

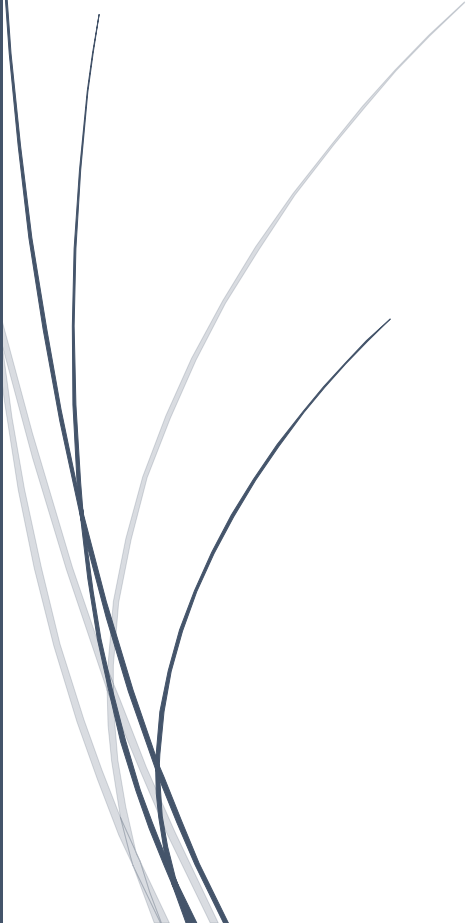
A dark blue vertical bar on the left side of the page. A blue arrow points to the right from the bar, containing the text "2015-2016".

2015-2016

PSAR

Rapport Semaine 25 janvier au 03 février 2016

Interface graphique pour la
logique du 1^{er} Ordre

Several thin, curved lines in dark blue and light grey originate from the bottom left and curve upwards and to the right.

Diallo Ousmane 3
Katuitshi-Ntumba Jean-Marc

1. Présentation

Introduction :

Le projet "**Interface graphique pour la Logique du 1^{er} Ordre**" consiste à développer un outil dynamique et robuste pour améliorer l'enseignement de la logique en Licence 3. Ce projet nous a été soumis dans le cadre de L'UE PSAR du master 1 Informatique spécialité **SAR**.

Il est sous la responsabilité de Mr Fabrice Kordon et suivi par Mme Béatrice Berard, Mr Mathieu Jaume et Mme Bénédicte Legastelois.

Nous venons par ce document vous présenter notre première approche du sujet.

Problématique :

Ce projet consiste à développer une application pour la mise en pratique de la logique du 1er ordre en licence 3 à l'UPMC.

Il convient de noter que cela sera implémenté en composants (couches) que nous détaillerons dans le cahier de charges.

Méthodes de travail

Nous avons décidé pour chaque couche de discuter et partager nos idées, avant de nous répartir le travail. Ensuite, après que chacun de nous ait fini, de travailler sur un composant de l'application, ou sur une classe, nous passerons à la mise en commun et à la réalisation des différents tests à différent niveau d'abstraction afin de nous assurer de gérer au mieux les incohérences et les bugs, nous utiliserons l'outil Git pour lequel nous avons une bonne maîtrise générale.

- *Gestion du projet*

Nous utiliserons Git pour tout ce qui est gestion de version et synchronisation du projet.

- *Gestion des documents dans le projet*

Nous essayerons dans la mesure du possible de travailler en latex tous les documents que nous produirons dans le cadre du projet (cahier de charges, rapport d'avancement,...) seront donc en latex

Organisation du projet

Le projet est structuré en modules ou couches indépendants :

1-Module d'analyseur syntaxique des formules

L'analyseur syntaxique sera un module du projet qui s'occupera de la vérification syntaxique des différentes formules qui seront soumises à une évaluation éventuelle.

Une fois que l'utilisateur (l'étudiant) saisie sa formule et demande une interprétation, nous l'appellerons en premier pour s'assurer que la formule est bien formée avant d'appeler notre interpréteur.

2-Module Interface graphique

Toute l'application sera présentée dans une interface conviviale, dans laquelle l'utilisateur aura la possibilité de construire ses propres formules, son propre environnement et demander l'évaluation de ses formules.

Il est évident que l'interface offrira un menu standard dans lequel il pourra sauvegarder/restaurer son environnement, ses formules dans/depuis le répertoire de son choix.

Notre choix du langage tient compte en premier lieu du temps de réponse et de la robustesse.

3-Module évaluant les formules(Moteur)

Un module d'interprétation déjà existe en python, il faudra faire communiquer les deux modules pour obtenir l'interprétation de(s) la formule(s).

Le module python attend deux paramètres :

- l'environnement bien définie
- la formule bien formée

Choix du langage de Programmation

Comme le moteur est développé en python, nous sommes partis de l'idée de faire communiquer deux langages car implémenter une interface graphique en python s'avère fastidieux.

Résultats des recherches

Notre application devra être implémentée pour les deux modules restants en java, il nous fallait donc trouver le Framework ou le middleware qui convient ;

Le choix du middleware ou du Framework dépend fortement :

Du temps de réponse : faire communiquer deux langages de programmation est toujours couteux en performance.

De la facilité de la prise en main : Pour notre projet nous avons voulu consacrer un minimum de deux semaines pour la prise en main.

Faire Communiquer java et python

La Bibliothèque **Jython** permet d'exécuter en Java un programme écrit en python, Il existe aussi **jeep** qui fait de même.

Désavantage : Il y a en effet une lenteur de l'interpréteur Jython. La lenteur n'est pas dans l'ordre des nanosecondes mais bien des secondes.

CORBA, ICE et ses variantes sont de langage de définitions d'interfaces (IDL) visant à faire communiquer deux entités (services, composants) écrit dans des langages différents ; Il s'agit de mapper (traduire des éléments fournis par l'IDL en éléments d'un langage de programmation).

Nous utiliserons cette deuxième approche au mieux dans le cadre de ce projet.

2. Résumé du Cahier fourni

Après une première analyse du document fourni, nous vous présentons dans cette section le résumé de ce que nous avons compris du projet.

Terme :

Un terme est défini de façon inductive comme suit :

- Toute variable est un terme
- Toute fonction de variable d'arité est un terme

Nous manipulerons en tout cinq (5) variables $\{v,w,x,y,z\}$ et 20 constantes $\{a,b,c,...,t\}$ dans notre projet.

Interprétation des termes :

Interpréter un terme, c'est lui associé une valeur appartenant à un domaine.

- Si le terme est une variable, c'est la valeur associée à la variable
- Sinon c'est la valeur globale résultant de l'évaluation des différents arguments du terme

Interprétation des termes du projet :

Dans ce projet les objets qu'on doit manipuler seront des fleurs qui ont les caractéristiques suivantes :

- espèce = {rose, paquerette, tulipe}
- taille = {grand, moyen, petit}

- couleur = {rouge, rose ,blanche}
- nom {none}

une fleur dans le jardin est représenté par un quintuplés $\{(x,y),e,t,c,n\}$ où (x,y) est sa position dans le jardin.

Construction d'une structure à partir du jardin:

Il est défini une structure M comprenant tous les coules (x,y) de :

- toutes les places contenant des fleurs
- sinon Error

Formule de la logique des prédicats:

Un Symbole de prédicat correspond au nom d'une propriété sur un terme.

Implémentation des formules

Les opérateurs dans le cadre du projet : négation, et, ou, implique, pour tout, il existe.

Une variable est libre si elle n'est associé à aucun quantificateur (existentiel ,pour tout), dans le cas de notre projet tous nos variables sont des variables liés.

Formules du projet

Il existe des formules unaire, binaires et ternaires :

P1-formule unaire

P2-formule binaires

P3-formule ternaire

P1= {est_rouge, est_rose, est_blanc, a_l_est, a_l_ouest, au_sud, au_nord , Rose, Paquerette, Tulipe, est_grand, est_moyen, est_petit}

P2={a_l_est_de, a_l_ouest_de, au_sud_de, au_nord_de, même_latitue, même_longitude ,plus_grand_que, plus_petit_que, même_taille_que, même_couleur_que}

P3 = {est_entre}

Interprétation des formules

Prédicat	Définition	Relation
Espèce	Rose,paquerette,Tulipe	Unaire
Taille	est_grand,est_moyen,est_petit	Unaire
Couleur	est_rouge,est_rose,est_blanc	Unaire
Position	a_l_est,a_l_ouest,au_sud,au_nord	Unaire
Comparaison de position	a_lest_de,a_l_ouest_de,au_sud_de,au_nord_de,même_latITUDE, même_longitude	Binaire
Comparaison de taille	Plus_petit_que,plus_grand_que,même_taille_que	Binaire
Comparaison de couleur	même_couleur_que	Binaire
Égalité	0=	Binaire
Comparaison de position	est_entre	Ternaire

NB : Toutes les formules soumises à la vérification sont des formules closes (une erreur est donc déclenchée lors de l'évaluation d'une formule avec une variable libre).

3. Conclusion

Il a été question dans ce support de faire un bilan d'où nous en sommes dans l'avancement du projet et quelles sont les pistes de solutions explorées afin de répondre aux questions les plus complexes qui dans notre cas était le choix d'un langage de programmation offrant la possibilité de converser avec du python.

Il ne remplace en rien le cahier de charges mais nous permet juste de vous exposer nos difficultés et les raisons de nos choix pour le développement de l'application.