Énumération de mots à délai constant

Antoine Amarilli

L'informatique théorique s'intéresse principalement à la complexité computationnelle de problèmes dits "de décision" où, étant donné une entrée, on souhaite déterminer si elle appartient ou non à une certaine classe. Cependant, ce formalisme convient mal pour des problèmes où l'on souhaite calculer une sortie potentiellement de grande taille, par exemple les résultats d'une requête sur une base de données, ou les facteurs d'un mot qui satisfont une certaine expression régulière. En effet, dans de tels cas, écrire toute la sortie prend déjà un temps trop important. Il faut donc une mesure de complexité plus fine qui permette de comptabiliser, par exemple, le temps nécessaire pour produire les quelques premiers résultats.

Le formalisme des *algorithmes d'énumération* permet de préciser cette intuition. Un algorithme d'énumération pour un problème se compose de deux phases. Dans une première phase dite de *précalcul*, l'algorithme prépare une structure de données à partir de l'entrée. Ensuite, dans une seconde phase dite d'*énumération*, l'algorithme écrit le flux des solutions du problème, l'une après l'autre. On mesure la complexité de la phase de précalcul en fonction de l'entrée, et on mesure celle de la phase d'énumération comme le *délai* maximal entre deux solutions successives. Dans ce contexte, la meilleure complexité possible est d'avoir un prétraitement *linéaire* en l'entrée, puis de produire des solutions à délai *constant*. Cette complexité est atteinte pour l'évaluation de requêtes dites *acyclic free-connex* [1] ou pour l'énumération de résultats de *spanners* c'est-à-dire d'automates avec captures [2]. Dans ces deux cas, bien sûr, les solutions énumérables *à délai constant* doivent être de taille constante. Si on souhaite pouvoir produire des solutions de taille non-bornée, par exemple pour des requêtes avec variables libres du second ordre, il faut accepter un délai linéaire en chaque solution.

L'objet de ce stage est de repousser les limites de ce qui est énumérable à délai constant, en explorant un modèle de calcul différent. Au lieu d'écrire chaque solution en entier, on s'autorise à le faire en modifiant une solution précédente. Par exemple, on dispose d'une mémoire tampon où l'on peut écrire des solutions, et d'une instruction `output` qui "produit" le contenu du tampon en temps constant indépendamment de sa taille. On souhaite déterminer quels résultats d'énumération sont à délai réellement constant dans ce modèle. Le but du stage est de prouver des premiers résultats sur cette question, notamment des algorithmes à délai constant et des bornes inférieures.

On s'intéressera en particulier au problème suivant : pour quels langages réguliers peut-on énumérer en délai constant les mots du langage ? Pour le langage $\Sigma = \{0, 1\}^*$, l'énumération des mots binaires à délai constant est faisable avec un code de Gray [3]. Pour le langage $\Sigma = 0^* + 1^*$, cela semble manifestement impossible avec un seul buffer, mais faisable si on dispose de deux. Quels langages peuvent être ainsi énumérés avec un ou plusieurs buffers ?

Supervision et environnement

Le stage se déroulera dans l'équipe DIG de Télécom Paris (Palaiseau, France). Il sera encadré par Antoine Amarilli (https://a3nm.net/), maître de conférences dans l'équipe.

Références

- [1]: Luc Segoufin. *A glimpse on constant-delay enumeration* (invited talk), STACS 2014 (en ligne : https://drops.dagstuhl.de/opus/volltexte/2014/4500/).
- [2]: Antoine Amarilli, Pierre Bourhis, Stefan Mengel, Matthias Niewerth. *Constant-delay enumeration for nondeterministic document spanners*, TODS, 2021 (en ligne : https://arxiv.org/abs/2003.02576).
- [3]: Article Wikipédia "Gray code" (https://en.wikipedia.org/wiki/Gray_code)