Slider : un Raisonneur Incrémental Évolutif

Jules Chevalier*, Julien Subercaze*
Christophe Gravier* Frédérique Laforest*

*Université de Lyon, F-42023, Saint-Étienne, France,
CNRS, UMR5516, Laboratoire Hubert Curien, F-42000, Saint-Étienne, France,
Université de Saint-Étienne, Jean Monnet, F-42000, Saint-Étienne, France.
prenom.nom@univ-st-etienne.fr, http://laboratoirehubertcurien.fr

Les solutions existantes pour le raisonnement incrémental souffrent principalement de leur incapacité à prendre en charge des ontologies complexes et ne sont pas conçues pour gérer de grandes quantités de connaissances. Dans cet article, nous présentons Slider (Chevalier et al. (2015)), un raisonneur incrémental évolutif par chaînage avant, permettant de raisonner sur des flux de données sémantiques.

Les principales caractéristiques de Slider sont les suivantes :
Exécution parallèle et passage à l'échelle. Le processus est parallélisé plutôt que distribué, malgré le coût de l'accès concurrent au triplestore.
Limitation des doublons. Les connaissances reçues ou inférées par le raisonneur sont accessibles de manière concurrente par tous les modules du système.
Support de flux de données. L'achitecture parallèle de Slider lui permet de recevoir des données de sources statiques et dynamiques.
Indépendance au fragment. Slider supporte nativement les fragments RDFS et ρdf et peut être utilisé pour des fragments plus complexes.

Notre système multi-processus est constitué de modules autonomes, chacun associé à une règle d'inférence, comme le présente la figure 1. Un *triplestore* unique est partagé par ces modules de manière synchronisée. Le partitionnement vertical, introduit par Abadi et al. (2007), permet un accès rapide aux données.

Les triples envoyés au système sont récupérés par le *distributeur général*. Cet élément reçoit les nouveaux triples, les stocke dans le triplestore, et les envoie aux règles pouvant les utiliser. Chaque règle associée à un *buffer* récupère les triples envoyés par le *distributeur général*. Lorsque qu'un *buffer* est plein, ou n'a pas reçu de nouveau triples depuis un temps prédéfini (timeout), un *exécuteur de règle* est instancié afin d'appliquer la règle d'inférence sur les triples présents dans le *buffer*. Ces *buffers* permettent d'améliorer la répartition de charge en appliquant les règles sur des ensembles de triples et non pas pour chaque triple, réduisant ainsi la quantité de processus légers instanciés. Ils assurent également que chaque triple est traité par le système en garantissant, même lors de l'exécution d'une règle, qu'ils sont recueillis par le *buffer*. Les triples inférés par les *exécuteurs de règle* sont ensuite récoltés par le *distributeur* de la règle correspondante. Ce *distributeur* les redistribue aux *buffers* des règles pouvant utiliser ces triples. Afin de déterminer si une règle peut utiliser un triple inféré par une autre