Compilateur HEPIAL - 2020

Stephane Malandain – Michaël Minelli - TCP

Hes·SO GENÈVE
Haute Ecole Spécialisée
de Suisse occidentale

L'avenir est à créer

h e p i a
Haute école du paysage, d'ingénierie
et d'architecture de Genève

Déroulement

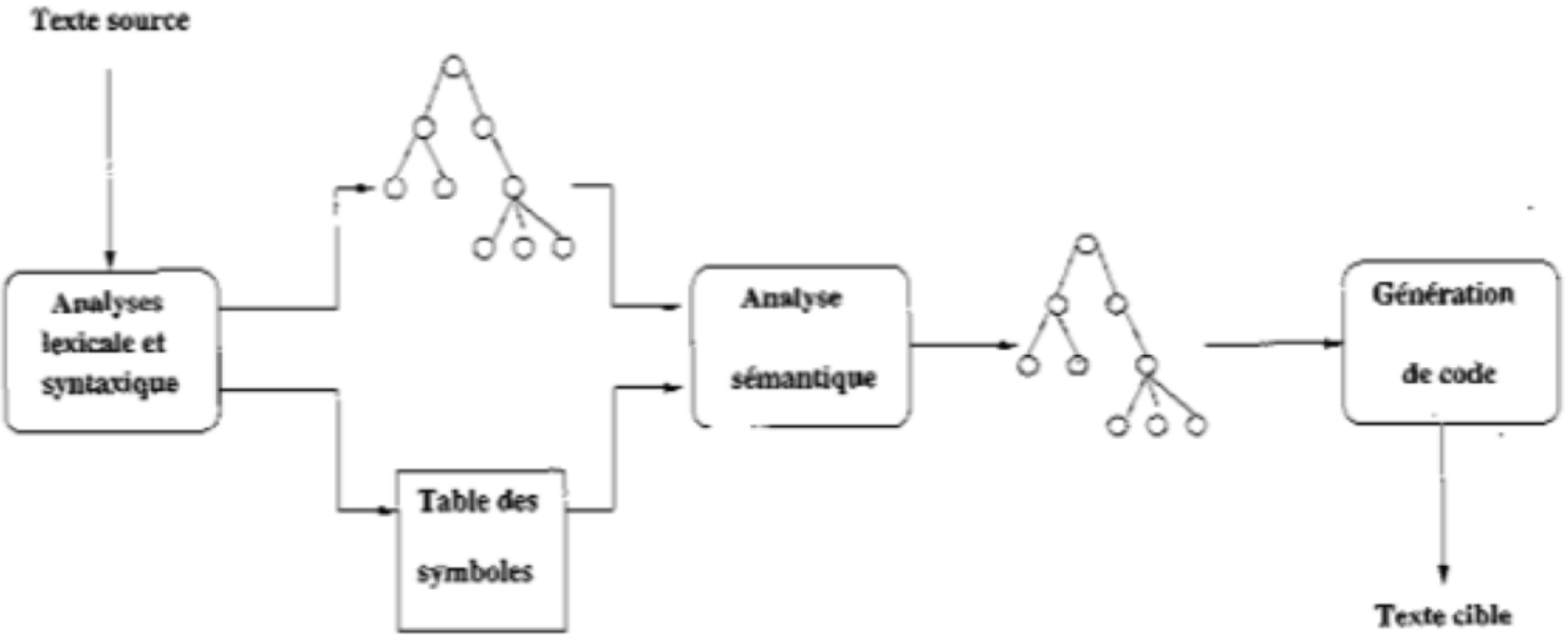
- Début du projet : 17 novembre 2020
- Fin du projet : 19 janvier 2021

- Groupes de 2 personnes

- Rendu : 22 janvier 2021
- Rapport + code

- Défense : 26 janvier 2021.
- Modalités : présentation + démo
- Lieu : HEPIA ou Teams (selon situation sanitaire)

HEPIAL 2020 : Composants



'Compilateur' jasmin

Fichier .class (java)

Grammaire

- Grammaire simplifiée
  - 2 types de données : entier, booleen
  - Pas de tableaux
  - Pas de procédures ni de fonctions
- Compilateur : HEPIAL -> JVM

Grammaire

- Grammaire

~~(1)~~	~~AXIOME~~	~~->~~	~~PROGRAMME~~
(2)	PROGRAMME	->	ENTETE DECLA* ***'debutprg'*** CORPS '***finprg'***
(3)	ENTETE	->	'***programme'*** ident
(4)	DECLA	->	DECLAVAR | DECLACONST
(5)	DECLAVAR	->	TYPE LIDENT **';'**
(6)	LIDENT	->	**ident** | LIDENT **',' ident**
(7)	DECLACONST	->	***'constante'*** TYPE **ident '='** EXPR *';'*
(8)	TYPE	->	***'entier'*** | ***'booleen'***
(9)	CORPS	->	INSTR *
(10)	INSTR	->	AFFECTATION | ECRIRE | LIRE | CONDITION | TANTQUE | POUR

Grammaire

- Grammaire (suite)

(11)	LIRE	->	'***lire'*** ident ' **;'**
(12)	ECRIRE	->	'***ecrire'*** EXPR ' **;'** | '***ecrire'*** **constanteChaine** ' **;'**
(13)	AFFECTATION	->	***ident*** '=' EXPR ';'
(15)	CONDITION	->	**'*si'*** EXPR **'*alors'*** CORPS **'*sinon'*** CORPS **'*finsi'***
(16)	TANTQUE	->	**'*tantque'*** EXPR **'*faire'*** CORPS **'*fintantque'***
(17)	POUR	->	**'*pour'*** ident **'*allantde'*** EXPR **'*a'*** EXPR **'*faire'*** CORPS **'*finpour'***
(18)	EXPR	->	EXPR OPEBIN EXPR | OPEUN EXPR | '(' EXPR ')' | OPERANDE
(19)	OPERANDE	->	**ident** | **constanteEnt** | '***vrai'*** | '***faux'***
(20)	OPEBIN	->	**'+'** | **'-'** | **'*'** | **'/'** | **'=='** | **'<>'** | **'<'** | **'>'** | **'<='** | **'>='** | **'*et'*** | **'*ou'***
(21)	OPEUN	->	**'~'** | **'*non'***

Grammaire

- Exemple de programme :

```
programme demo1
entier n;
entier result;
debutprg
      ecrire "Nombre a elever au carré: ";
   lire n;
   result = n*n;
      ecrire "Résultat:";
   ecrire result;
finprg
```

HEPIAL 2020

- Deux étapes principales à réaliser :

  1. Création de la table des symboles et de l'arbre syntaxique

  2. Analyse sémantique, production du code et optimisation

Table des symboles (TDS)

- Correspond au 'dictionnaire'
- Très simple car pas de fonction

- Type Hashmap
- Couple clé / Valeur

- ex :

```
var1, Entier
var2, Booleen
var3, Entier
```

Arbre Abstrait

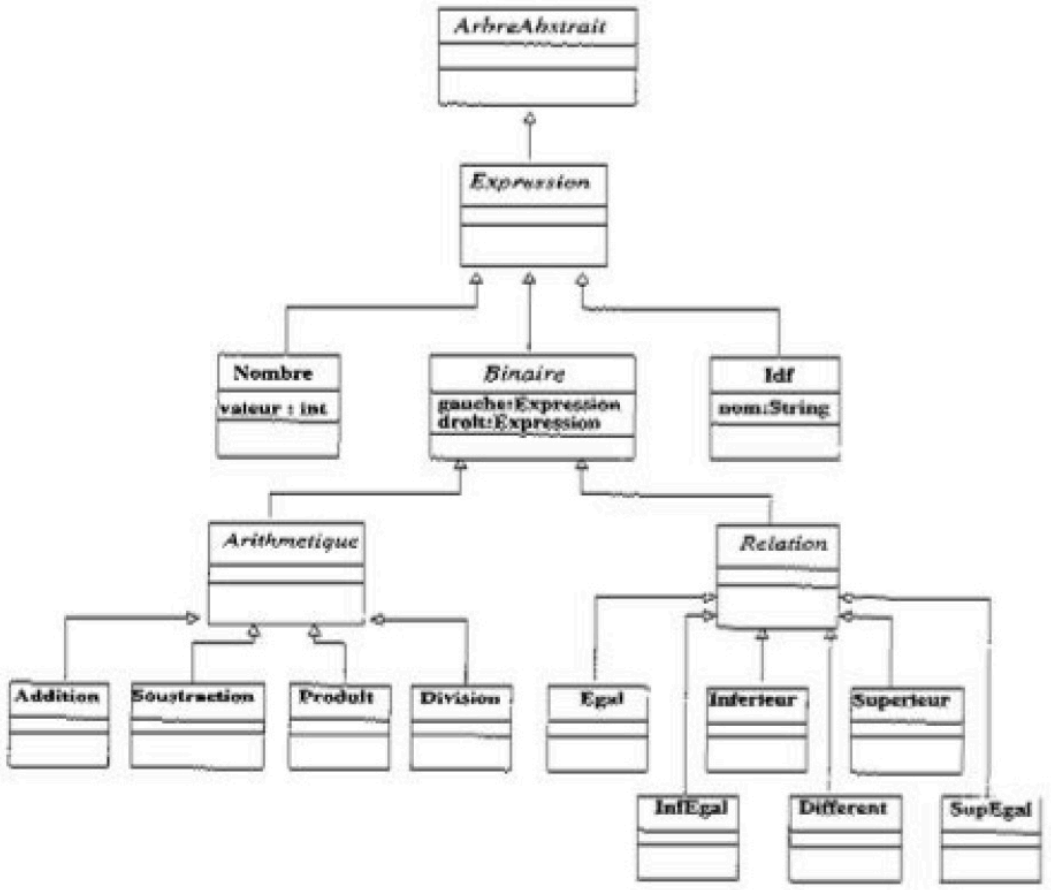


FIGURE 10 – graphe d'héritage complet des classes d'expressions

Arbre Abstrait

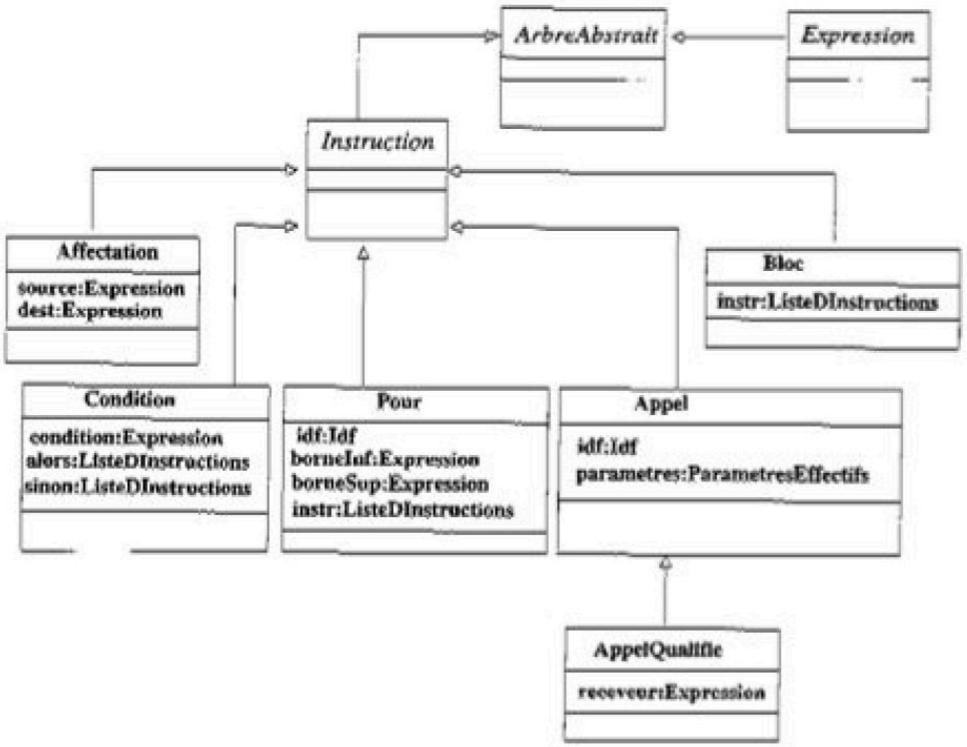


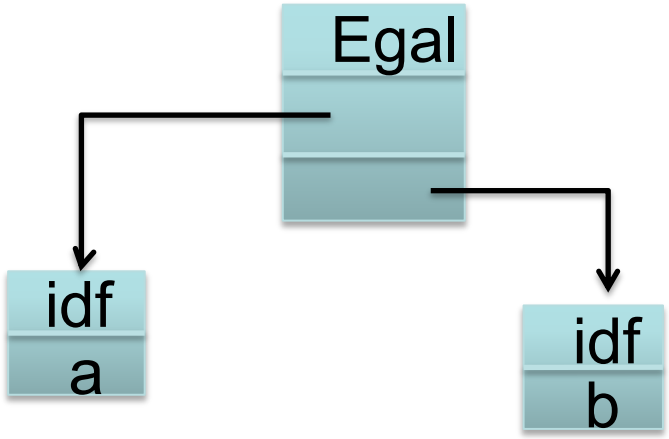
FIGURE 11 – Les classes Expression et Instruction sont des sous-classes de la classe abstraite ArbreAbstrait

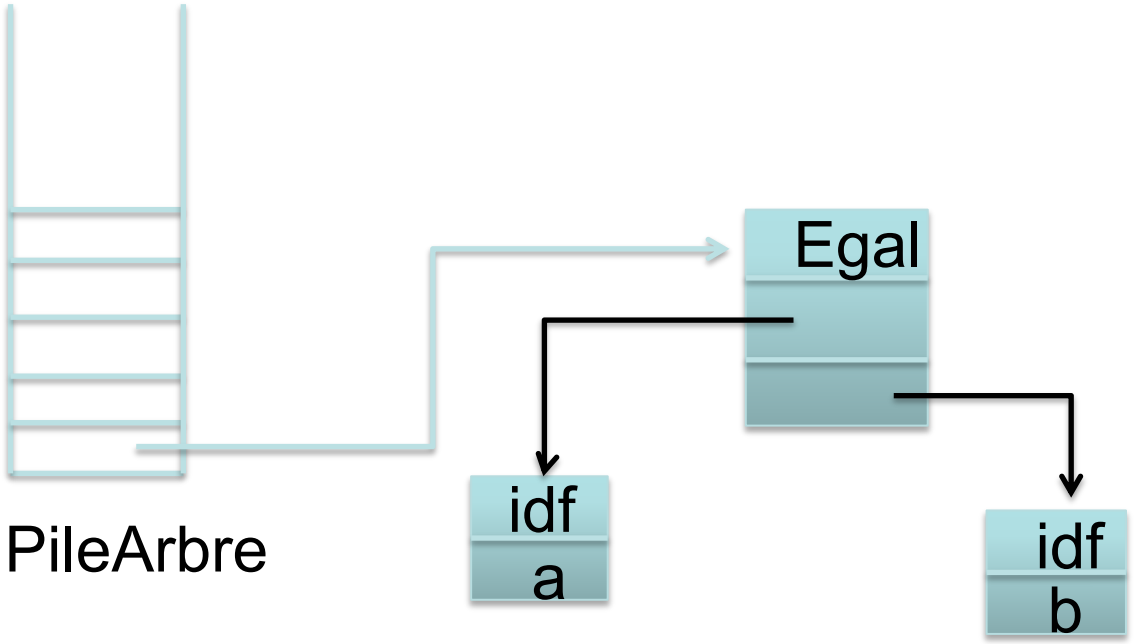
Exemple de code

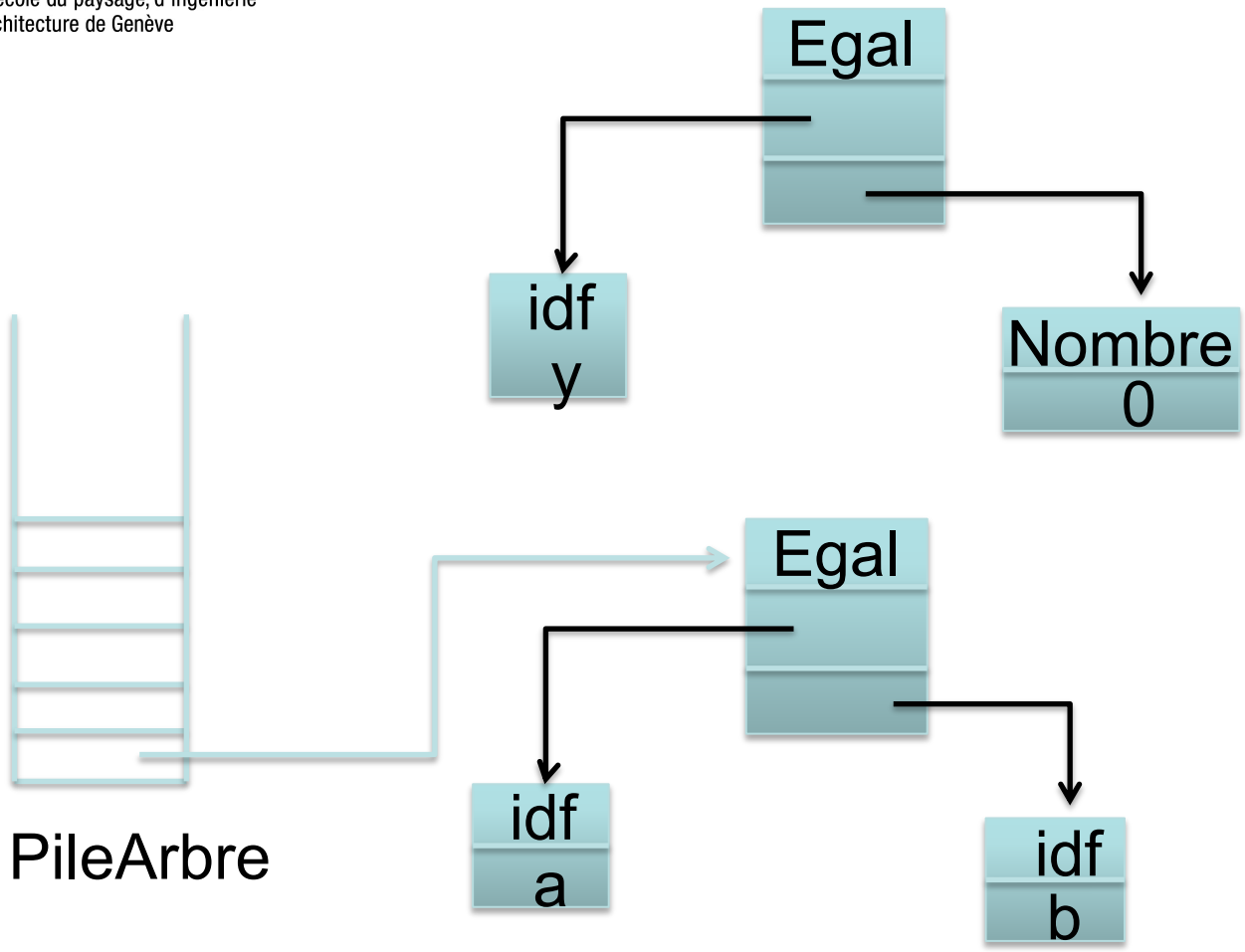
```
si (a == b) alors
      si (y == 0) alors x = 1 sinon x = 2
      finsi
sinon
      si a alors x = 3 * x ; y = 0
      finsi
finsi
```

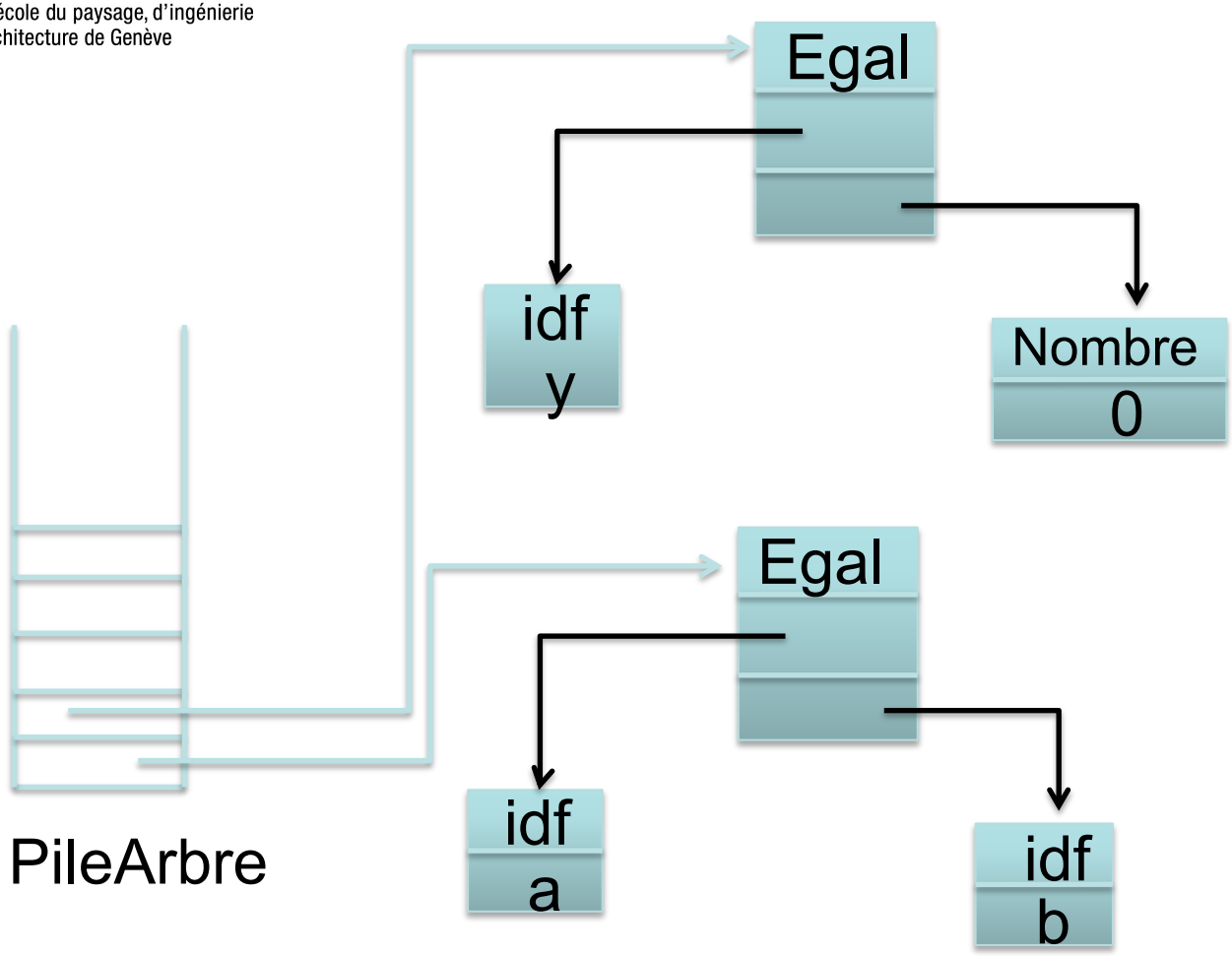
PileArbre

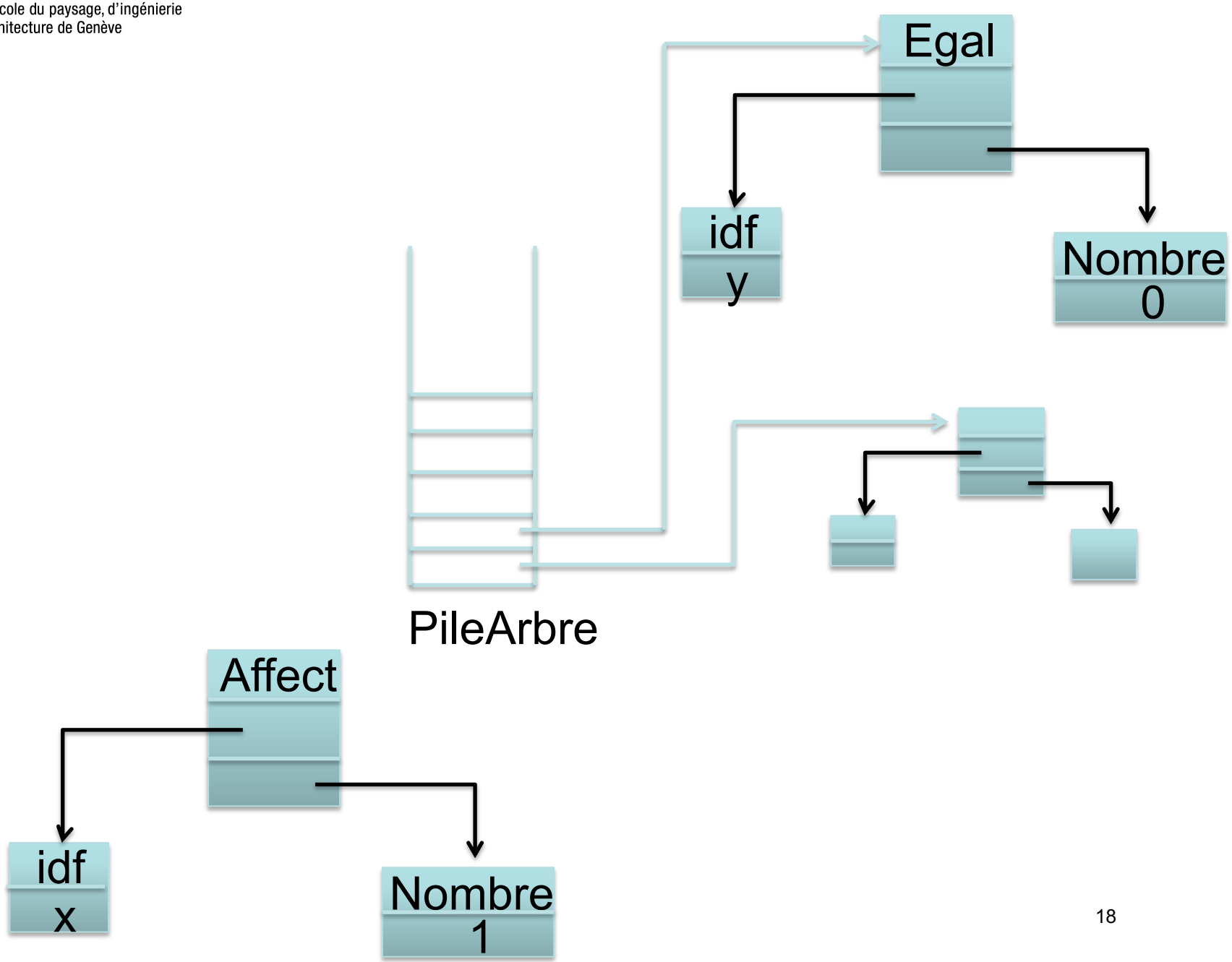
PileArbre

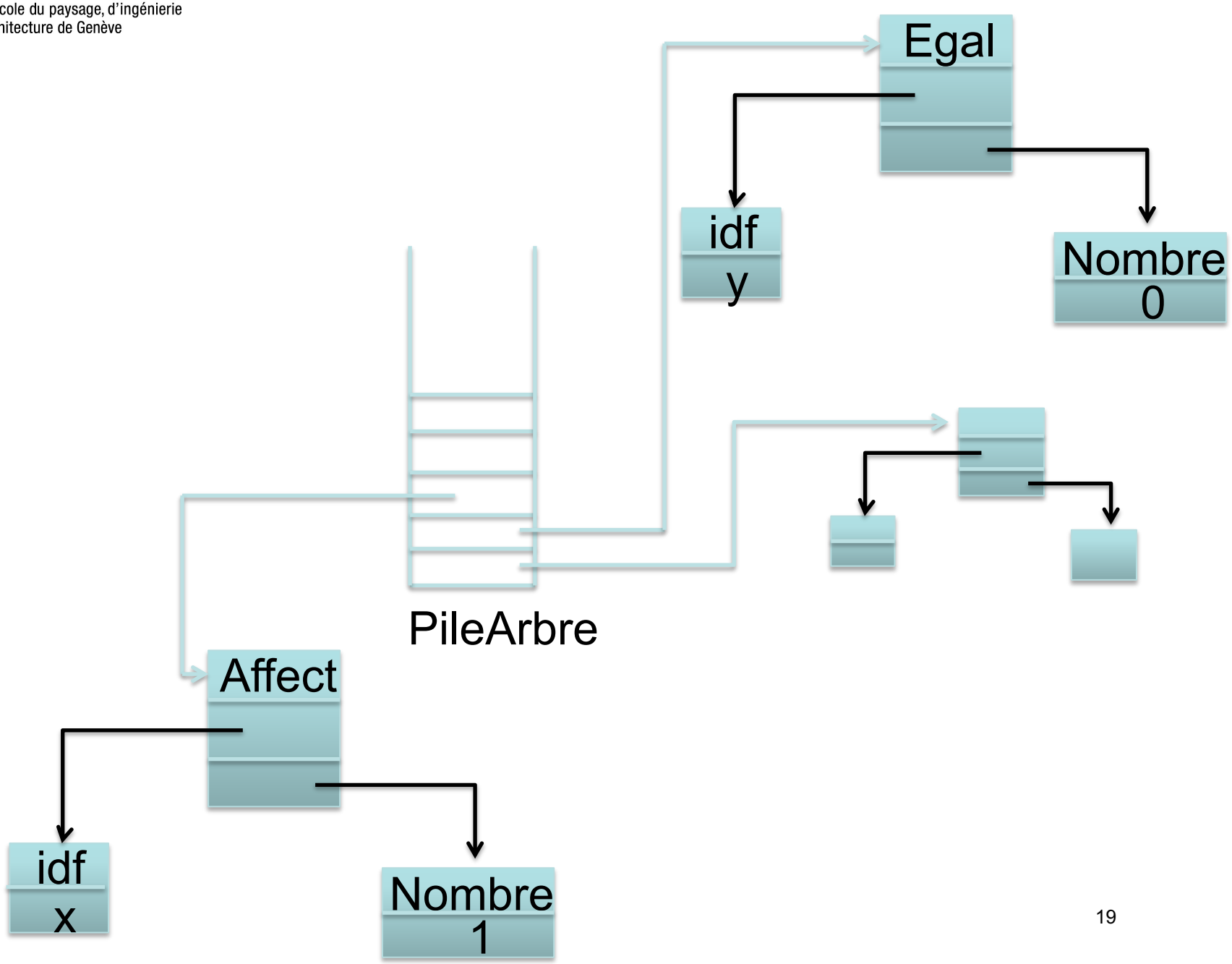


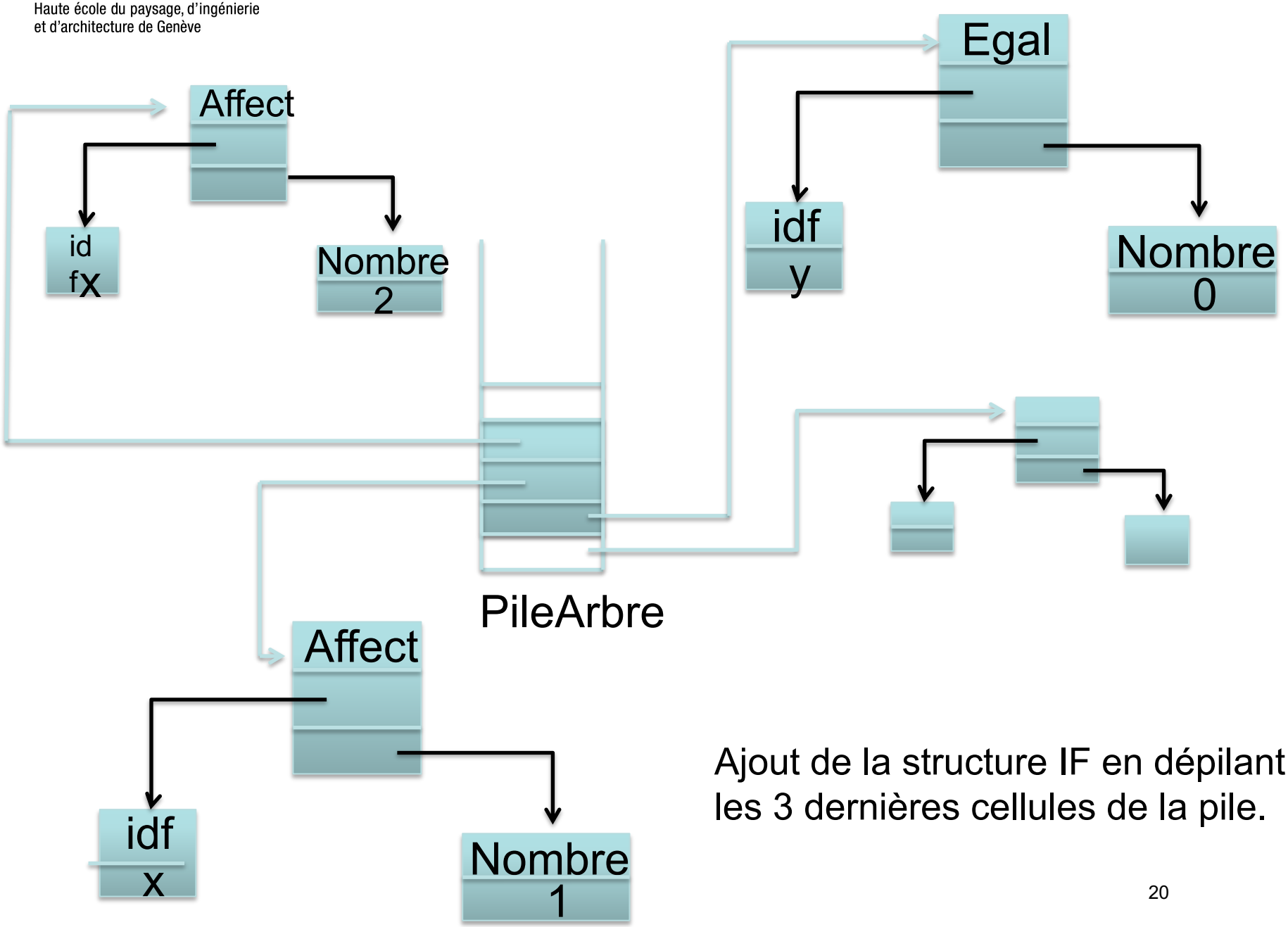

Egal
idf
a
idf
b
PileArbre


Egal
idf
y
Nombre
0
Egal
idf
a
idf
b
PileArbre

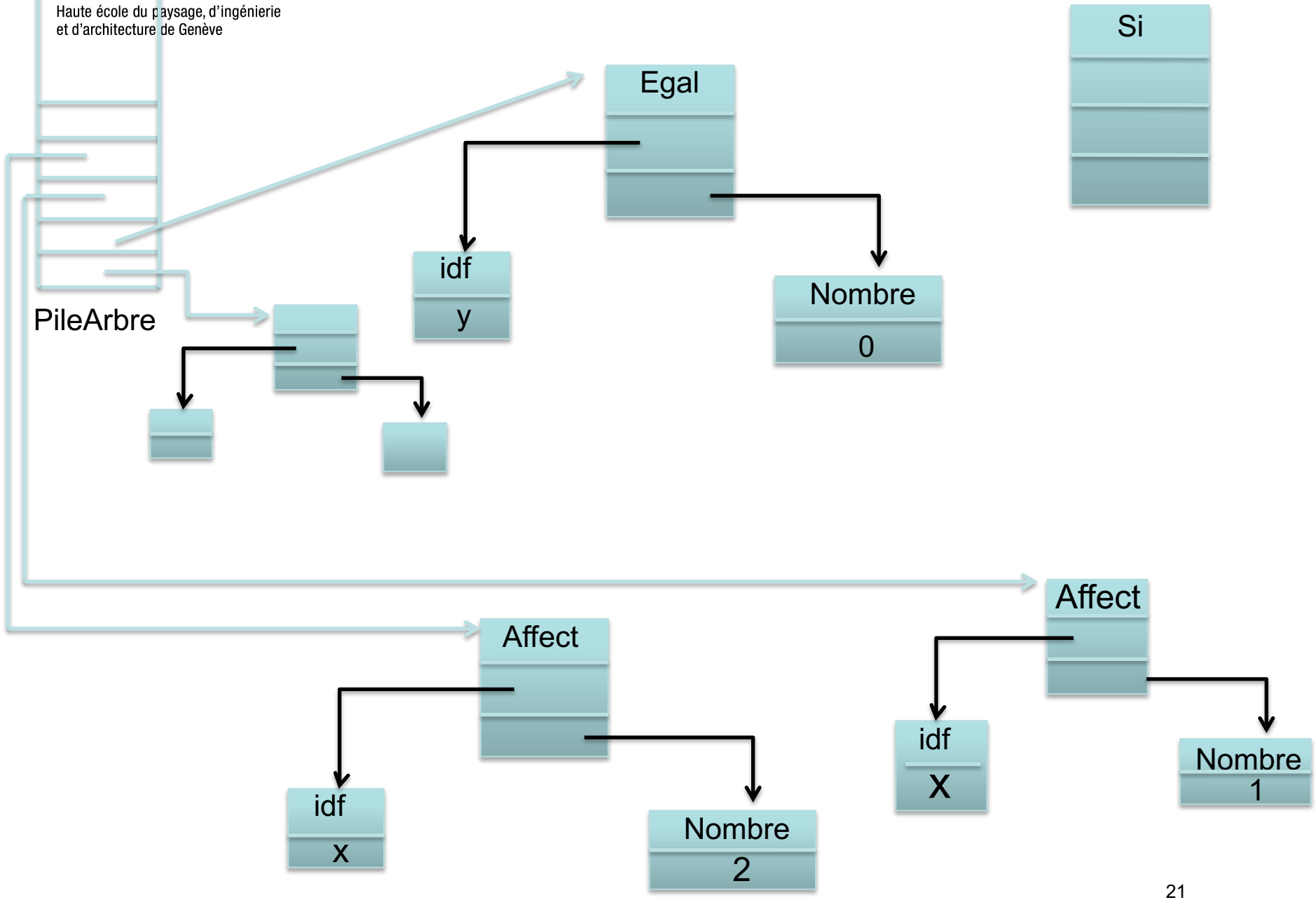

Egal
idf
y
Nombre
0
Egal
idf
a
idf
b
PileArbre

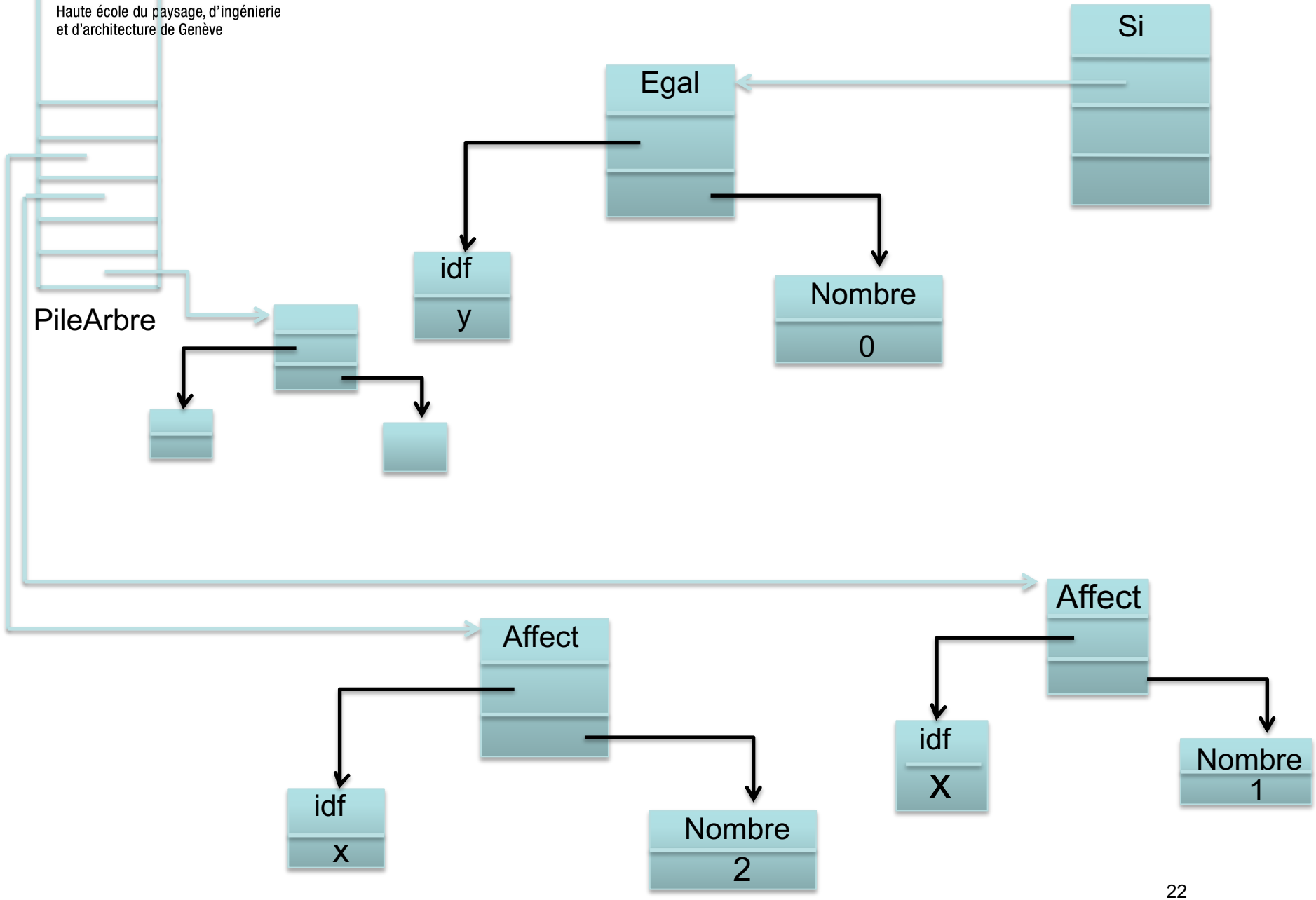

Egal
idf
y
Nombre
0
PileArbre
Affect
idf
x
Nombre
1

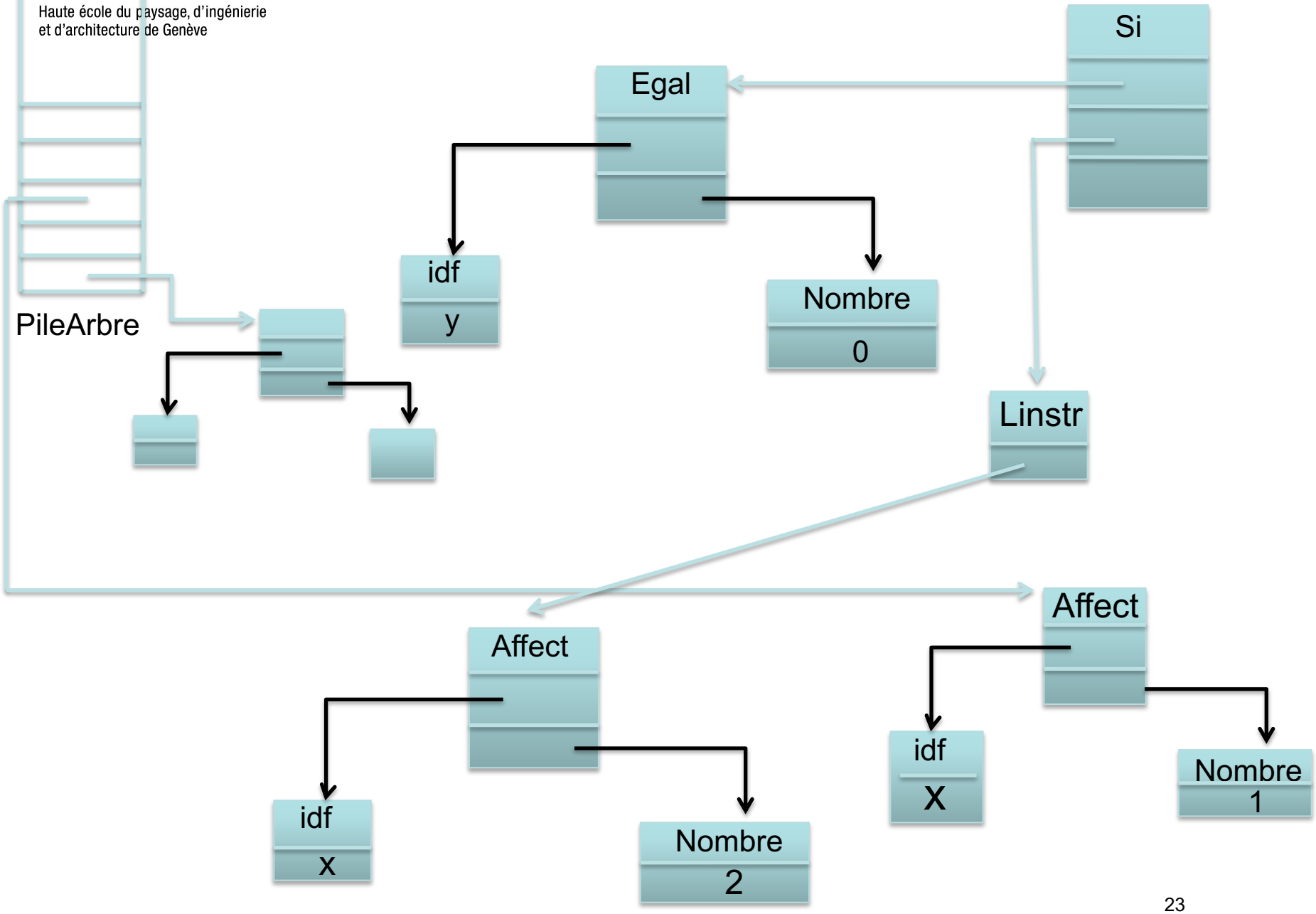

Egal
idf
y
Nombre
0
PileArbre
Affect
idf
x
Nombre
1

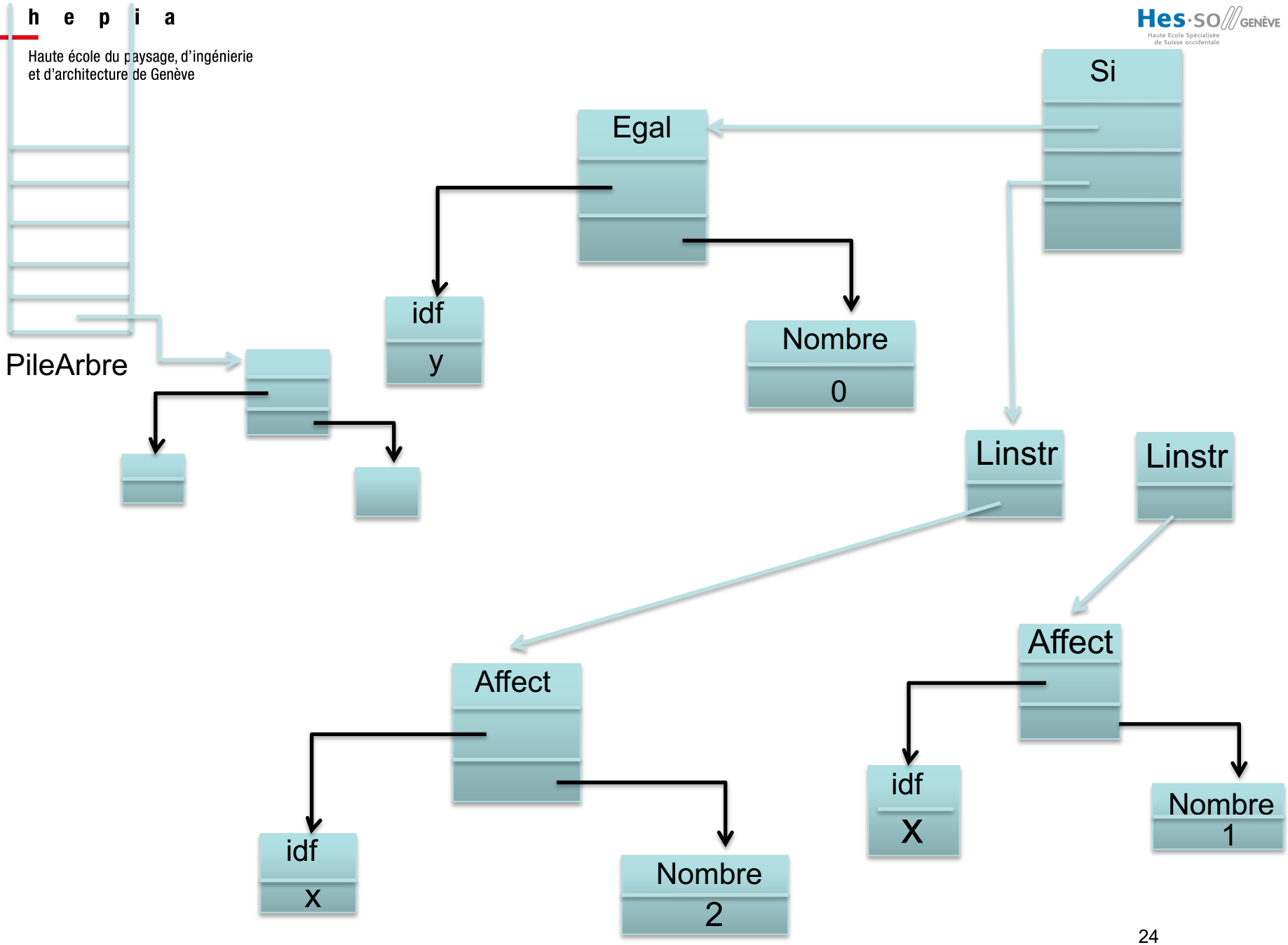


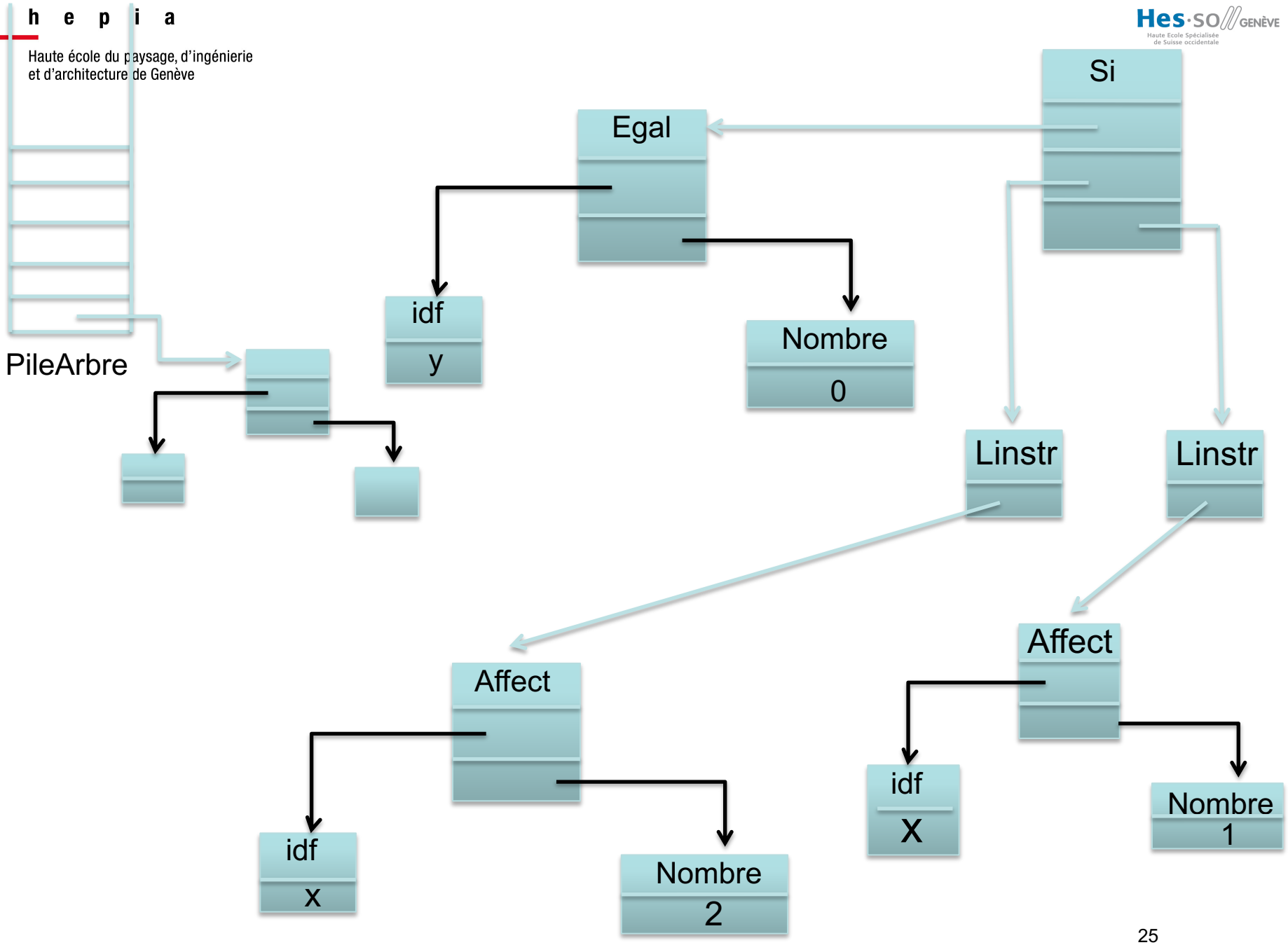
Ajout de la structure IF en dépilant les 3 dernières cellules de la pile.

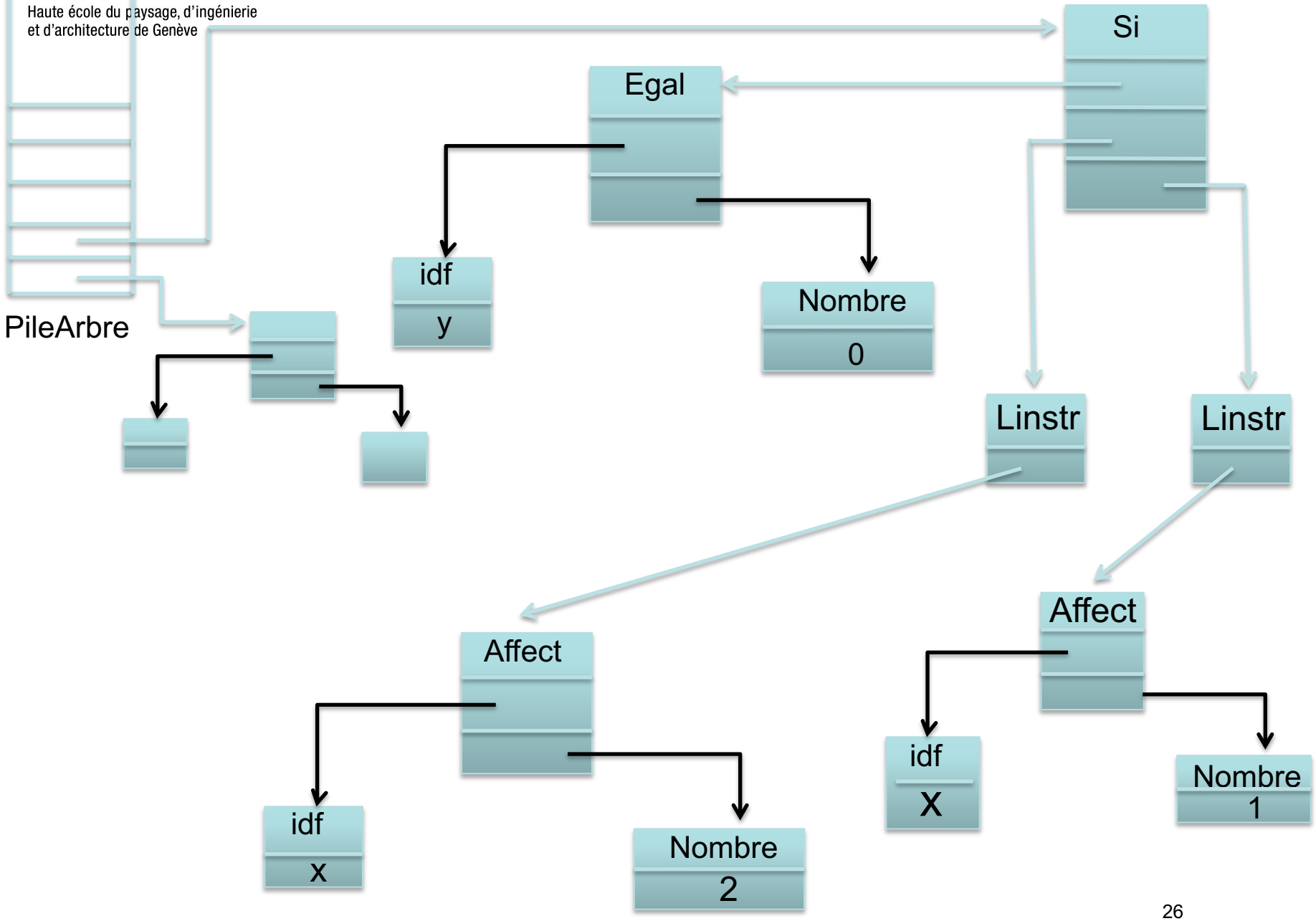

Si
Egal
idf
y
Nombre
0
PileArbre
Affect
idf
x
Nombre
2
Affect
idf
X
Nombre
1


Si
Egal
idf
y
Nombre
0
PileArbre
Affect
idf
x
Nombre
2
Affect
idf
X
Nombre
1

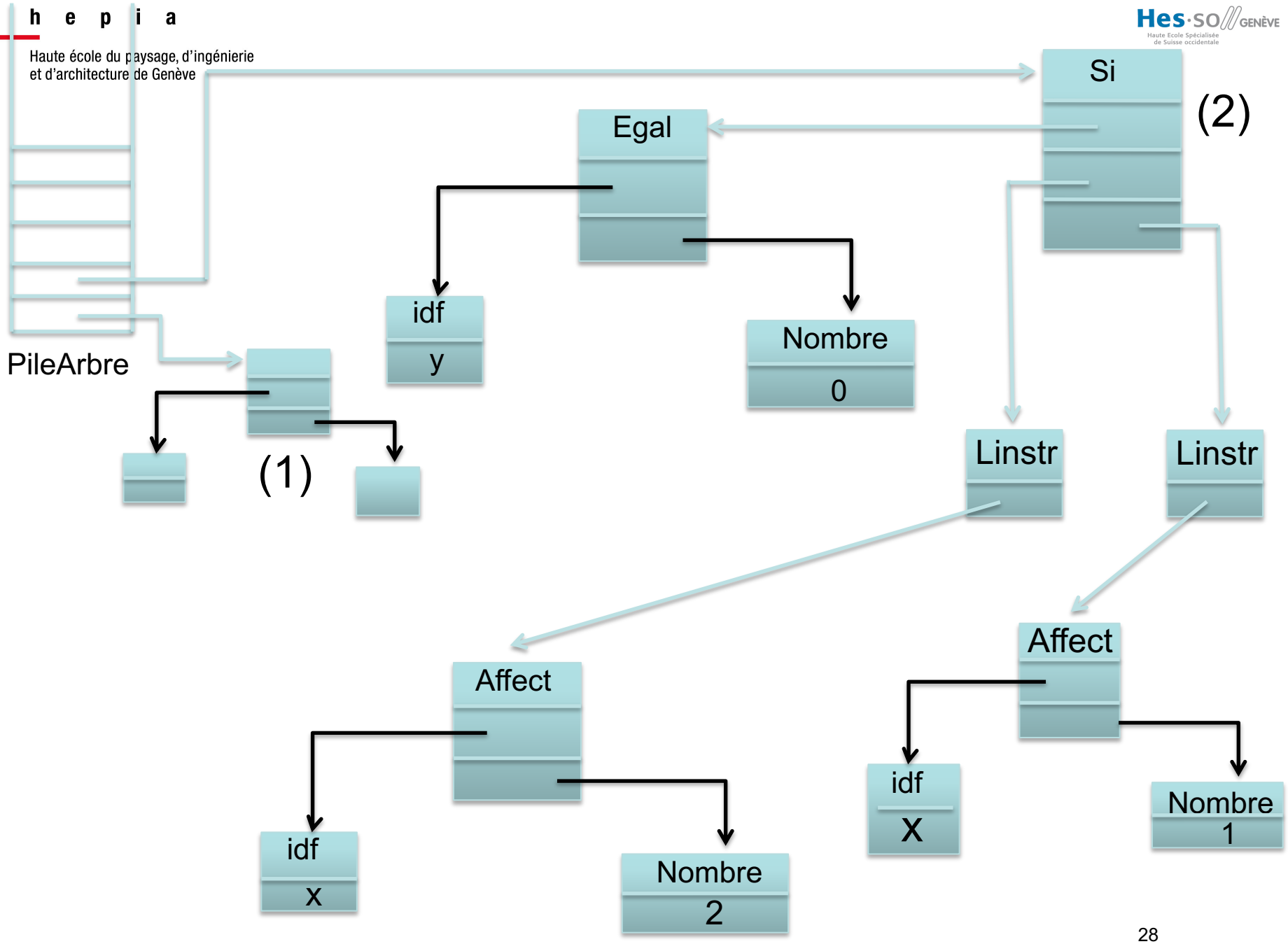

Si
Egal
idf
y
Nombre
0
PileArbre
Linstr
Affect
Affect
idf
x
Nombre
2
idf
X
Nombre
1


PileArbre
Si
Egal
idf
y
Nombre
0
Linstr
Linstr
Affect
idf
x
Nombre
2
Affect
idf
X
Nombre
1


PileArbre
Si
Egal
idf
y
Nombre
0
Linstr
Linstr
Affect
idf
x
Nombre
2
Affect
idf
X
Nombre
1


Si
Egal
idf
y
Nombre
0
PileArbre
Linstr
Linstr
Affect
Affect
idf
x
Nombre
2
idf
X
Nombre
1

```
si (a == b) alors                                          (1)
      si (y == 0) alors x = 1 sinon x = 2 finsi            (2)
sinon
      si a alors x = 3 * x ; y = 0 finsi                   (3)
finsi
```


Si
(2)
Egal
idf
y
Nombre
0
PileArbre
(1)
Linstr
Linstr
Affect
Affect
idf
x
Nombre
2
idf
X
Nombre
1

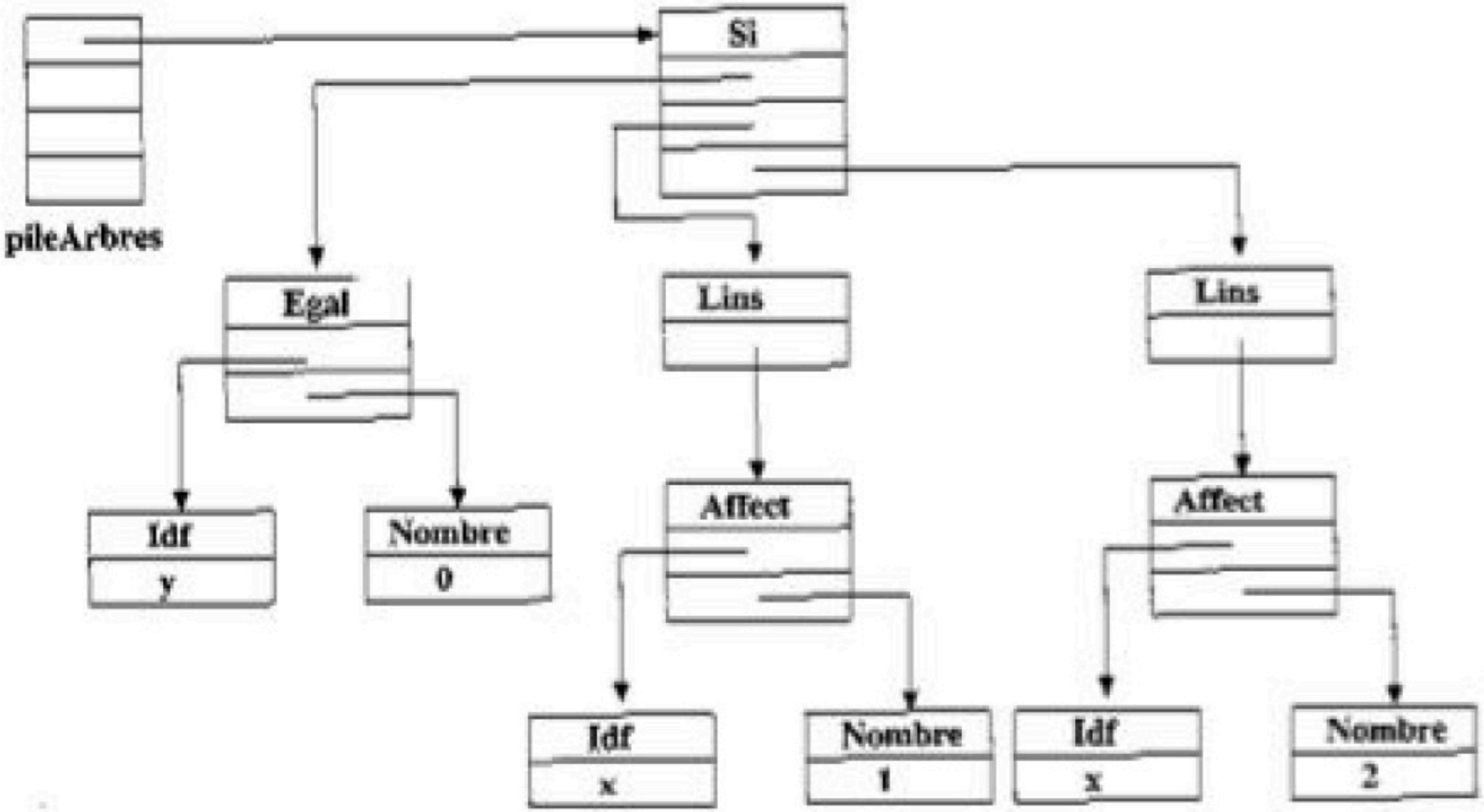


FIGURE 14 – L'arbre de la partie 'Alors' est au sommet de la pile

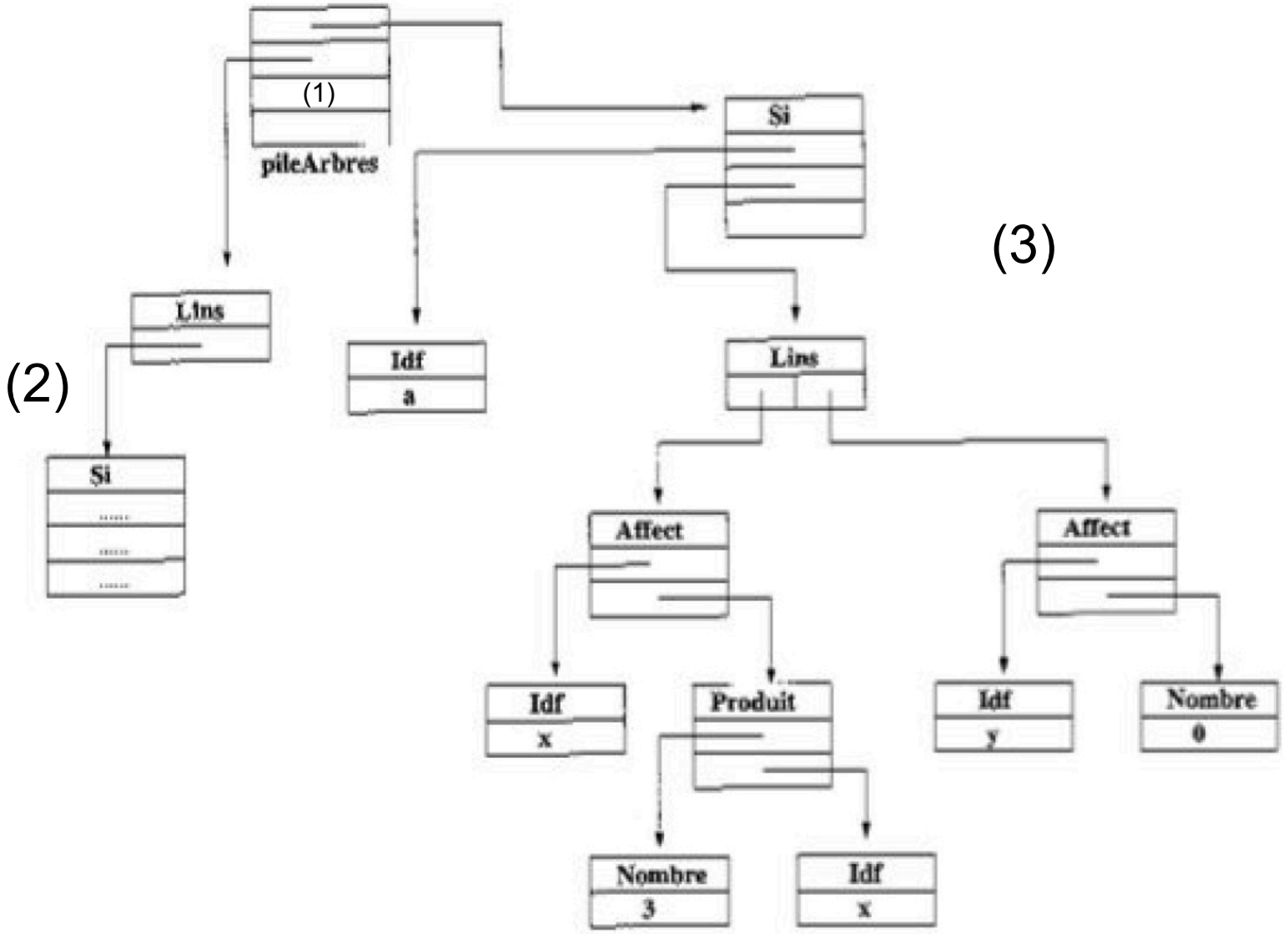


FIGURE 15 – L'arbre de la partie 'Sinon' est au sommet de la pile