TD 10 - NLTK, ambiguïtés de portée

Pour traiter de l'ambiguïté de portée des quantificateurs on cherche à implémenter la méthode du *Cooper Storage*.

L'idée de la méthode est de remplacer les expressions quantifiées par autant de variables au moment de la composition. Les contributions réelles des expressions sont stockées au sein d'une liste. Lorsque la composition est terminée, on retire une à une les expressions de cette liste et on les applique successivement à la formule obtenue compositionnellement.

Selon l'ordre dans lequel les expressions sont récupérées, différentes portées vont être générées. Ce n'est donc plus une ambiguïté de type qui va décider de la portée des expressions quantifiées.

Exemple

L'exemple (1a) possède les deux lectures indiquées en (1b) et (1c).

(1) a. Tout oiseau aime une libellule.

b. $\forall x.(bird(x) \rightarrow \exists y.(dragonfly(y) \wedge love(x, y)))$

c. $\exists y.(dragonfly(y) \wedge \forall x.(bird(x) \rightarrow love(x, y)))$

Pour les obtenir on considère que le "cœur" de la formule est fourni par le verbe, i.e. $love(x, y)$. Chacune des expressions référentielles peut être envisagée comme une formule "à trou", i.e. :

– Tout oiseau : $\forall x.(bird(x) \rightarrow \psi)$

– une libellule : $\exists y.(dragonfly(y) \wedge \phi)$

Pour construire la formule finale on sait que la contribution du verbe doit faire partie de chacune des formules associées aux expressions quantifiées (c.à.d. $love(x, y)$ doit être un sous constituant des représentations de *tout oiseau* et *une libellule*). Mais la relation entre les expressions quantifiées est sous-spécifiée : on ne sait pas laquelle fait partie de l'autre (c.à.d. que la représentation de *tout oiseau* peut être un sous constituant de *une libellule* ou vice-versa). On a donc deux possibilités pour reconstruire la formule finale :

– on identifie ψ à $love(x, y)$, et donc ϕ à la contribution de *tout oiseau* : l'existential a portée large

– on identifie ϕ à $love(x, y)$, et donc ψ à la contribution de *une libellule* : l'universel a portée large

Implémentation

Pour implémenter le Cooper Storage :

- Étudiez la grammaire `gram0-store.fcfg` pour comprendre le principe du Cooper Storage appliqué aux noms propres (pour vérifier vos résultats, utilisez le script `scriptCS-155.py`)
- Étendez la grammaire pour qu'elle traite les *NP* quantifiés :
 - Limitez vous d'abord au cas des verbes intransitifs (donc sans ambiguïté de portée)
 - Introduisez ensuite les verbes transitifs et vérifiez que plusieurs lectures sont bien générées. Des ambiguïtés sont-elles générées dans le cas de *NP* non quantifiés ?
 - Introduisez les verbes ditransitifs et observez vos résultats ; sont-ils satisfaisants ?
- Étendez la grammaires à tous les phénomènes mentionnés dans le TP précédent.
- Vous pouvez modifier le script `python` pour adapter la sortie à votre convenance