

Prolog - Projet

Université Paris I. Maîtrise de Logique. Année 2004-2005.
Nicolas Stroppa - stroppa@enst.fr

1 Introduction

1.1 Conseils

Pour que le projet se déroule au mieux, il peut être utile de suivre quelques recommandations simples. En particulier, votre code doit être *lisible* par une personne extérieure au projet. Il doit donc être commenté : indiquez quel est l'objectif de vos procédures, ce qu'elles attendent en entrée, ce qu'elles renvoient, si certains paramètres d'entrées/sorties peuvent être échangés (réversibilité), etc. Quand il y a plusieurs possibilités, choisissez la solution la plus simple, n'oubliez pas que vous devrez savoir expliquer le fonctionnement du programme...

En outre, n'hésitez pas à consulter des documentations disponibles en ligne.

- la documentation de Swi-Prolog : <http://hcs.science.uva.nl/projects/SWI-Prolog/Manual/>
- un tutoriel : <http://www.amzi.com/AdventureInProlog/>

Et la documentation hors ligne :

- Prolog Programming for Artificial Intelligence. Ivan Bratko. 3rd edition, 2001. Addison-Wesley.

1.2 Le sujet

Le but du projet est de pouvoir identifier les phrases correctes d'une langue qui est un français très très simplifié et de les analyser le cas échéant. Pour des raisons de simplicité, une phrase sera modélisée par une liste de constante, chaque constante désignant un mot. Ainsi, la phrase *Jean parle avec Marie* sera représentée en Prolog : `['jean', 'parle', 'avec', 'marie']`. De même, on opérera un certain nombre de simplifications pour éviter des problèmes : *aujourd'hui* deviendra *aujourd'hui*, *l'* deviendra *le* ou *la* selon le contexte, et *arme à feu* *armeafeu*. Les accents seront également supprimés, etc. Pour notre problème, analyser une phrase signifiera associer une information à chaque mot.

À la fin du projet, vous aurez alors un prédicat principal **analyse**(P, A) qui échouera si la phrase P n'est pas valide, et qui mettera dans A l'analyse de P dans le cas contraire.

En Traitement Automatique du Langage Naturel, il est habituel de distinguer plusieurs niveaux d'analyse (morphologique, syntaxique, sémantique, etc.). Dans la suite, il s'agira d'effectuer un traitement morpho-syntaxique des phrases présentées en entrée.¹

Remarque : Le but du projet n'est pas que vous réussissiez à répondre à toutes les questions ; il s'agit de vérifier que les principes du fonctionnement de Prolog sont assimilés.

2 Pré-traitement

Tout d'abord, on va se munir de connaissances linguistiques sur les mots qu'on manipule. Ainsi, on va rentrer dans la base de connaissance Prolog le fait que 'chien' est un nom, 'gentil' est un adjectif, etc. On ne mettra pas d'adjectifs ou de noms au pluriel, car il s'agira de traiter ces cas automatiquement.

Aidez-vous du lexique en annexe pour constituer votre propre lexique (il ne doit pas être forcément très gros, juste assez pour rendre compte des phénomènes que l'on va modéliser ; vous pouvez le compléter au fur et à mesure des besoins du projet).

Ensuite, pour que le traitement soit plus simple durant la phase d'analyse, on va considérer qu'une phrase est une liste de (mots + information), l'information étant une liste d'informations que l'on a sur un mot.

Un prédicat que vous écrirez devra ainsi transformer la phrase donnée en exemple en : [['jean', []], ['parle', []], ['avec', []], ['marie', []]].

Chaque mot est représenté par une liste à 2 éléments : la forme de départ, et une liste d'information sur lui. Au départ, bien entendu, cette liste est vide car rien n'a encore été analysé.

3 Se servir de la base

La première étape va être de se servir des connaissances fournies pour analyser les mots de la phrase. Ainsi, si 'parle' est un verbe (on le sait car il y a **verbe**('parle') dans notre fichier Prolog), on va rajouter 'V' dans la liste d'information sur 'parle'.

Un prédicat que vous écrirez devra ainsi transformer la phrase donnée en exemple en : [['jean', ['NP']], ['parle', ['V']], ['avec', ['Adv']], ['marie', ['NP']]].²

1. Gardez à l'esprit que tout ceci est une simplification de la bien plus complexe réalité.

2. Avec 'NP' pour noms propres, 'V' pour verbe, 'Adv' pour adverbe, etc.

Dans le cas où le mot analysé n'est pas dans la base, on ne peut pour l'instant pas ajouter d'information à son sujet, sa liste associée restera vide. Ainsi, `[['les', []], ['chiens', []], [aiment', []], ['les', []], ['pommes', []]]` deviendra `[['les', ['Det']], ['chiens', []], [aiment', []], ['les', ['Det']], ['pommes', []]]` car on n'a pas de noms au pluriel dans notre fichier.

4 Un peu de morpho-syntaxe

Dans cette section, nous allons introduire des règles visant à modifier les mots de façon à rendre compte de la façon dont ils sont formés. Ainsi, on va essayer de remplacer le mot `['portes', []]` par `['portes', ['N', 'Plur']]` si on a trouvé dans le lexique le fait que 'porte' était un nom.

Dans cette section, vous devrez faire deux choses :

- Écrire des prédicats qui permettent de faire cette transformation (`['portes', []] → ['portes', ['N', 'Plur']]`) pour des règles simples (s au pluriel pour les noms et les adjectifs, conjugaison simple (`['parlons', []] → ['parlons', ['V', '1PP']]`), etc.

Pour cela, on utilisera le prédicat prédéfini `atom_concat(Atom1, Atom2, Atom3)` qui concatène les atomes `Atom1`, `Atom2` et met le résultat dans `Atom3`. Ce prédicat est réversible, ainsi le but `atom_concat('porte', 's', A)` donnera `A = 'portes'` et le but `atom_concat(A, 's', 'portes')` donnera `A = 'porte'`.

- Pour les mots qui n'ont pas réussi à être reconnus dans la section 3, essayez d'appliquer les règles que vous venez d'écrire pour ajouter de l'information sur les mots. Ici, en supposant que les entrées 'chien', 'aime' et 'pomme' soient dans la base, la phrase `[['les', []], ['chiens', []], [aiment', []], ['les', []], ['pommes', []]]` deviendra `[['les', ['Det']], ['chiens', ['N', 'Plur']], [aiment', ['V', 'Plur']], ['les', ['Det']], ['pommes', ['N', 'Plur']]]`.

5 Syntaxe

À ce niveau, tous les mots devraient avoir une information associée. Rajoutez des règles simples pour vérifier que la phrase est syntaxiquement correcte, par exemple qu'une phrase ne peut contenir à la fois deux verbes ou qu'un nom est toujours précédé d'un déterminant. Faites échouer votre prédicat principal `analyse` dans le cas contraire.

6 Le mot de la fin

Bon courage !

7 Annexe

```
%%% - Lexique - %%%  
% determinants  
det('le').  
det('la').  
det('un').  
det('une').  
det('les').  
det('les').  
det('des').  
det('des').  
  
% verbes  
verbe('dormir').  
verbe('bouger').  
verbe('detaler').  
verbe('danser').  
verbe('marcher').  
verbe('manger').  
verbe('attraper').  
verbe('croire').  
verbe('penser').  
verbe('donner').  
verbe('parler').  
  
% Nom propres: Paul, Pierre, Marie, Anne  
nompropre('Paul').  
nompropre('Pierre').  
nompropre('Marie').  
nompropre('Anne').  
  
% noms communs  
nom('chat').  
nom('souris').  
nom('voisine').  
nom('enfant').  
nom('fille').  
nom('mere').  
nom('pere').  
nom('fils').  
nom('maitresse').  
nom('glace').  
nom('bonbon').
```

```

nom('punition').
nom('pomme').
nom('fraise').
nom('vanille').
nom('fromage').
nom('maternelle').
nom('piscine').
nom('plage').

% adjectifs
adj('noir').
adj('blanc').
adj('roux').
adj('sage').
adj('bavard').

% prepositions : à, de
prep('a').
prep('de').

% conjonctions de coordination : et, ou
ccconj('et').
ccconj('ou').

% conjonctions de subordination : que
sconj('que').

%% exemples de règle pour associer une information à un mot connu du lexique
trans([M, I1], [M, ['N'|I1]]) :-
    nom(M).

trans([M, I1], [M, ['V'|I1]]) :-
    verbe(M).

```