Rapport TP5 IAGL

ENGEL Arthur, TRINQUART Matthieu, SANCHEZ Martin

Choix du jeux de données

Nous avons choisi les jeux de données suivants parmi ceux proposés sur le moodle :

— Comparison of Object Database Management Systems (première matrice) [1]

— Comparison of Free and Open Source Software licenses (première matrice) [2]

— Comparison of Programming Languages (première matrice) [3]

Sélection des données

Voici comment nous avons procédé pour sélectionner les données.

Comparison of Object Database Management Systems

Pour les databases, nous avons supprimé quelques bases de données pour diverses raisons :

— ConceptBase : Parce que nous n'avions pas d'information sur le langage Telos ce qui nous posait problème pour le CSV Language.

— Db4o : Parce qu'elle utilisait une licence "Custom", et que nous n'avions pas d'information sur cette licence.

— Zope Object Database : Car elle utilisait la "Zope Public License", et qu'il y a très peu d'information sur celle-ci dans le CSV licence.

Nous en avons également enlevé d'autres de façon arbitraire afin d'en avoir moins de quinze, en s'efforcant de garder une bonne diversité des données.

1. https ://en.wikipedia.org/wiki/Comparison_of_object_database_management_systems
2. https ://en.wikipedia.org/wiki/Comparison_of_free_and_open-source_software_licenses
3. https ://en.wikipedia.org/wiki/Comparison_of_programming_languages

Dans notre fichier CSV nous avons gardé les informations sur le "SQL support". La ligne "langage" fournit des informations qui ne sont pas des langages de programmation à proprement parler (.NET, XML) nous avons donc créé des colonnes spécifiques à ceux-ci dans le csv des BDD (et non celui des langages).
On remarque aussi qu'il y a environ autant de DBMS qui utilise des licences libre que de licences propriétaire, cela pourrait servir de point de comparaison.

Comparison of Free and Open Source Software licenses

Pour les licences, nous avons gardé toutes les licences utilisées par les DBMS que nous avons gardés. Nous en avons gardé trois de plus (Eclipse Public License, Mozilla Public License, Cryptix General License) nous semblant suffisamment différentes des autres afin de pouvoir comparer les licences utilisées par les DBMS avec d'autres, et pour avoir une plus grande variabilité des données. Nous avons également ajouté nous-même une ligne pour les licences propriétaires, pour pouvoir comparer les programmes propriétaires et libres. Nous partons du principe que les licences propriétaires ne donnent aucun droit donné par les licences open-source, hormis le "Patent grant" et le "TM grant".

Comparison of Programming Languages

La matrice originale faisait 124 lignes par 11 colonnes ; il était donc nécessaire de réduire cette quantité de données. Dans un premier temps nous avons réduit le nombre de lignes, en supprimant en priorité les langages peu connus et qui n'étaient pas dans les autres matrices. Nous avons ainsi conservé 15 lignes, avec parmi celles-ci des langages non utilisés par les DBMS, afin de pouvoir établir une comparaison.
Ensuite nous avons rajouté quelques lignes car nous avions besoin d'informations sur certains langages comme : Picolisp et Typescript.
Nous avons considéré que Typescript était équivalent a Javascript.
Pour Picolisp, nous sommes allés chercher les informations sur l'article wikipédia[4] du langage. Nous avons vu sur cet article que le langage était procédural (entre autres). Plus tard dans notre travail, nous avons appris après quelques recherches que le paradigme procédural était un paradigme dérivé de l'impératif, ce qui n'apparaissait pas sur le treillis. En effet, l'article sur le Picolisp n'indiquait pas l'impératif comme étant un paradigme de ce langage, nous l'avons donc rajouté nous-même.

4. https ://en.wikipedia.org/wiki/PicoLisp

Nettoyage des données

Voici comment nous avons procédé pour nettoyer les données.

Comparison of Object Database Management Systems

Pour binariser les données nous avons créé une section par caractérisation. Pour SQL_support nous avons créé une colonne par type de support (SQL superset, JPA, None...) et si la database avait ce support nous mettions 1 sur la ligne et 0 si la database ne l'avait pas.

Pour XML et .NET nous utilisons le même principe, nous créons une colonne chacun et mettons 1 dans la case si la Database utilise cette technologie et 0 sinon. La matrice de langage n'est pas dans BDD.csv mais dans BDD2Languages.csv qui chaque fait la liaison entre le csv BDD et Langages. Nous utilisons le même principe pour les licences qui sont dans le fichier BDD2Licences.csv qui fait la liaison entre BDD et Licences.

Comparison of Free and Open Source Software licenses

Pour le nettoyage des données liées aux licences, nous avons fait fit des colonnes Author et Latest version, qui, en plus de ne pas être binarisables, ne semblaient pas nous apporter d'informations utiles. Nous avons également fait le choix de ne pas utiliser la colonne Publication Date (que nous aurions pu binariser en créant des colonnes englobant des périodes par exemple), car nous ne jugions pas cette colonne utile non plus.
Pour la binarisation des données que nous avons gardées, voici comment nous avons procédé :

- Toutes les cellules vides (”?”) ont été changées en 0.

- Pour ”Linking”, ”Distribution”, ”Modification”, ”Sublicensing” : Ces colonnes correspondent à des valeurs similaires dans les cellules de la matrice, nous les avons donc binarisées de la même manière. Elles ont toutes un cas particulier, le copyleft. Il permet une liberté totale vis-à-vis de la colonne concernée, sous réserve de redistribuer le programme utilisant le programme source sous les mêmes conditions. Pour illustrer ce cas particulier, nous avons divisé chacune de ces quatre colonnes en deux, une version ”Permissive” et une version ”Copyleft”.
  - Les cellules ”vertes” de l'article wikipédia (”Permissive”, ”Public Domain”) ont été changées en 1 pour la colonne ”Permissive” et 0 pour la colonne ”Copyleft”

correspondante.
- Les cellules ”jaunes” (”With restrictions”[5], ”No network use”, ”Limited”..) en 0 dans la colonne ”Permissive” et 0 dans la ”Copyleft”, car dans une optique ”puriste”, nous considérons que ce genre de restriction ne constitue pas un usage libre.
- Il reste les cellules ”bleues”, qui correspondent à un copyleft ou une variante (généralement, un copyleft restreint à une licence en particulier) de celui-ci. Ici nous mettons 0 dans la colonne ”Permissive” et 1 dans la ”Copyleft”.

— Pour ”Patent grant” et ”TM grant”, nous avons eu un cas de figure similaire au ”copyleft”, nous avons des grant ”Yes”, ”No”, et ”Manually”. Nous avons donc séparé chacune de ces deux colonnes en deux, une ”Manual” et une ”Automatic”, pour marquer cette subtilité.
- Les cellules ”Yes” de l'article ont été changées en 1 dans la colonne ”Manual” et 0 dans la colonne ”Automatic” du csv.
- Les cellules ”No” ont été changées en 0 dans la colonne ”Manual” et 0 dans la colonne ”Automatic”.
- Les cellules ”Manually” ont été changées en 0 dans la colonne ”Manual” et 1 dans la colonne ”Automatic”.

— Pour ”Private Use”, les cellules ”vertes” ont été changées en 1, et la cellule ”no network use” a été changée en 0. Encore une fois dans une optique ”puriste”, nous considérons qu'une utilisation partielle n'est pas libre.

Comparison of Programming Languages

Nous avons retiré les colonnes textuelles, car en plus de ne pas être facilement binarisables, elles n'apportent pas beaucoup d'informations. Ensuite nous avons binarisé ce qui restait du fichier, cette matrice contenait déjà en grande partie des booléens donc la binarisation était triviale. Pour la colonne ”Standardized” nous avons décidé de choisir 1 dans le cas où c'est standardisé, peu importe le standard et 0 dans le cas contraire, car nous ne jugions pas ces distinctions comme étant très utiles.

5. Il y a une des cellules ”With restrictions” en bleu, il s'agit d'un cas particulier, qui rentre dans le cadre du copyleft

Modélisation de nos fichiers

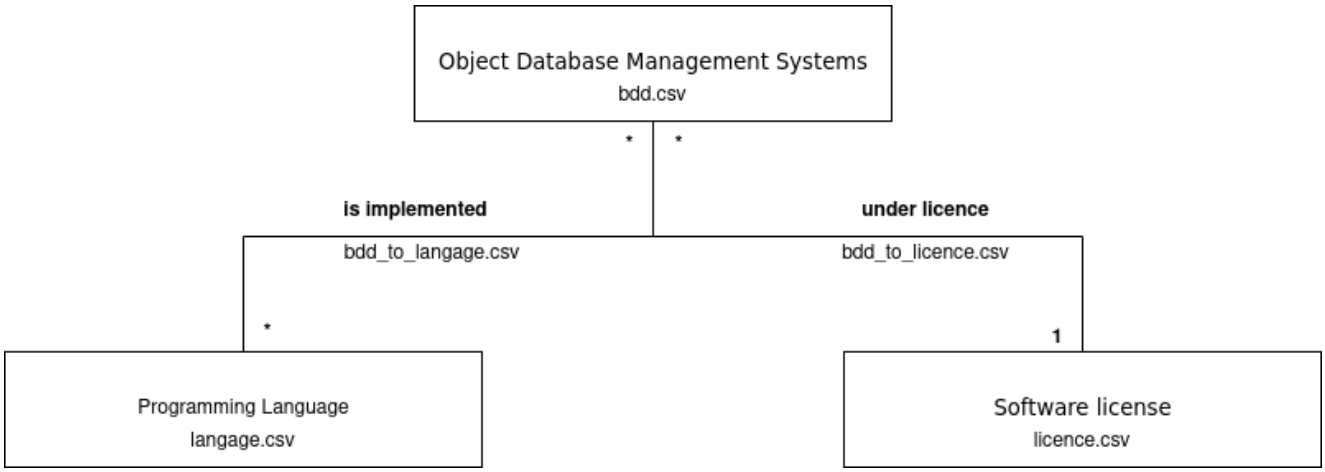

FIGURE 1 – Modélisation uml entre nos différents CSV

Ce schéma montre la dépendance entre nos différents fichiers CSV, en effet on remarque que "Object Database Management System" est l'élément central. Dans ce fichier on fait référence a des Licences (Software Licence) et à des langages (Programming Language).

Treillis obtenu par FCA4j

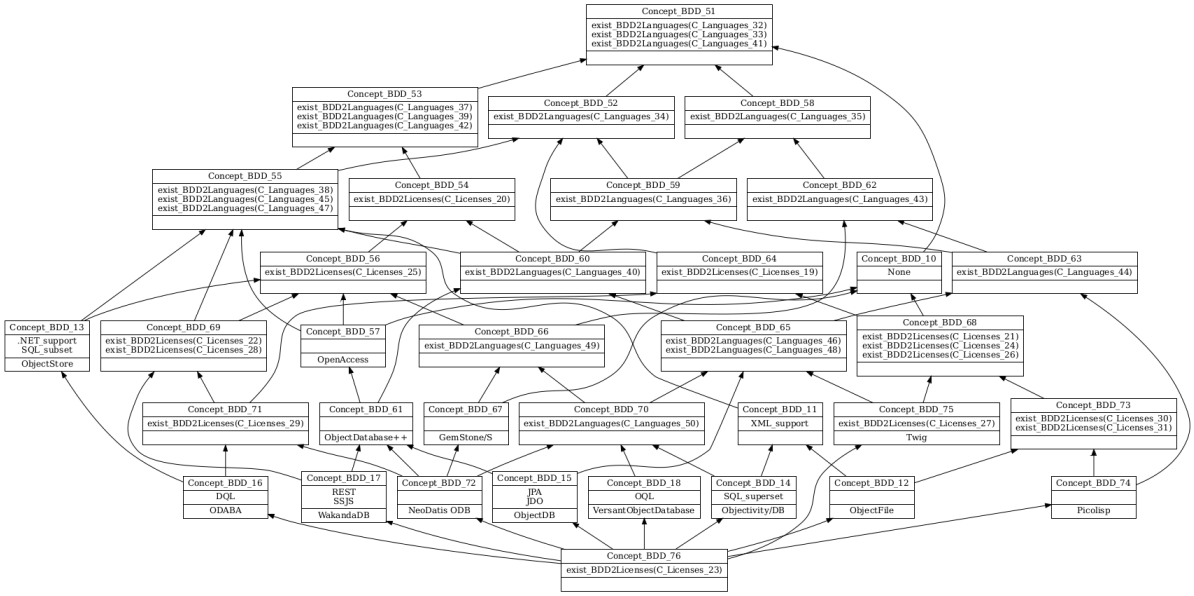

FIGURE 2 – Treillis DBMS

Le treillis met en évidence plusieurs aspects sur nos données. La première chose que met en valeur de treillis est que tous les DBMS sélectionnées utilisent un langage utilisant le paradigme orienté objet ; ce qui est logique, car nos données concernent des bases de données orientées objet.
Une autre information que met en évidence le treillis est que tout les DBMS utilises des langage procéduraux, ce qui par transitivité implique qu'elles utilisent des langages impératifs.

On voit bien qu'aucune DBMS n'utilise les licenses Eclipse Public License et Mozilla Public License, car elles sont situées tout en bas du treillis, sans DBMS associée.

On voit que les différents concept de licences se situent au milieu ou en bas du treillis et au contraire les différents paradigme de langage ce situe plus vers le haut du treillis. Cela est sûrement dut au fait que le développement d'un DBMS entraine des contraintes technologiques, alors que le concept de licence est complètement découplée du développement de celle-ci. Cela est également lié au fait que beaucoup des DBMS présents dans nos données utilisent le C++. Cependant, cela ne remet pas nécessairement la déduction précédente en cause, C++ est un choix de langage multi-paradigmes à haute performance

très populaire.

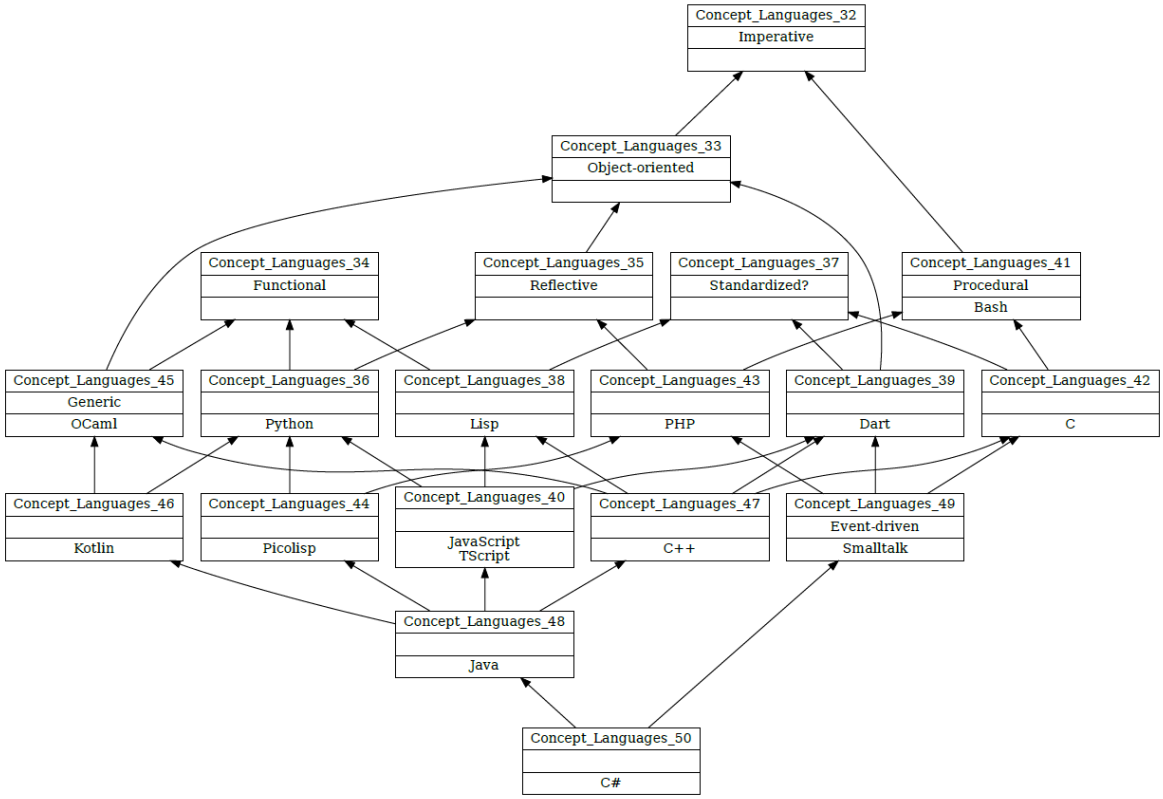

FIGURE 3 – Le treillis généré par FCA4J pour language.

On voit sur le treillis qu'à l'exception de LISP, tous les langages sont impératifs. Il s'agit du paradigme le plus courant.

On voit également que la présence du paradigme orienté objet implique celle du paradigme impératif. Cela nous semblait logique, car en programmation orientée objet, les objets maintiennent un état, ce qui nous semblait incompatible avec un paradigme fonctionnel par exemple. Cependant, en regardant l'article wikipédia comparant les langages, nous avons trouvé des langages objets qui ne sont pas impératifs. Il s'agit donc d'un hasard de nos données et non pas de la réalité.

On remarque aussi que les langages utilisés par les bases de données se trouvent surtout vers le bas du treillis, ce qui pourrait vouloir dire qu'on a besoin d'un certain nombre de paradigmes pour construire un système de bases de données. On voit ici que, si on s'en tient strictement aux paradigmes, Objectivity/DB pourrait utiliser Kotlin au lieu de Python par exemple, car Kotlin permet d'utiliser les mêmes paradigmes que Python, en plus de la programmation générique. D'un autre coté utiliser un langage comme Bash serait impossible, car il n'implémente pas le paradigme orienté objet.

Si l'on s'en tient strictement aux treillis, on ne pourrait pas utiliser Dart pour développer un DBMS, car tous les DBMS de notre jeu de données utilisent un langage possédant les paradigmes impératifs, procéduraux, et orientés objets. Cependant il est plus probable qu'il s'agisse d'un problème d'échantillon peu représentatif que d'un vrai obstacle technologique : alors qu'il nous semble logique qu'un DBMS objet utilise un langage ayant un paradigme orienté objet, la même déduction pour les langages procéduraux ne nous semble pas aussi évidente.

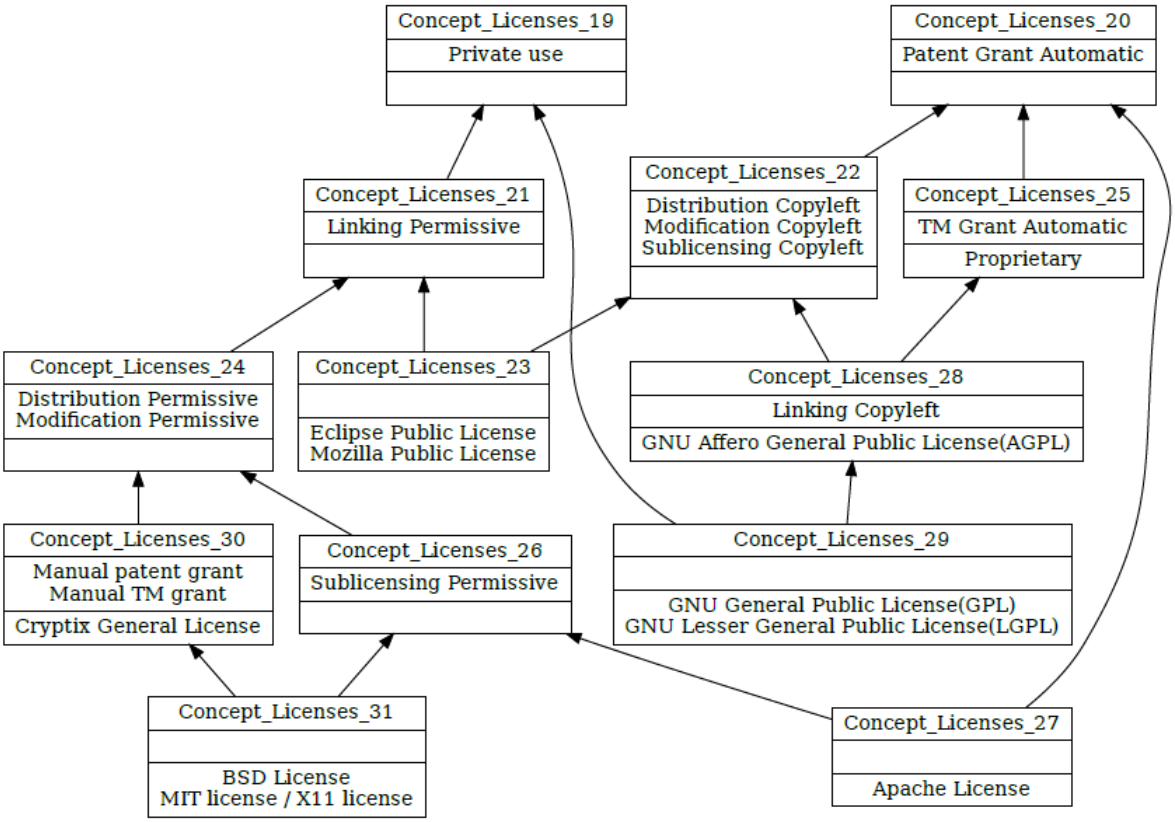

FIGURE 4 – Le treillis généré par FCA4J pour licences.

Sur ce treillis on observe une tendance : les licences libres permettent optent généralement pour une approche soit tout copyleft soit toute permissive. Les seules exceptions sont l'Eclipse Public Licence et la Mozilla Public Licence, qui ont un linking permissif. Cela a du sens car le linking est un peu particulier, il s'agit simplement d'utiliser le programme comme une librairie. Il s'agit d'une concession moins "lourde" que la distribution, la modification, ou le sous-licenciement en quelque sorte.

On observe aussi que La Cryptix General License est la seule licence libre à ne pas proposer de sous-licenciement, car l'information n'était pas présente dans les données originales, et a donc été interprétée comme un 0.

On voit bien la démarcation entre les caractéristiques mutuellement exclusives : les "permissive" et les "copyleft" sont séparés de part et d'autre du treillis, de même que les "automatic" et les "manual".

On observe que la licence "Proprietary" se retrouve naturellement vers le haut du

treillis, car il s'agit de la licence la moins permissive de toutes.

Feature Model

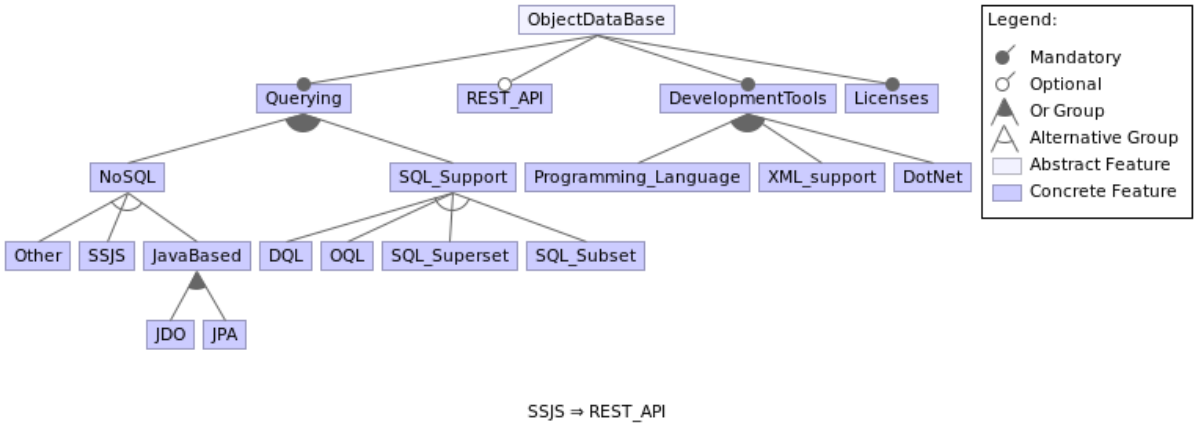

FIGURE 5 – Le feature model pour les BDD.

Pour le Feature Model nous avons mis en évidence que toutes les DataBase devait forcément contenir :

- DevelopmentTools représente les technologies utilisées pour développer la DataBase, il englobe le langage de programmation utilisé (lien avec le feature Model Programming_Language), et éventuellement XML et .NET, que nous avons séparés de Programming_Language car ce ne sont pas des langages de programmation à proprement parler.
- Licences qui fait le lien avec le FeatureModel des Licences.
- Querying qui indique le langage de querying supporté par la DataBase. Si la DataBase a un support SQL alors il contient soit DQL, OQL, SQL_Superset ou SQL_Subset. Si c'est une dataBase NoSQL alors soit il contient JavaBased (qui contient soit JDO ou JPA), SSJS ou d'autres formes de NoSQL. SSJS à une contrainte sur RESTAPI car nous avions remarqué que sur nos données que le support de SSJS est relié à la présence d'une API REST.

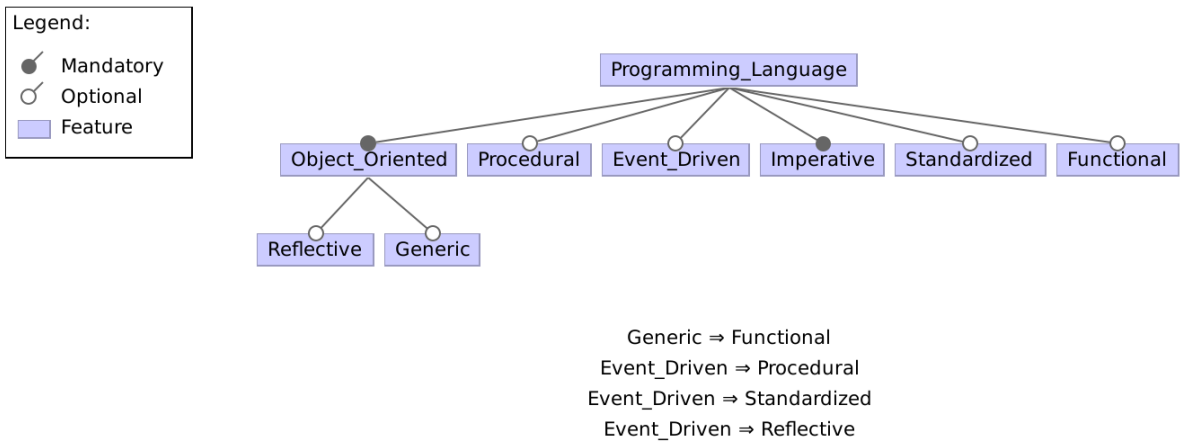

FIGURE 6 – Le feature model pour les langages.

Ce Feature Model nous montre les différentes catégorisation des langages de programmation. On remarque qu'un langage est forcément impératif et orienté objet, cela ne représente pas la réalité de tous les langages de programmation, en revanche dans le cas des bases de données objet c'est nécessaire. Nous n'avons pas mis procédural en obligatoire pour les raisons énoncées ci-dessus.
Les languages a objet génériques sont fonctionnel, encore une fois, cela est spécifique à nos données. Les langages événementiels sont procéduraux, standardisés et réflectif (donc objet), encore une fois, certainement un hasard de nos données.

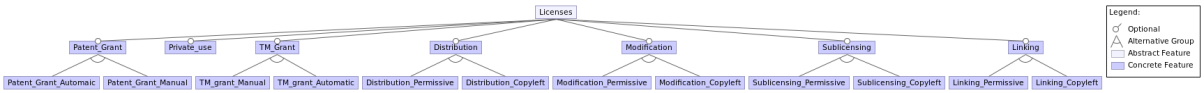

FIGURE 7 – Le feature model pour les licences.

Ce Feature Model des licences est très peu restrictif. En effet, les différentes caractéristiques d'une licence ne sont pas intrinsèquement liées, la définition des termes d'une licence est très libre.

Nous avons donc représenté chaque caractéristique d'une licence comme étant optionnelle, en fonction de si on veut l'autoriser ou non, et nous avons effectué le même découpage que dans notre csv pour chacune d'entre elles. Ce découpage présentant des caractéristiques mutuellement exclusives, nous avons fait le choix de les représenter avec un XOR.

Nous parlions tout à l'heure du fait que le linking était une concession moins importante que la modification ou la distribution par exemple, et pourrions donc raisonnablement assumer que Modification -> Linking par exemple. Cependant nous avons fait le choix de garder ce Feature Model le plus flexible possible.