Platinum Parsing
Travail de Bachelor

Patrick Champion
Prof. François Birling

Printemps 2017
HEIG-VD

TODO: inclure logo ici

Table des matières

1 Introduction

TODO: à faire à la fin
(dans ce document, PP = Platinum Parsing)

2 Études des cas d'utilisation

Afin d'analyser les fonctionnalités nécessaires de PP, nous allons utiliser une approche par cas d'utilisation. La première étape de cette analyse est de définir quels seront les acteurs qui auront besoin d'utiliser notre outil :

- le **concepteur** de langage veut pouvoir définir sa grammaire EBNF [1], ainsi que la vérifier (erreur de syntaxe, ambiguïté, etc.)
- le **développeur** de compilateur veut pouvoir se baser sur une grammaire EBNF pour construire facilement les règles de production. Il souhaite également avoir le contrôle sur la sortie produite, cela à l'aide d'un langage riche
- le **mainteneur** du compilateur veut pouvoir apporter des modifications/améliorations au sans devoir revoir tout le processus depuis le début
- l'**utilisateur** du compilateur veut pouvoir l'utiliser simplement, celui-ci devrait aussi optionnellement fournir des options de compilation lui offrant un certain contrôle

À partir de ces acteurs, nous pouvons construire un diagramme des cas d'utilisation de PP :

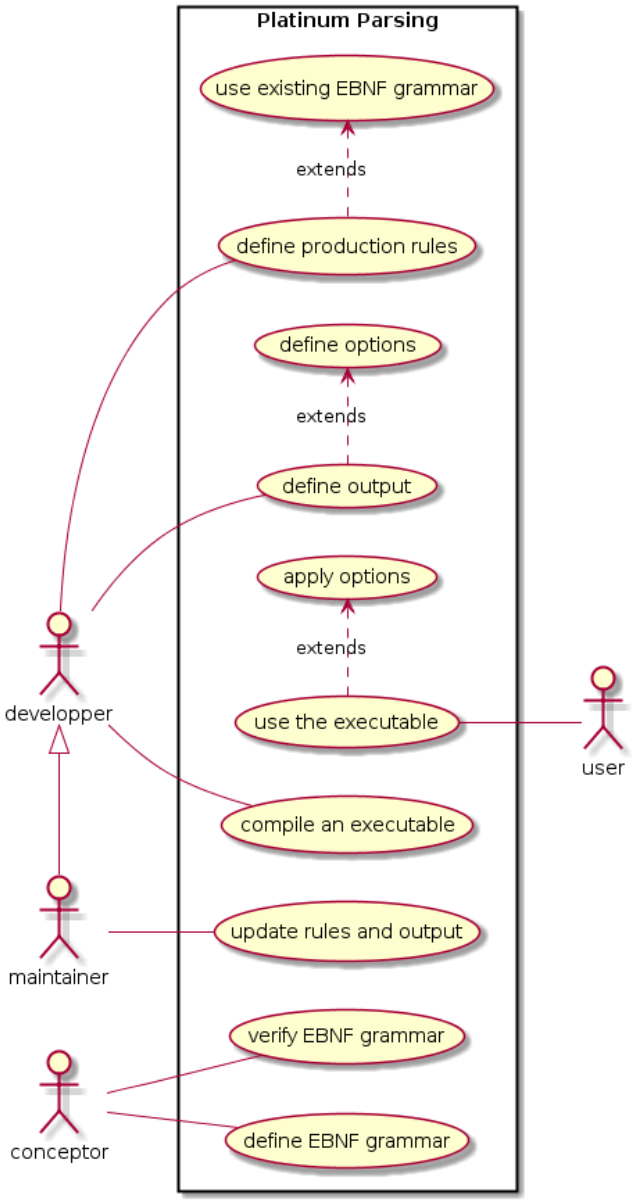

FIGURE 1 – Diagramme des cas d'utilisation

2.1 Cahier des charges

Le diagramme ci-dessus nous offre un bon aperçu des fonctionnalités que nous devons implémenter. Nous allons cependant préciser par écrit celles-ci afin d'avoir un cahier des charges complet. Dans cette section les fonctionnalités utilisant *doit* sont obligatoires, celles utilisant *devrait* sont optionnelles.

TODO: faire CC

3 Technologies

3.1 Choix du langage

Un choix crucial dans ce projet est le choix du langage de programmation que nous allons utiliser pour implémenter notre solution. Plusieurs langages ont été choisis en début de projet pour être comparés :

— C#
— F#
— Haskell

Pour choisir entre ces différentes options, nous allons nous baser sur plusieurs critères pour lesquels chaque langage sera mis en compétition. Les différents critères seront pondérés en fonction de leur importance dans ce projet, chaque langage obtiendra une note pour les critères (de 0 à 3) et c'est la moyenne finale qui nous donnera une bonne information du potentiel de chaque langage :

No	**Critère**	**Poids**
	Critères fonctionnels	
1.1	Possibilités du paradigme	3
1.2	Bibliothèque(s) utile(s)	3
1.3	Quantité de code nécessaire	2
1.4	Portabilité	2
	Critères de développement	
2.1	Communauté active	3
2.2	Ressources en ligne	3
2.3	Outils de développement	1
	Critères de performances	
3.1	Vitesse d'exécution	3
3.2	Mémoire utilisée	2

FIGURE 2 – Critères pour le choix du langage

Les détails des critères testés se trouvent dans l'annexe 6.3 :

No	**C#**	**F#**	**Haskell**
-	-	-	-

FIGURE 3 – Notes des langages pour les différents critères

Les notes finales sont :

1.
2.
3. .

Le choix est donc... TODO: faire choix

3.2 Cadre de développement

4 Architecture du projet

4.1 Décomposition en sous-projets

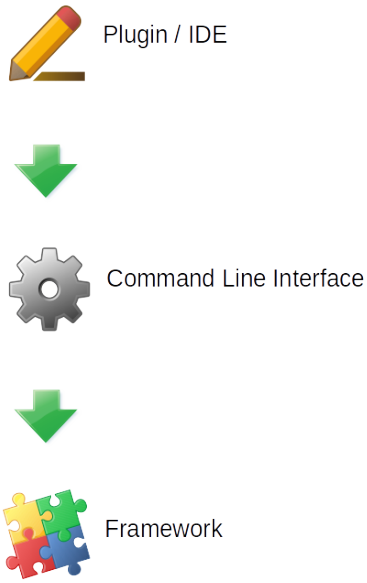

FIGURE 4 – Sous-projets de PP

TODO: expliquer schéma

4.2 Processus de transformation

PP est un outil qui peut se décomposer en plusieurs étapes successives (un *pipeline*), chacune prenant des données en entrée et fournissant d'autres données en sortie pour la prochaine étape du processus. Dans cette section nous allons expliciter quelles sont ces étapes ainsi que les données intermédiaires :

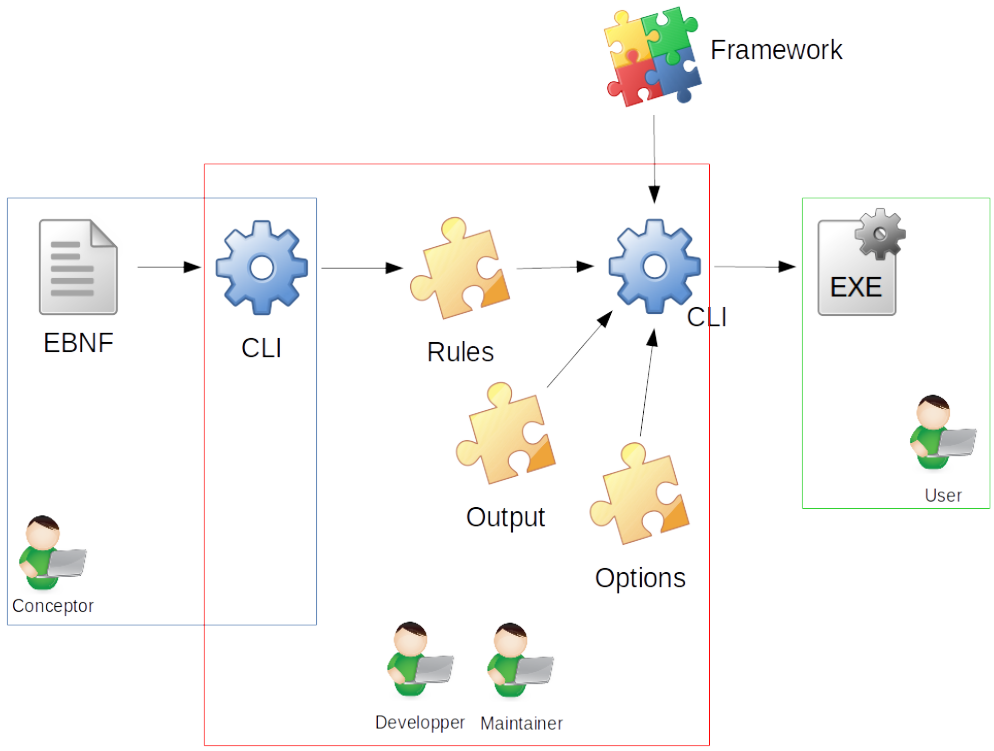

FIGURE 5 – Pipeline de transformation

TODO: expliquer schéma

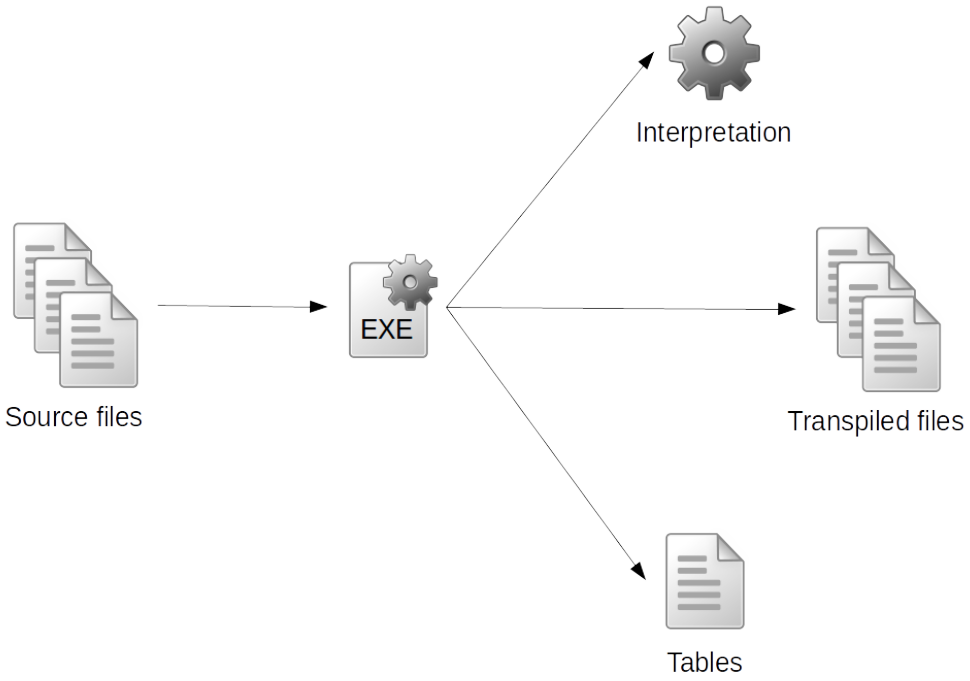

FIGURE 6 – Sorties possibles de l'exécutable final

TODO: expliquer schéma

5 Conclusion

TODO: à faire à la fin

6 Annexes

6.1 Énoncé original

Pour faciliter l'écriture de compilateurs, ce travail de diplôme vise à réaliser un outil complet avec un environnement de développement intégré, permettant :

- d'éditer une grammaire selon une syntaxe de type BNF
- d'analyser automatiquement cette BNF, de détecter les erreurs de syntaxe, incohérences et conflits de règles.
- de générer dans n'importe quel langage de programmation cible, sur la base d'un template textuel fourni par l'utilisateur, les tables pour le lexer et le parser LALR.

Ce travail de diplôme vise à développer un outil capable de remplacer "Gold Parsing System", un outil libre intéressant, mais malheureusement fortement pénalisé par des bugs récurrents. L'IDE sera réalisé sous la forme d'un plugin du logiciel oStudio d'Objectis, avec les technologies très en vogue C# et WPF. Le générateur de compilateurs sera mis en ligne sur Internet en libre accès, permettant à la communauté Internet de bénéficier gratuitement de cette technologie.

6.2 Table des figures

6.3 Critères testés pour le choix du langage

6.4 Références

[1] EBNF, Extended Backus-Naur form,
`https://en.wikipedia.org/wiki/Extended_Backus%E2%80%93Naur_form`