Optimisation d'analyse syntaxique par réécriture de programme

Définition (Grammaire)

Ensemble des règles qui définissent un langage donné.

Définition (Analyse syntaxique)

- Détermine l'appartenance d'un mot à un langage engendré par une grammaire ;
- Construit l'arbre de dérivation (suite des dérivations qui a permis d'engendrer le mot).

Exemple (Grammaire)

Soit $\mathcal{G}$ la grammaire suivante :

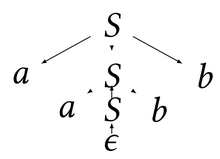
$$S \rightarrow aSb \quad (1)$$
$$S \rightarrow \epsilon \quad (2)$$

qui génère le langage $\{a^n b^n \mid n \geq 0\}$.

Exemple (Analyse syntaxique)

$aabb$ appartient au langage généré par $\mathcal{G}$.

L'arbre de dérivation ci-dessous en est une preuve.



Grammaire catégorielle abstraite [de Groote, 2001] et Acgtk [Pogodalla, 2016]

- Grammaire catégorielle abstraite (ACG) :
  - Cadre qui permet d'encoder des formalismes grammaticaux (comme les grammaires hors-contexte et les grammaires d'arbres adjoints).
- ACGtk :
  - Boîte à outils pour travailler sur les ACG ;
  - Analyseur syntaxique intégré.
- Objectif : Optimiser l'analyse syntaxique dans Acgtk.

Datalog [Ullman, 1983]

- Datalog :
  - Langage de requêtes et de règles très proche de Prolog ;
  - Utilisé pour interroger des bases de données déductives.
- Base de données déductives :
  - Système déterminant des faits nouveaux à partir de faits initiaux et de règles.

Équivalence

L'analyse syntaxique avec les ACG est équivalente à la recherche de démonstrations pour une requête dans un programme Datalog.

Résultats

- La réécriture implémentée est efficace ;
- Représentation graphique des temps d'évaluation pour un programme réécrit (en vert) et le programme original (en bleu) ;
- Résultats pour deux types de grammaires différents :
  - Grammaire hors-contexte $\mathcal{G}$ qui génère le langage $\{a^n b^n \mid n \geq 0\}$;
  - Grammaire légèrement sensible au contexte $\mathcal{G}'$ qui génère le langage $\{a^n b^n c^n d^n \mid n \geq 0\}$.

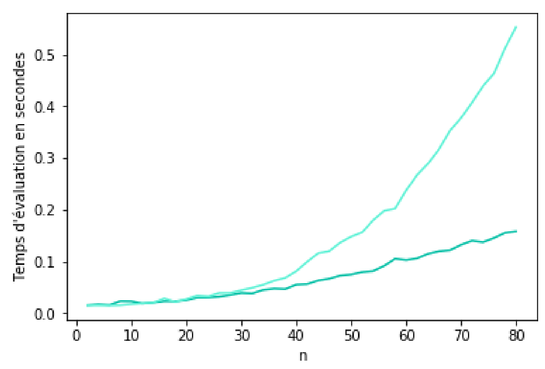


FIGURE – Temps de réponse de l'analyse syntaxique en fonction de n pour la grammaire $\mathcal{G}$

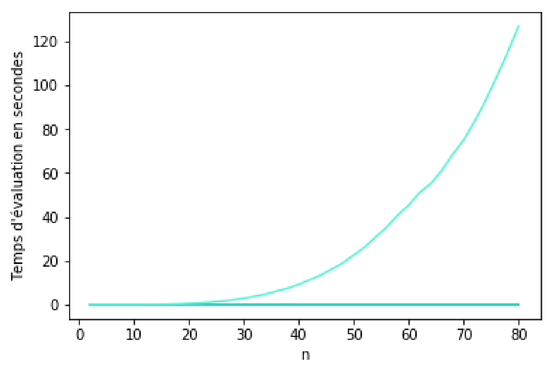
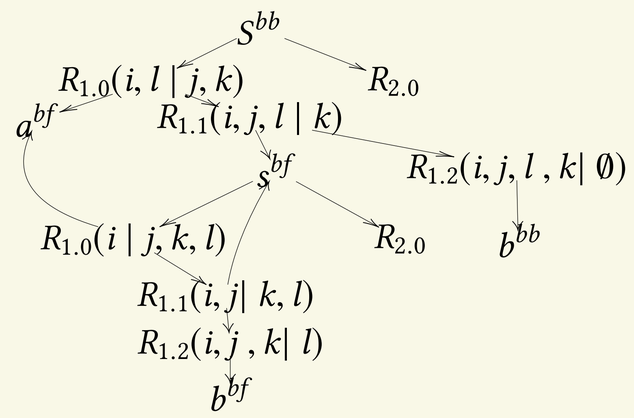


FIGURE – Temps de réponse de l'analyse syntaxique en fonction de n pour la grammaire $\mathcal{G}'$

Réécriture Magic Set Beeri and Ramakrishnan [1991]

- Optimise l'évaluation d'un programme Datalog en le transformant ;
- Nécessite du pré-traitement :
  - Création du Rule / Goal Graph ;
  - Séparation des prédicats.
- Étapes de réécriture :
  - Création des règles magiques (***I***) ;
  - Génération des règles supplémentaires de rang 0 (***II***) ;
  - Génération des règles supplémentaires de rang k (***III***) ;
  - Transformation des règles initiales (***IV***) ;

Rule/Goal Graph



Programme Datalog réécrit générant $a^n b^n$

$$S\_bb(i, l) \leftarrow sup_{1.2}(i, l, k), b(k, l) \quad (\textbf{\textit{IV}})$$
$$sup_{1.0}(i, l) \leftarrow magic\_S\_bb(i, l) \quad (\textbf{\textit{II}})$$
$$sup_{1.1}(i, l, j) \leftarrow sup_{1.0}(i, l), a(i, j) \quad (\textbf{\textit{III}})$$
$$sup_{1.2}(i, l, k) \leftarrow sup_{1.1}(i, l, j), S\_bf(j, k) \quad (\textbf{\textit{III}})$$

$$magic\_S\_bf(j, k) \leftarrow sup_{1.1}(i, l, j) \quad (\textbf{\textit{I}})$$

$$S\_bb(i, l) \leftarrow sup_{1.2}(i, l, k), b(k, l) \quad (\textbf{\textit{IV}})$$
$$sup_{2.0}(i, l) \leftarrow magic\_S\_bf(i, l) \quad (\textbf{\textit{II}})$$
$$sup_{2.1}(i, l, j) \leftarrow sup_{2.0}(i, l), a(i, j) \quad (\textbf{\textit{III}})$$
$$sup_{2.2}(i, l, k) \leftarrow sup_{2.1}(i, l, j), \_bf(j, k) \quad (\textbf{\textit{III}})$$

$$magic\_S\_bf(j, k) \leftarrow sup_{2.1}(i, l, j) \quad (\textbf{\textit{I}})$$

$$S\_bb(i, i) \leftarrow sup_{3.0}(i, i) \quad (\textbf{\textit{IV}})$$
$$sup_{3.0}(i, i) \leftarrow magic\_S\_bb(i, i) \quad (\textbf{\textit{II}})$$

$$S\_bf(i, i) \leftarrow sup_{4.0}(i, i) \quad (\textbf{\textit{IV}})$$
$$sup_{4.0}(i) \leftarrow magic\_S\_bf(i, i) \quad (\textbf{\textit{II}})$$

Références

C. Beeri and R. Ramakrishnan. On the power of magic. *The Journal of Logic Programming*, 10(3) :255 – 299, 1991. doi : 10.1016/0743-1066(91)90038-Q. Special Issue : Database Logic Progamming.

P. de Groote. Towards Abstract Categorial Grammars. In *Proceedings of 39th Annual Meeting of the Association for Computational Linguistics*, pages 148–155, 2001. URL `http://aclweb.org/anthology/P01-1033`.

M. Kanazawa. Parsing and generation as datalog queries. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL 2007)*, pages 176–183, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P07-1023`.

S. Pogodalla. ACGTK : un outil de développement et de test pour les grammaires catégorielles abstraites. In L. Danlos and T. Hamon, editors, *Actes de la 23ème Conférence sur le Traitement Automatique des Langues Naturelles, 31ème Journées d'Études sur la Parole, 18ème Rencontre des Étudiants Chercheurs en Informatique pour le Traitement Automatique des Langues (JEP-TALN-RECITAL 2016)*. Association pour le Traitement Automatique des Langues, Association Francophone pour la Communication Parlée, 7 2016. URL `https://hal.inria.fr/hal-01328702`. Démonstration.

J. D. Ullman. *Principles of Database and Knowledge-Base Systems : Volume I*. W. H. Freeman & Co., New York, NY, USA, 2nd edition, 1983. ISBN 0716780690.

Maxime GUILLAUME et Guilherme RAZET

Institut des sciences du Digital - Management et Cognition