pire qu'en héritage multiple standard ;
- solution :
  - double compilation avec ou sans décalage sur self
    et choix de la version à l'édition de liens.
- conclusion :
  - aussi efficace en héritage simple que l'héritage simple standard.

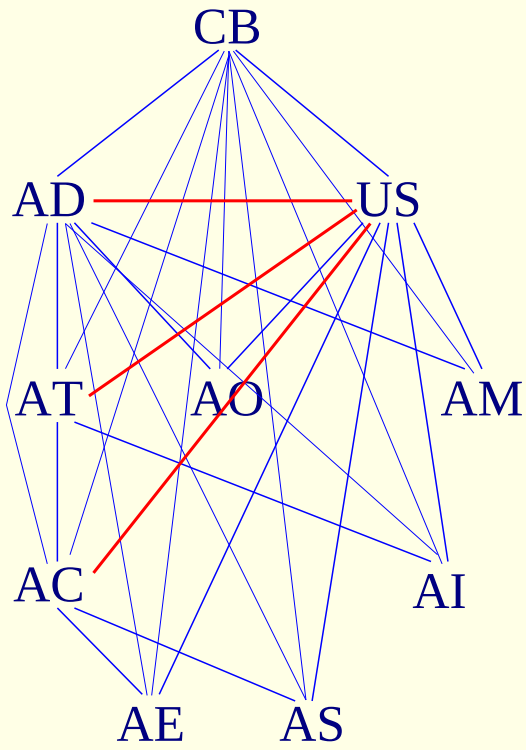
Coloration de classes : problème et variations

- coloration de graphe : union
  - du graphe de comparabilité
  - du graphe de conflit (incomparabilité avec sous-classes communes) ;
- problème d'optimisation : minimisation
  - du nombre de couleurs (NP-difficile) ;
  - de la taille des tables
    - coloration unidirectionnelle,
    - coloration bidirectionnelle.

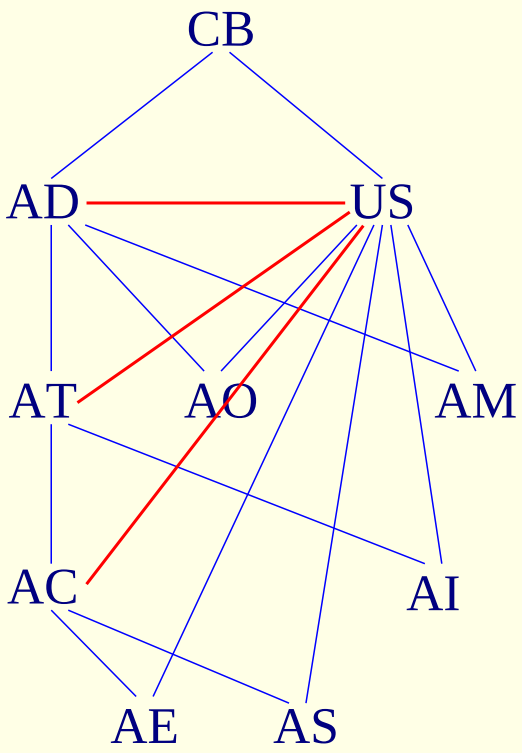
    classe de complexité inconnue : probablement NP-difficile.
- donc solutions heuristiques

Coloration de classes : exemple



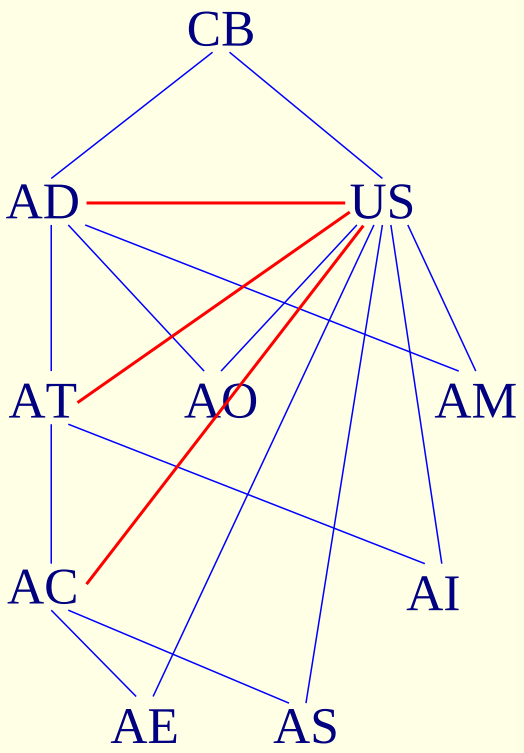


Minimisation du nombre de couleurs



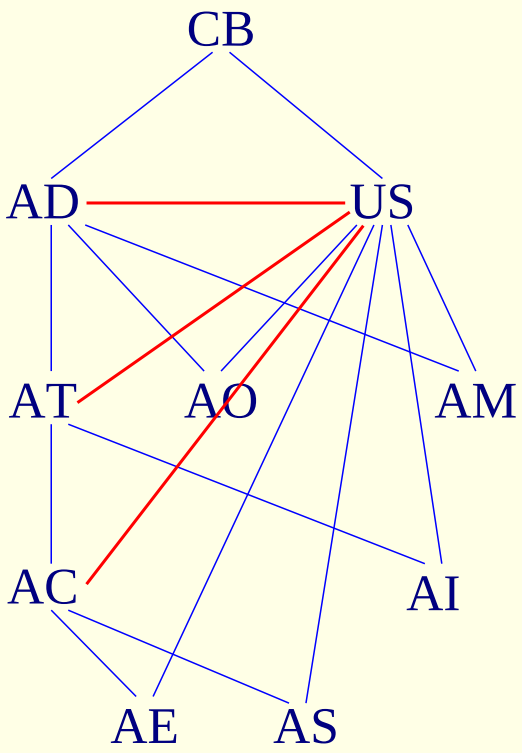
CB					
CB	AD				
CB		US			
CB	AD		AT		
CB	AD		AT	AC	
CB	AD	US	AO		
CB	AD	US	AM		
CB	AD	US	AT	AC	AE
CB	AD	US	AT	AC	AS
CB	AD	US	AT	AI	
0	1	2	3	4	5

Coloration unidirectionnelle



CB					
CB	AD				
CB		US			
CB	AD		AT		
CB	AD		AT	AC	
CB	AD	US	AO		
CB	AD	US	AM		
CB	AD	US	AT	AC	AE
CB	AD	US	AT	AC	AS
CB	AD	US	AT	AI	
0	1	2	3	4	5

Coloration bidirectionnelle



	CB				
	CB	AD			
US	CB				
	CB	AD	AT		
	CB	AD	AT	AC	
US	CB	AD	AO		
US	CB	AD	AM		
US	CB	AD	AT	AC	AE
US	CB	AD	AT	AC	AS
US	CB	AD	AT	AI	
-1	0	1	2	3	4

La coloration en compilation séparée

- la coloration est une technique globale :
  elle nécessite de connaître toutes les classes ;
- mais elle est calculable à l'édition de liens ;
- le code généré par la compilation séparée ne dépend de la coloration que par les couleurs, des constantes calculées par la coloration ;
- ces constantes pourraient être résolues par l'édition de liens,
  de la même façon que la résolution d'adresses.
- solution alternative :
  compilation dans un langage intermédiaire (C, machine virtuelle).

Historique de la coloration

- introduite simultanément par [Dixon etal 89], [Pugh et Weddel 90] et [Ducournau 89-91] en YAFOOL ;
- expérimentation systématique par [Pugh et Weddel 90], [André et Royer 92] et [Ducournau 97 et 01] ;
- appliquée :
  - aux méthodes par [Dixon etal 89] et [André et Royer 92] ;
  - aux attributs par [Pugh et Weddel 90] et [Ducournau 89-91] ;
  - aux classes par [Vitek etal 97] et [Ducournau 01] ;
- avec un critère de minimisation :
  - du nombre de couleurs pour [Dixon etal 89] et [André et Royer 92] ;
  - de la taille des tables pour [Ducournau 89-91] ;
  - en coloration bidirectionnelle pour [Pugh et Weddel 90] ;
- et une utilisation à l'édition de liens [Pugh et Weddel 90].

Historique de la coloration

L'essentiel était dans [Pugh et Weddel 90] mais personne ne l'a lu !

Comparaisons : techniques d'implémentation sans tables

... à la SMALL EIFFEL [Colnet et Zendra, 97sq]

L'architecture des processeurs entraîne :

- la parallélisation des indirections,
  - le surcoût de l'héritage multiple serait annulé,
  - mais les expérimentations de K. Driesen donnent un avantage aux thunks (contradictoire ?) ;
- des prédictions de branchements conditionnels,
  - qui donnent l'avantage aux arbres de décision,
- bientôt des prédictions de branchements calculés.
  - qui ramèneront les tables à égalité.

Conclusion

Un schéma d'implémentation de l'héritage multiple aussi efficace que l'héritage simple :

- coloration bidirectionnelles des méthodes ;
- coloration bidirectionnelle des classes :
  - pour le test de sous-type ;
  - pour les décalages pour les accès aux attributs ;
- double compilation pour les accès aux attributs de self [Myers 95] ;
- mise en œuvre à l'édition de liens.
- les diverses expérimentations [Pugh et Weddel 90], [Ducournau 89-91, 97 et 01] démontrent :
  - une très bonne compacité (taux de trou < 50 %) : à comparer aux nombre de tables quadratique de l'implémentation standard ;
  - des temps de calcul raisonnables.

Perspectives

- poursuite de l'étude algorithmique :
  extension des résultats de la coloration de classes
  à la coloration de méthodes ;
- application de la coloration en vraie grandeur :
  détournement d'un compilateur existant ;
- application des techniques globales en compilation séparée :
  analyse de types et détection du code mort.