École Polytechnique de Montréal

Département de génie informatique et génie logiciel

INF8215

Intelligence artificielle : méthodes et algorithmes

Travail pratique # 2 : Programmation logique

Soumis par

Rafael Blanchet-Lecomte, # 1678307

Vincent Audet, # 1637998

Montréal

Le samedi 25 mars 2017

# Projet # 1

## *Personnes*

TODO

## *Objets*

TODO.

# Projet # 2

## *Valider une séquence*

La première fonction sert à valider une séquence en fonction d’une liste de contraintes (soit une liste de contraintes de colonnes ou de lignes). Ainsi, pour ce faire, on commence par écarter toutes les valeurs nulles (case blanche) de la séquence en appelant récursivement la fonction *valid\_seq* sur la queue d’une séquence débutant par un 0.

Par la suite, on regarde la tête de la liste de contraintes (la liste de séquence a maintenant pour tête une valeur de 1 puisque nous avons écarté tous les 0) et si cette valeur est positive, on la décrémente et on appelle récursivement la fonction de validation avec la valeur décrémentée et la queue de la séquence. Cela fait en sorte que, en diminuant la taille du bloc de 1 et en écartant la tête de la séquence, on vient de confirmer que la première case de ce bloc est belle et bien noire.

Finalement, *valid\_seq* s’assure également que les listes de contraintes vides ou ne possédant qu’un 0, ou encore que les listes de séquences vides soient correctement traitées (voire code).

## *Valider les lignes*

La deuxième fonction a pour but de s’assurer que les lignes sont valides (en utilisant la fonction de validation de séquence décrite ci-haut). Ainsi, pour ce faire, nous avons tout d’abord inspecté la première ligne (à l’indice 1) par rapport à la tête de la liste de contraintes.

Ensuite, en incrémentant l’index en cours et en écartant la tête de la liste des contraintes, nous avons été en mesure de parcourir l’ensemble des lignes.

Finalement, pour la validation en soit, nous avons fait appelle à une fonction d’extraction (voir plus bas) pour y extraire chaque ligne pour être en mesure de la passer à notre fonction de validation (décrite ci-haut) et ainsi s’assurer qu’elle est effectivement valide.

## *Valider les colonnes*

La troisième fonction a exactement le même but que la précédente mais pour les colonnes. Ainsi, on procède exactement de la même façon pour parcourir l’ensemble des colonnes et des contraintes et y extraite chaque colonne individuellement pour être en mesure de les valider une à une.

## *Extraire*

Finalement, la quatrième et dernière fonction est une fonction d’extraction. C’est d’ailleurs cette fonction qui permet d’aller chercher une ligne (ou une colonne) individuellement pour pouvoir la faire valider par la fonction *valid\_seq*. Pour ce faire, on utilise la fonction *nth1* de PROLOG qui retourne vrai si le résultat (troisième argument) est bel et bien à l’index (premier argument) de la liste (deuxième argument) spécifié lors de l’appel. Lors de cet appel, on utilise un coupe-choix pour ne conserver que la solution en cours.

Par la suite, on parcourt récursivement l’ensemble de la liste de données, toujours en séparant la tête de la queue et en y confirmant à chaque fois la présence des données à l’index spécifié à l’aide de la fonction *nth1*.

# Bibliographie

* <https://gist.github.com/noniq/4147547>

Il est à noter que nous avons fortement utilisé le « repo GIT » ci-haut pour comprendre la nature du travail et la façon de le réaliser. Notre code est donc relativement similaire à celui retrouvé dans le lien ci-haut puisqu’une fois compris, il était très difficile d’approcher le problème de manière différente. Cependant, l’ampleur des commentaires à la fois dans le code et dans ce rapport vous démontreront bien notre compréhension du travail pratique.