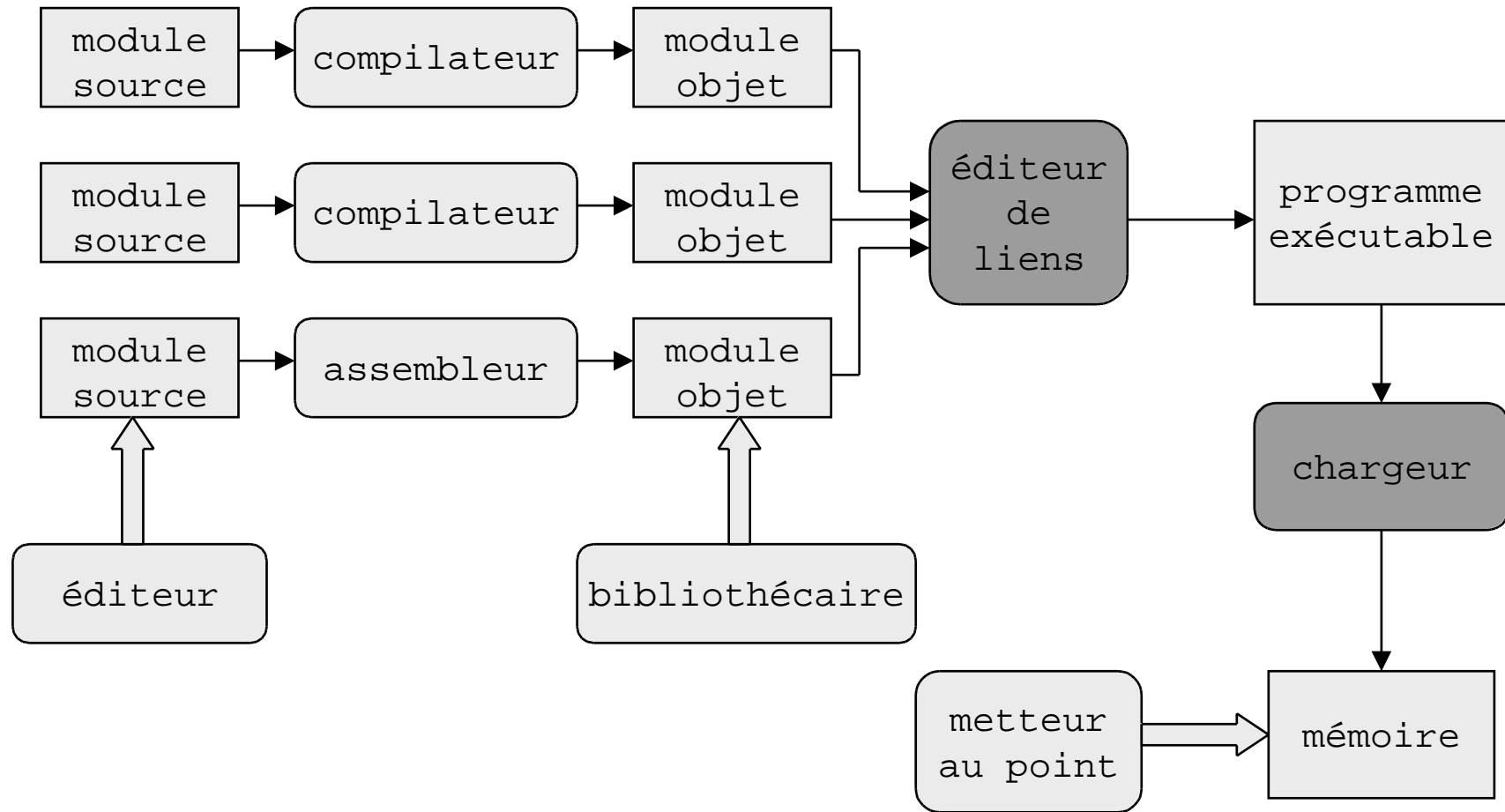


L'édition de liens et le chargement



Notion de module translatable (1)

- Emplacements disjoints des modules à l'exécution
- Architecture matérielle (Multics, iAPX432, etc.)
 - adresse = couple $\langle n, d \rangle$
 - n désigne l'espace
 - d le déplacement dans l'espace
- En général => pouvoir mettre les modules n'importe où
 - sections différentes: code, données constantes, variables
 - indiquer la nature de la section
 - indiquer les emplacements contenant des adresses relatives à une section
 - => ajouter à chacun de ces emplacements l'adresse de début de la section correspondante

Notion de module translatable (2)

```
AJOUT10: move.w #10,D0    0: 30 3C 00 0A      012340: 30 3C 00 0A
          jmp      C      4: 4E F9 00 00 00 0E 012344: 4E F9 00 01 23 4E
AJOUT16: move.w #16,D0   A: 30 3C 00 10      01234A: 30 3C 00 10
C:       add.w   D1,D0    E: D1 01      01234E: D1 01
          move.w D0,MEMO 10: 33 C0 00 00 00 02 012350: 33 C0 00 02 33 22
          rts      16: 2E 75      012356: 2E 75
          .data
LOC:     ds.w    0: 00 00      023320: 00 00
MEMO:    ds.w   1  2: 00 00      023322: 00 00
```

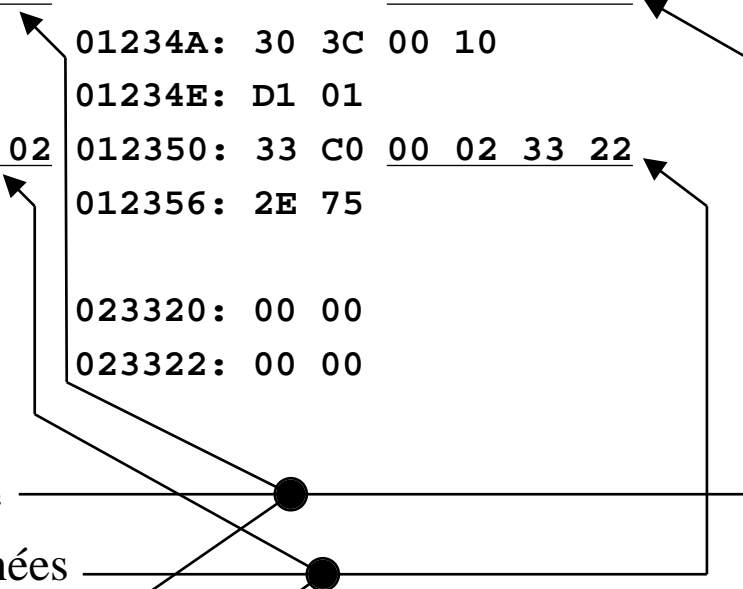
informations de translation:

6 section code: adresse 4 octets relative code

12 section code: adresse 4 octets relative données

section code placée en: 012340

section donnée placée en: 023320



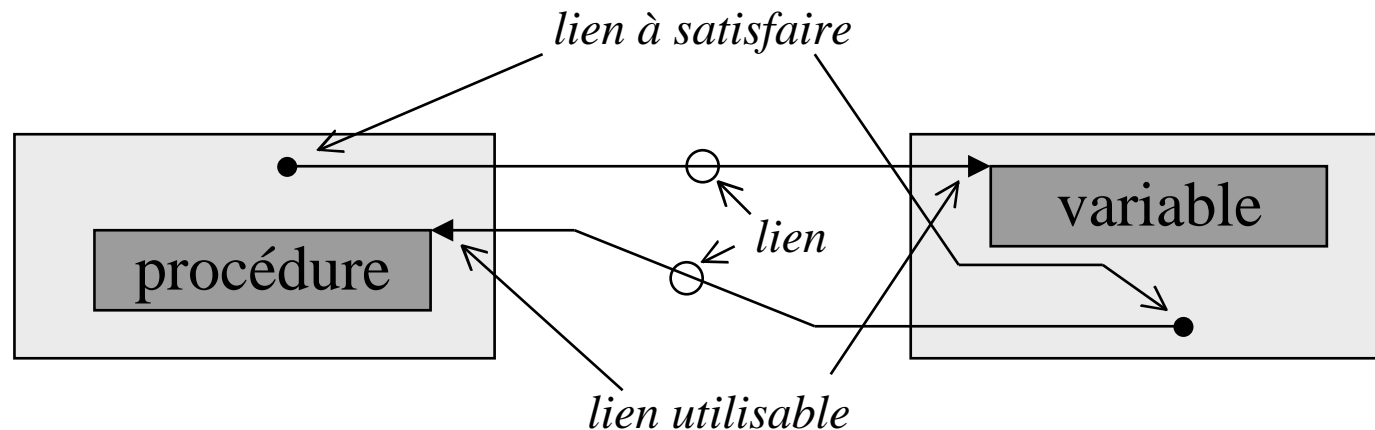
Notion de lien

- intermédiaire: *nom* = identificateur du texte source
- catégorie des objets (implicite ou explicite):

internes => aucun lien

exportés ou *publics* => lien utilisable

importés ou *externes* => lien à satisfaire



Exemples d'expression du lien

Pascal Multics 6.02

```
program P(A)
```

```
var def B: integer;
```

```
procedure A(x:integer);
```

```
    pascal;
```

```
begin
```

```
    ...
```

```
end.
```

Pascal Multics (après)

```
program P;
```

```
$IMPORT 'A(pascal)':A$
```

```
$EXPORT B$
```

```
var B: integer;
```

```
procedure A(x:integer);
```

```
    external;
```

```
begin
```

```
    ...
```

```
end.
```

Pascal microméga

```
program P;
```

```
var B: integer;
```

```
procedure A(x:integer);
```

```
    external;
```

```
begin
```

```
    ...
```

```
end.
```

Ada

```
package E is
```

```
    procedure A(x:integer);
```

```
end E;
```

```
package P is
```

```
    B: integer;
```

```
end P;
```

```
with E;
```

```
package body P is
```

```
    -- contenu de P
```

```
end P;
```

Représentation des liens

- lien utilisable: $\langle nom, section, adresse\ relative \rangle$
- lien à satisfaire

$\langle nom, liste\ d'emplacement \rangle$

chaînage des emplacements

```
tant que liste <> nil faire
  x := contenu[liste];
  contenu[liste] := valeur;
  liste := x;
fait;
```

tableau des emplacements

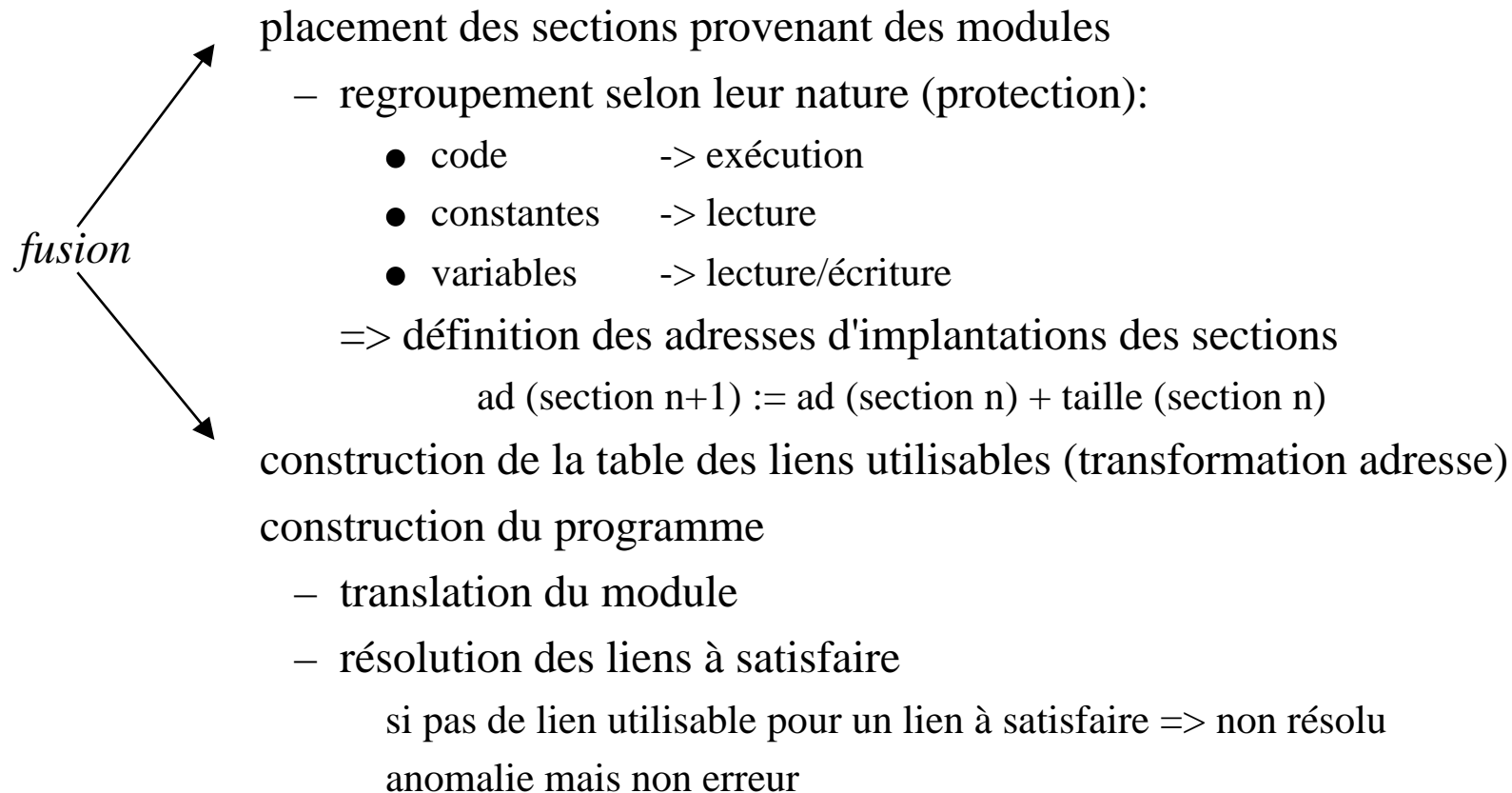
```
pour i de 1 à n faire
  contenu[liste[i]] :=
    contenu[liste[i]]
    + valeur;
fait;
```

$\langle nom, numéro \rangle$

informations de translation: ajouter à l'emplacement e la valeur du lien n

Édition simple (liste de modules)

● Trois étapes



Construction de la table des liens

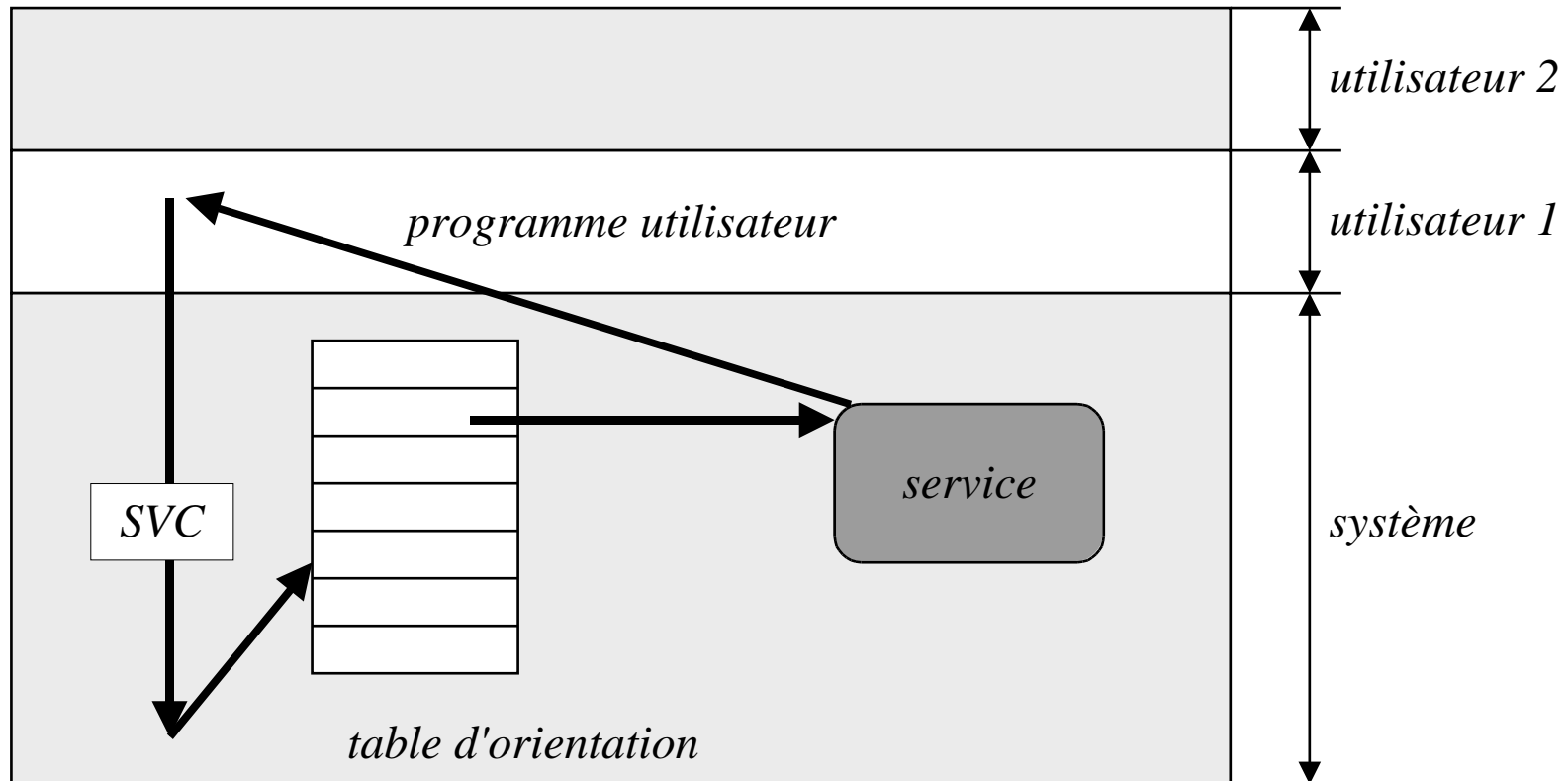
- LU_M liens utilisables du module M
 - LAS_M liens utilisables du module M
 - $LU_M \quad LU_{M'}$ doubles définitions
 - $LAS_L = \bigcup_{M \in L} LAS_M - \bigcup_{M \in L} LU_M$ liens non résolus dans L
 - étape 2: construire la table de tous les liens, avec leur état
- rencontre d'un lien utilisable dans M , déjà à l'état utilisable dans la table*
- état à_satisfaire à la fin*

nom	état	valeur
A	à_satisfaire	
B	utilisable	3281

Bibliothèque de modules (nature)

- bibliothèque de calcul
 - sous programme standard: sinus, cosinus, racine_carrée, etc.
 - sous programmes liés à un langage
- bibliothèque d'interface système
 - sous programmes exécutant les appels systèmes
 - préparent les paramètres, et exécutent l'appel
 - appel système traité différemment d'un appel de sous programme:
 - contrôles incontournables
 - pas d'établissement de liaison par l'éditeur de liens (indépendance)
 - instruction particulière permettant le changement de mode
- bibliothèque d'utilisateur
 - sous programmes pour des besoins spécifiques

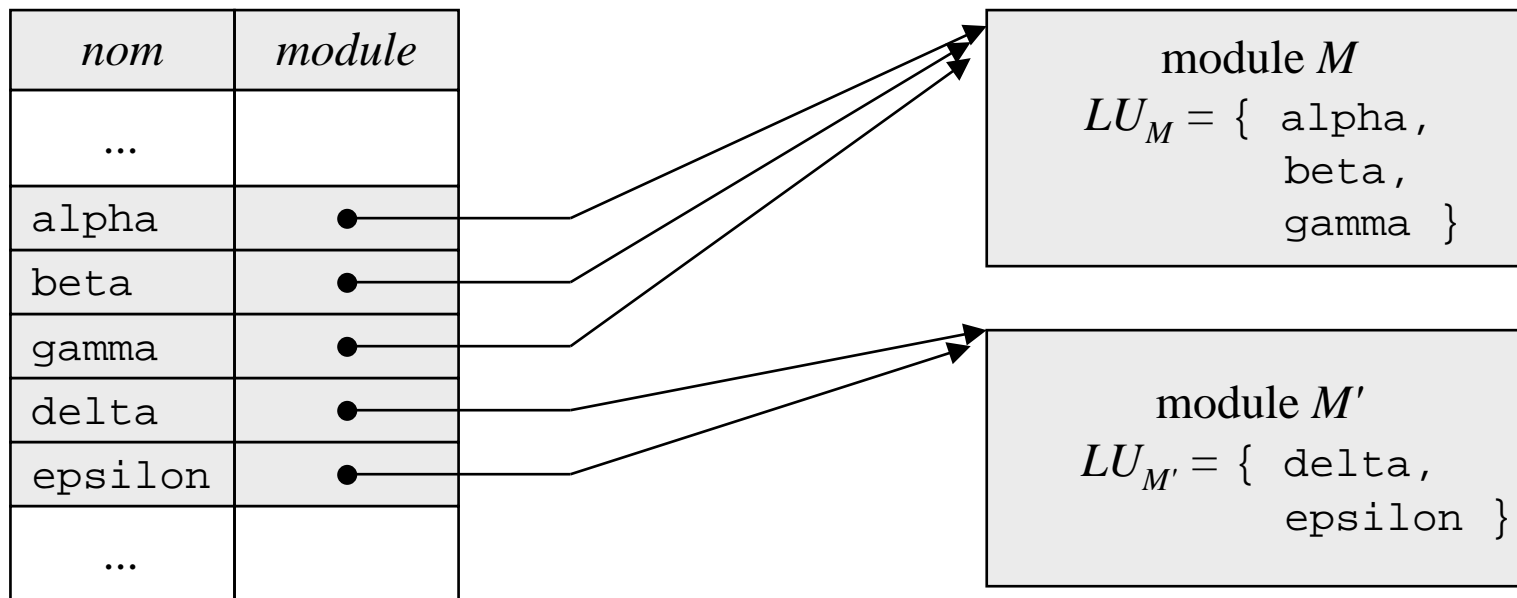
L'appel système



changement mode maître-esclave
contrôle des paramètres
pas de liaisons par l'EdL

INT sur IAPX ?86
TRAP sur M68K

Bibliothèque de modules (structure)



en général, on a: $M, M' \in B, LU_M \cap LU_{M'} =$

utilisation d'une liste de bibliothèque B_1, B_2, \dots, B_p

Adjonction de modules (1)

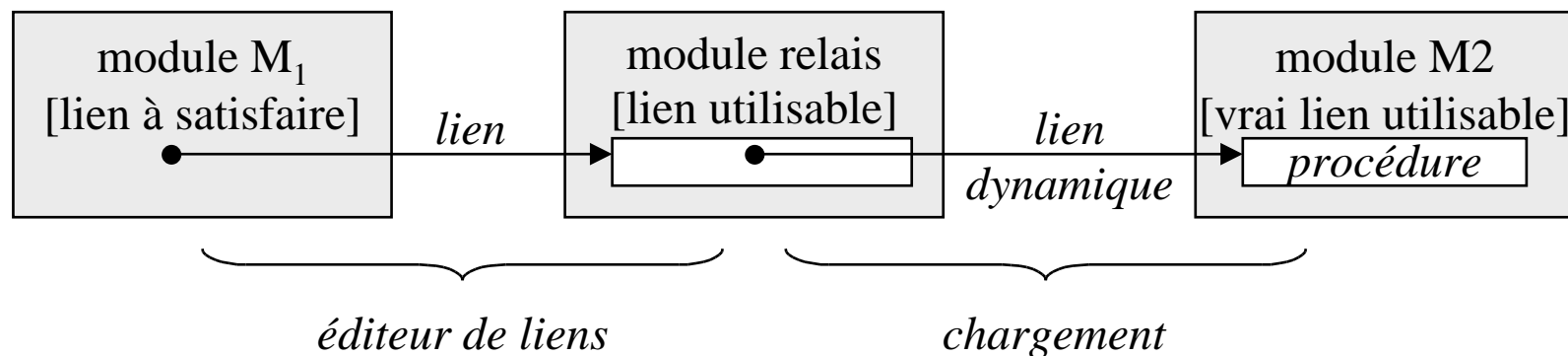
- Pour chaque lien à satisfaire de la table,
parcourir B_1, B_2, \dots, B_p et y rechercher le lien ←
 - si trouvé, ajouter le module à la liste, et traiter les liens
 - sinon, mettre son état à inexistant
- Nécessité de structure pour l'efficacité de la recherche
- À la fin, $LIN = LAS_L - \bigcup_{M \in L, B_i} LU_M$

Adjonction de modules (2)

- Parcourir les B_1, B_2, \dots, B_p
pour chacune, rechercher les liens à satisfaire
 - si trouvé, ajouter le module à la liste, et traiter les liens
 - sinon, mettre son état à inexistanten fin de bibliothèque, remettre inexistant \rightarrow à_satisfaire
- À la fin, il est possible que $LAS_L \cap \bigcup_M LU_M$
- Ordre à respecter
entre bibliothèques:
L à satisfaire dans M B_i , utilisable dans M' $B_j \Rightarrow i < j$
entre les modules d'une même bibliothèque si parcours séquentiel:
L à satisfaire dans M B , utilisable dans M' $B \Rightarrow M$ avant M'

Édition de liens dynamique

- Retarder l'édition de liens jusqu'au chargement ou au moment de l'utilisation
- Permet le partage des modules entre plusieurs programmes
- Soit au niveau module, soit au niveau bibliothèque



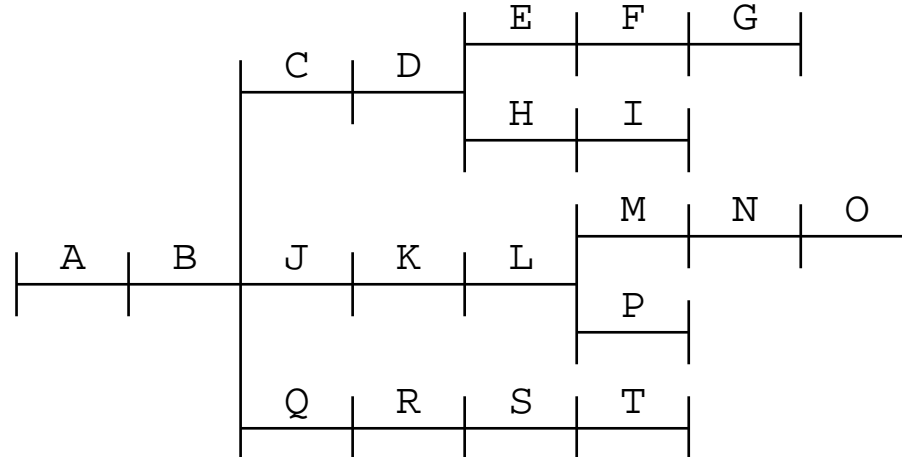
Notion de recouvrement

- Programmes de grande taille

Mémoire virtuelle de grande taille

=> le système met en mémoire réelle ce qui est nécessaire
recouvrement (*overlay*) géré par l'utilisateur

- regroupement des modules en parties pouvant se recouvrir
- édition de liens avec recouvrement
- instructions spécifiques de chargement de ces parties.



Références croisées

- But: savoir qui utilise quoi et où
 - mise au point: limiter la recherche des modules en cause
 - maintenance: quels modules sont concernés par un changement
 - amélioration du découpage en modules
 - organisation des bibliothèques

nom lien	module qui exporte	modules et emplacements qui importent
alpha	M1	M2(10, 50, 86), M3(26)
beta	M2	M1(104), M4(34)
gamma	M3	M4(246, 642)
delta	M4	M5(68, 94), M6(4), M7(456, 682, 624)

Chargement

- Édition de liens => programme exécutable
- Chargement = mise en mémoire et lancement
 - lecture en emplacement fixe => trop restrictive
 - avec translation => mise en mémoire n'importe où
 - informations de translation (traducteurs -> éditeur de liens)
 - adresse de lancement = adresse 1ère instruction à exécuter
 - traducteur reconnaît le programme principal et fournit l'adresse
 - éditeur de lien la transforme en adresse dans programme
 - environnement du programme exécutable => à construire
 - programme autonome = machine nue
 - programme sur machine abstraite = construire la machine
 - faire l'édition de liens dynamique, partage de sous programmes, ...

Conclusions

- translation => permettre l'exécution d'un module n'importe où
- lien => information permettant à un module d'accéder à un autre
- édition de liens => 2 phases
 - construction de la table des liens
 - remplacement des liens à satisfaire et construction du programme
- bibliothèque => moyen de compléter une liste de modules
 - sans structure -> ordre des modules est important
 - avec structure -> accès par lien utilisable
- édition de liens dynamique => partager les bibliothèques
- recouvrement => diminuer l'encombrement total du programme
- références croisées => savoir *qui utilise quoi*
- chargement => mise en mémoire, machine abstraite, lancement