

## MIF14 - [STRATIFICATOR](#)

### 1. Exécuter le programme

Nous avons décidé de développer notre outil de stratification en Java. Pour lancer le .jar fourni, il vous faudra utiliser une version de java supérieure ou égale à 16.

Voici comment exécuter le .jar :

```
java -jar stratificator.jar program1.dl
```

Les paramètres sont :

- h pour afficher l'aide.
- v pour afficher le programme parsé dans la console.
- o [fichier.txt] pour spécifier le fichier de sortie.

Exemple :

```
java -jar stratificator.jar -o result.txt -v program1.dl
```

Attention : le fichier d'entrée doit toujours être passé comme dernier paramètre !

### 2. Parser

Notre parser fonctionne essentiellement avec des regex. En effet, une fois le fichier à traiter ouvert par un BufferedReader, nous le traitons ligne par ligne. Afin de pouvoir parser le fichier dans les meilleures conditions, nous nous sommes donnés quelques sécurités. Dans un premier temps, nous remplaçons tous les “,” en “ ‘ ”, nous supprimons tous les espaces qui ne sont pas nécessaires et nous ignorons les commentaires. S'il reste une erreur dans la ligne, un message d'erreur est affiché dans la console avec le numéro de ligne concernée.

Une fois cette transformation faite, nous comparons la ligne avec deux regex différents qui vont nous permettre de savoir s'il s'agit d'un EDB ou d'un IDB. Nous avons remarqué que les EDB et les IDB sont des expressions qui partagent un pattern commun :

```
\s*\w+\s*(\s*'?\w+'?\s*(,\s*'?\w+'?\s*)*\s*)\s*
```

Ceci revient à reconnaître qu'une ligne est égale à quelque chose de cette forme : Expression(param1, paramX, ...). Les IDBs partagent le même pattern, mais possèdent en plus le signe qui définit les règles “:-” et parfois le mot “not” suivi d'une ou plusieurs expressions.

### 3. Modèles

Lorsque le parser a défini si la ligne en cours de traitement était un EDB ou un IDB, on construit un objet avec le modèle associé. Nous avons le modèle Expression, dans lequel nous allons pouvoir placer nos EDB, et le modèle Rule, dans lequel nous allons placer nos IDB. Il peut être intéressant de noter que nous découpons nos Expressions en un prédicat suivi d'une liste de paramètres et que les Rules possèdent pour tête une Expression et pour corps une liste d'autres Expressions. Une expression est négative si elle commence par un "not". Finalement, toutes les lignes parsées sont soit ignorées, soit transformées en IDB ou EDB et sont stockées dans un dernier modèle que l'on appelle Program.

### 4. Algorithme et output

Une fois le programme parsé et les modèles construits, nous récupérons les EDB et les différentes Rules et les utilisons dans notre implémentation Java de l'algorithme fourni. Celui-ci nous permet de récupérer un objet Stratification contenant les différentes strates.

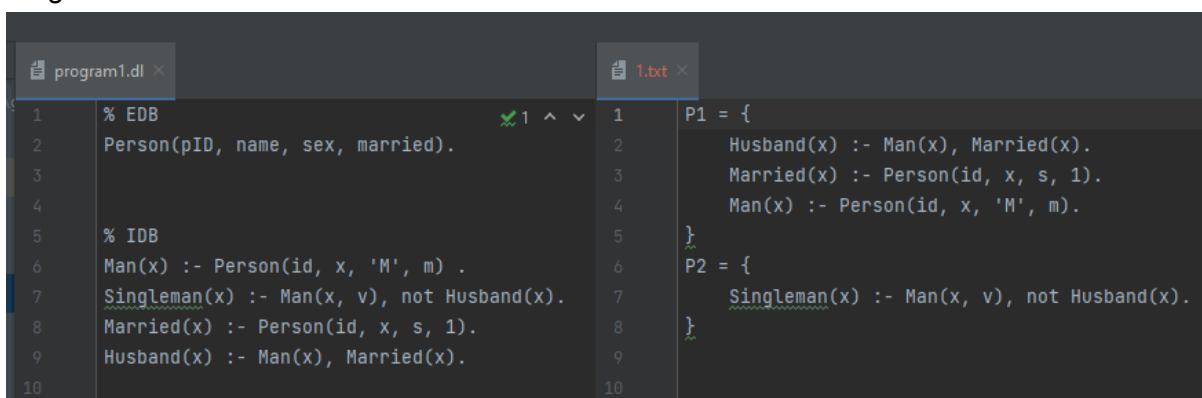
Pour afficher ces strates, nous avons deux options. On peut afficher le résultat dans la console ou dans un fichier de sortie, à l'aide de l'option -o [fichier]. Le paramètre -v nous permet aussi d'afficher le programme parsé pour s'assurer qu'aucune erreur ne s'est glissée dans le processus.

### 5. Tests

Comme préconisé, nous avons ajouté une dizaine de programmes pour tester notre algorithme de stratification. Il n'y a pas de stratégie particulière avec ces tests. Nous avons juste cherché à émettre suffisamment de règles pour pouvoir les lier et ainsi avoir plusieurs strates. Nous avons aussi mis dans ces programmes celui qui était donné dans le sujet à titre de comparaison.

### 6. Annexes

Programme 1 :



```
program1.dl
1 % EDB
2 Person(pID, name, sex, married).
3
4
5 % IDB
6 Man(x) :- Person(id, x, 'M', m) .
7 Singleman(x) :- Man(x, v), not Husband(x).
8 Married(x) :- Person(id, x, s, 1).
9 Husband(x) :- Man(x), Married(x).
10

1.txt
1 P1 = {
2   Husband(x) :- Man(x), Married(x).
3   Married(x) :- Person(id, x, s, 1).
4   Man(x) :- Person(id, x, 'M', m).
5 }
6 P2 = {
7   Singleman(x) :- Man(x, v), not Husband(x).
8 }
9
10
```

## Programme 2 :

program2.dl ×	2.txt ×
1 % EDB	1 P1 = {
2 Person(pID, name, sex, married).	2 Husband(x) :- Man(x), Married(x).
3	3 Married(x) :- Person(id, x, s, 1).
4	4 Man(x) :- Person(id, x, 'M', m).
5 % IDB	5 Woman(x) :- Person(id, x, 'W', m).
6 Man(x) :- Person(id, x, 'M', m).	6 }
7 Woman(x) :- Person(id, x, 'W', m).	7 P2 = {
8 Single(x) :- Man(x, v), not Husband(x).	8 Single(x) :- Man(x, v), not Husband(x).
9 Single(x) :- Woman(x, v), not Husband(x).	9 Single(x) :- Woman(x, v), not Husband(x).
10 Married(x) :- Person(id, x, s, 1).	10 Test(x) :- Single(x).
11 Husband(x) :- Man(x), Married(x).	11 }
12 Test(x) :- Single(x).	12 }

## Programme 3 :

program3.dl ×	3.txt ×
1 % EDB	1 P1 = {
2 F(x,y).	2 P(x, y) :- F(x, y).
3 C(x,y).	3 P(x, z) :- P(x, y), F(y, z).
4	4 R(x, y) :- P(x, y), C(x, 'R').
5	5 R(x, y) :- P(x, y), C(y, 'R').
6 % IDB	6 }
7 S(x) :- P(x,x), not R(x,x).	7 P2 = {
8 R(x,y) :- P(x,y), C(x,'R').	8 S(x) :- P(x, x), not R(x, x).
9 R(x,y) :- P(x,y), C(y,'R').	9 }
10 P(x,y) :- F(x,y).	10 P3 = {
11 P(x,z) :- P(x,y), F(y,z).	11 A(X) :- P(x, x), not S(x).
12 A(X) :- P(x,x), not S(x).	12 }

## Programme 4 :

program4.dl ×	4.txt ×
1 % EDB	1 P1 = {
2 E(x).	2 S(x) :- F(x).
3 T(x).	3 V(x) :- T(x).
4 F(x).	4 }
5	5 P2 = {
6 % IDB	6 P(x) :- not S(x), E(x).
7 R(x) :- P(x), not Q(x).	7 Q(x) :- not V(x), E(x).
8 Q(x) :- not V(x), E(x).	8 }
9 P(x) :- not S(x), E(x).	9 P3 = {
10 V(x) :- T(x).	10 R(x) :- P(x), not Q(x).
11 S(x) :- F(x).	11 }

## Programme 5 :

program5.dl	5.txt
1 % EDB	1 P1 = {
2 Feather(x).	2 Human(x) :- Hair(x), not Feather(x).
3 Hair(x).	3 Speak(x) :- Human(x).
4 Fur(x).	4 Bird(x) :- Feather(x), not Hair(x).
5	5 Cat(x) :- Fur(x), not Hair(x), not Feather(x).
6 % IDB	6 Growl(x) :- Dog(x).
7 Dog(x) :- Hair(x), Fur(x), not Feather(x).	7 Dog(x) :- Hair(x), Fur(x), not Feather(x).
8 Bird(x) :- Feather(x), not Hair(x).	8 }
9 Cat(x) :- Fur(x), not Hair(x), not Feather(x).	9
10 Human(x) :- Hair(x), not Feather(x).	10
11 Growl(x) :- Dog(x).	
12 Speak(x) :- Human(x).	

## Programme 6 :

program6.dl	6.txt
1 % EDB	1 P1 = {
2 Person(pID, name, sex, married, parent).	2 Parent(x) :- Person(id, x, s, m, 'Y').
3	3 Single(x) :- Person(id, x, s, 'Y', p).
4 % IDB	4 Man(x) :- Person(id, x, 'M', m, p).
5 Man(x) :- Person(id, x, 'M', m, p).	5 Woman(x) :- Person(id, x, 'W', m, p).
6 Woman(x) :- Person(id, x, 'W', m, p).	6 }
7 Parent(x) :- Person(id, x, s, m, 'Y').	7 P2 = {
8 Single(x) :- Person(id, x, s, 'Y', p).	8 Mother(x) :- Woman(x), Parent(x), not Single(x).
9 Father(x) :- Man(x), Parent(x), not Single(x).	9 Parents(x, y) :- Father(x), Mother(y).
10 Mother(x) :- Woman(x), Parent(x), not Single(x).	10 Father(x) :- Man(x), Parent(x), not Single(x).
11 Parents(x,y) :- Father(x), Mother(y).	11 }

## Programme 7 :

program7.dl	7.txt
1 % EDB	1 P1 = {
2 Vehicle(vID, tires, wings, propellers).	2 Motorcycle(x) :- RoadCompatible(x).
3	3 Plane(x) :- AirCompatible(x).
4 % IDB	4 WaterCompatible(x) :- Vehicle(x, 0, 0, 3).
5 RoadCompatible(x) :- Vehicle(x, 4, 0, 0).	5 RoadCompatible(x) :- Vehicle(x, 4, 0, 0).
6 RoadCompatible(x) :- Vehicle(x, 2, 0, 0).	6 RoadCompatible(x) :- Vehicle(x, 2, 0, 0).
7 AirCompatible(x) :- Vehicle(x, 0, 2, 0).	7 Bicycle(x) :- RoadCompatible(x).
8 AirCompatible(x) :- Vehicle(x, 0, 0, 8).	8 AirCompatible(x) :- Vehicle(x, 0, 2, 0).
9 WaterCompatible(x) :- Vehicle(x, 0, 0, 3).	9 AirCompatible(x) :- Vehicle(x, 0, 0, 8).
10 Car(x) :- RoadCompatible(x).	10 Car(x) :- RoadCompatible(x).
11 Motorcycle(x) :- RoadCompatible(x).	11 Boat(x) :- WaterCompatible(x).
12 Bicycle(x) :- RoadCompatible(x).	12 }
13 Plane(x) :- AirCompatible(x).	13 P2 = {
14 Boat(x) :- WaterCompatible(x).	14 FastRoadTravel(x) :- RoadCompatible(x), not Bicycle(x).
15 Helicopter(x) :- AirCompatible(x), not WaterCompatible(x).	15 Helicopter(x) :- AirCompatible(x), not WaterCompatible(x).
16 SlowAirTravel(x) :- AirCompatible(x), not Plane(x).	16 SlowAirTravel(x) :- AirCompatible(x), not Plane(x).
17 FastRoadTravel(x) :- RoadCompatible(x), not Bicycle(x).	17 Travel(x) :- SlowAirTravel(x).
18 Travel(x) :- SlowAirTravel(x).	18 Travel(x) :- FastRoadTravel(x).
19 Travel(x) :- FastRoadTravel(x).	19 }
20 FastTravel(x) :- Travel(x), not SlowAirTravel(x).	20 P3 = {
21 SlowTravel(x) :- Travel(x), not FastRoadTravel(x).	21 SlowTravel(x) :- Travel(x), not FastRoadTravel(x).
22	22 FastTravel(x) :- Travel(x), not SlowAirTravel(x).
	23 }

## Programme 8 :

program8.dl	8.txt
1 % EDB	1 P1 = {
2 Substance(sID, allowed, dangerous, liquid).	2 Liquid(x) :- Substance(x, allowed, dangerous, 'Y').
3	3 Alcohol(x) :- Allowed(x), Dangerous(x), Liquid(x).
4 % IDB	4 Allowed(x) :- Substance(x, 'Y', dangerous, liquid).
5 Allowed(x) :- Substance(x, 'Y', dangerous, liquid).	5 Dangerous(x) :- Substance(x, allowed, 'Y', liquid).
6 Dangerous(x) :- Substance(x, allowed, 'Y', liquid).	6 }
7 Liquid(x) :- Substance(x, allowed, dangerous, 'Y').	7 P2 = {
8 Tobacco(x) :- Allowed(x), Dangerous(x), not Liquid(x).	8 Tobacco(x) :- Allowed(x), Dangerous(x), not Liquid(x).
9 Cannabis(x) :- Dangerous(x), not Allowed(x), not Liquid(x).	9 Cannabis(x) :- Dangerous(x), not Allowed(x), not Liquid(x).
10 Alcohol(x) :- Allowed(x), Dangerous(x), Liquid(x).	10 SeflmadeAlcohol(x) :- Dangerous(x), not Allowed(x), Liquid(x).
11 SeflmadeAlcohol(x) :- Allowed(x), not Allowed(x), Liquid(x).	11 }
12 CanSmoke(x) :- Tobacco(x), not Cannabis(x).	12 P3 = {
13 CanDrink(x) :- Alcohol(x), not SeflmadeAlcohol(x).	13 CanSmoke(x) :- Tobacco(x), not Cannabis(x).
	14 CanDrink(x) :- Alcohol(x), not SeflmadeAlcohol(x).
	15 }

## Programme 9 :

program9.dl	9.txt
1 % EDB	1 P1 = {
2 Food(fID, color, fruit, vegetable, junk).	2 Red(x) :- Food(x, 'red', fruit, vegetable, junk).
3	3 Fruit(x) :- Food(x, color, 'Y', vegetable, junk).
4 % IDB	4 Yellow(x) :- Food(x, 'yellow', fruit, vegetable, junk).
5 Fruit(x) :- Food(x, color, 'Y', vegetable, junk).	5 Corn(x) :- Vegetable(x), Yellow(x).
6 Vegetable(x) :- Food(x, color, fruit, 'Y', junk).	6 Junk(x) :- Food(x, color, fruit, vegetable, 'Y').
7 Junk(x) :- Food(x, color, fruit, vegetable, 'Y').	7 Tomato(x) :- Vegetable(x), Fruit(x), Red(x).
8 Yellow(x) :- Food(x, 'yellow', fruit, vegetable, junk).	8 Vegetable(x) :- Food(x, color, fruit, 'Y', junk).
9 Red(x) :- Food(x, 'red', fruit, vegetable, junk).	9 Banana(x) :- Fruit(x), Yellow(x).
10 Banana(x) :- Fruit(x), Yellow(x).	10 }
11 Corn(x) :- Vegetable(x), Yellow(x).	11 P2 = {
12 Strawberry(x) :- Fruit(x), not Vegetable(x), Red(x).	12 Strawberry(x) :- Fruit(x), not Vegetable(x), Red(x).
13 Beet(x) :- Vegetable(x), not Fruit(x), Red(x).	13 FruitSalad(x, y) :- Fruit(x), Red(x), not Tomato(x), Banana(y).
14 Tomato(x) :- Vegetable(x), Fruit(x), Red(x).	14 Beet(x) :- Vegetable(x), not Fruit(x), Red(x).
15 FruitSalad(x, y) :- Fruit(x), Red(x), not Tomato(x), Banana(y).	15 }

## Programme 10 :

program10.dl	10.txt
1 % EDB	1 P1 = {
2 Sport(sID, run, jump, swim, throw, sync).	2 500MHurdle(x) :- Run(x), Jump(x).
3	3 SynchronizedSwimming(x) :- Swim(x), Jump(x), Sync(x).
4 % IDB	4 Throw(x) :- Sport(x, run, jump, swim, 'Y', sync).
5 Jump(x) :- Sport(x, run, 'Y', swim, throw, sync).	5 Run(x) :- Sport(x, 'Y', jump, swim, throw, sync).
6 Run(x) :- Sport(x, 'Y', jump, swim, throw, sync).	6 Handball(x) :- Run(x), Throw(x).
7 Swim(x) :- Sport(x, run, jump, 'Y', throw, sync).	7 Sync(x) :- Sport(x, run, jump, swim, throw, 'Y').
8 Throw(x) :- Sport(x, run, jump, swim, 'Y', sync).	8 Jump(x) :- Sport(x, run, 'Y', swim, throw, sync).
9 Sync(x) :- Sport(x, run, jump, swim, throw, 'Y').	9 Swim(x) :- Sport(x, run, jump, 'Y', throw, sync).
10 Handball(x) :- Run(x), Throw(x).	10 }
11 500M(x) :- Run(x), not Jump(x).	11 P2 = {
12 500MHurdle(x) :- Run(x), Jump(x).	12 500M(x) :- Run(x), not Jump(x).
13 SynchronizedSwimming(x) :- Swim(x), Jump(x), Sync(x).	13 Duathlon(x, y) :- Run(x), not Handball(x), Swim(y), not SynchronizedSwimming(y).
14 Duathlon(x, y) :- Run(x), not Handball(x), Swim(y), not SynchronizedSwimming(y).	14 }
15 MultiSport(x, y) :- Run(x), not Duathlon(x, y), Jump(y).	15 P3 = {
	16 MultiSport(x, y) :- Run(x), not Duathlon(x, y), Jump(y).
	17 }

## Programme 11 :

program11.dl	program12.dl	11.txt	12.txt
1 s(a,b).		1 P1 = {	
2 s(b,c).		2 r(X, Y) :- s(X, Y).	
3 r(X,Y) :- s(X,Y).		3 r(X, Y) :- r(X, Z), s(Z, Y).	
4 r(X,Y) :- r(X,Z), s(Z,Y).		4 t(X, Y) :- r(X, Y), not s(X, Y).	
5 t(X,Y) :- r(X,Y), not s(X,Y).		5 }	
6		6	

## Programme 12 :

program12.dl ×	12.txt ×
1 Pred(X).	1 P1 = {
2 A(x):- Pred(x).	2 A(x) :- Pred(x).
3 B(x):- not A(x).	3 }
4 C(x):- not B(x).	4 P2 = {
5 D(x):- not C(x).	5 B(x) :- not A(x).
6 E(x):- not D(x).	6 }
7 F(x):- not E(x).	7 P3 = {
8 G(x):- not F(x).	8 C(x) :- not B(x).
9 H(x):- not G(x).	9 }
10 I(x):- not H(x).	10 P4 = {
11 J(x):- not I(x).	11 D(x) :- not C(x).
12 K(x):- not J(x).	12 }
13 L(x):- not K(x).	13 P5 = {
14 M(x):- not L(x).	14 E(x) :- not D(x).
15 N(x):- not M(x).	15 }
16 O(x):- not N(x).	16 P6 = {
17 P(x):- not O(x).	17 F(x) :- not E(x).
18 Q(x):- not P(x).	18 }
19 R(x):- not Q(x).	19 P7 = {
20 S(x):- not R(x).	20 G(x) :- not F(x).
21 T(x):- not S(x).	21 }
22 U(x):- not T(x).	22 P8 = {
23 V(x):- not U(x).	23 H(x) :- not G(x).
24 W(x):- not V(x).	24 }
25 X(x):- not W(x).	25 P9 = {
26 Y(x):- not X(x).	26 I(x) :- not H(x).
27 Z(x):- not Y(x).	27 }
	28 P10 = {
	29 J(x) :- not I(x).
	30 }
	31 P11 = {
	32 K(x) :- not J(x).
	33 }
	34 P12 = {
	35 L(x) :- not K(x).
	36 }
	37 P13 = {
	38 M(x) :- not L(x).
	39 }
	40 P14 = {
	41 N(x) :- not M(x).
	42 }
	43 P15 = {
	44 O(x) :- not N(x).
	45 }
	46 P16 = {
	47 P(x) :- not O(x).
	48 }
	49 P17 = {
	50 Q(x) :- not P(x).
	51 }
	52 P18 = {
	53 R(x) :- not Q(x).
	54 }