Représentation des connaissances
2020/2021

TD n°2 - Partie 1

Documentation de Protégé : http ://protegeproject.github.io/protege/getting-started/

Documentation SWRL de Protégé : https ://github.com/protegeproject/swrlapi/wiki

SWRL : A Semantic Web Rule Language Combining OWL and RuleML :
https ://www.w3.org/Submission/SWRL/

Nous attendons que vous rendiez au plus tard le 15 janvier à 23h59 (heure de Paris) dans le devoir "Rendu du TD du 11 janvier 2021 - partie 1" le fichier OWL correspondant aux questions.

Téléchargez le fichier td2.owl avec Protégé et vérifiez dans la configuration des raisonneurs que les inférences sur les propriétés et les individus sont bien sélectionnées. Choisissez aussi le raisonneur Pellet.

Pour pouvoir utiliser des expressions de classes OWL dans une règle, on ne peut pas utiliser l'onglet SWRLTab. Il faut ajouter la vue Rule (Window<Ontology Views>Rules) dans un autre onglet. Dans cet onglet, le « et » n'est plus « ^ » mais « , ».

Introduction et écriture de la première règle

Classes prédéfinies : `ProcessDomain` (modélisation des processus, indépendamment de leur type) et `AgileDomain` (modélisation des méthodes agiles), `SequentialProcess` (processus séquentiels, les sous-processus s'exécutent les uns après les autres), `ParallelProcess` (tous les sous-processus s'exécutent en parallèle).

Les itérations sont toutes d'une semaine.

L'individu `Iteration1` est un processus séquentiel dont les sous-processus sont les suivants (dans l'ordre) : `InstallDevelopmentEnvironment -> BuildStoryLogin -> BuildStoryCreateCase`.

Remarquer que `hasSubsequentProcess` est transitive et que `hasNextProcess` ne l'est pas bien que `hasNextProcess` soit sous-propriété de `hasSubsequentProcess` (patron courant).

Pour calculer la longueur d'un processus séquentiel, on a besoin de deux règles une pour le cas initial et une autre pour les autres cas.

La propriété `durationFromStart`, pour chaque processus d'une séquence, représente la durée depuis que le processus initial de la séquence a commencé.

Nous allons commencer par expliciter la valeur de `durationFromStart` du processus initial d'une séquence qui est égal à la durée du processus (`durationInWorkHours`). La règle est la suivante :

```
SequentialProcess(?sp) ^ hasStartProcess(?sp, ?stp)
       ^ durationInWorkHours(?stp, ?spd)
       -> durationFromStart(?stp, ?spd)
```

Rentrer la règle, faire tourner le raisonneur et regarder l'individu `InstallDevelopmentEnvironment`. Sa valeur pour la propriété `durationFromStart` est maintenant de 8.0, identique à celle de `durationInWorkHours`. Regarder les explications.

Travail à faire

Pour les questions qui suivent, il faudra choisir entre une règle SWRL ou un axiome.

1. Spécifier `durationFromStart` pour les sous-processus non initiaux d'un processus séquentiel
2. Spécifier `durationFromStart` pour les processus parallèles (`ParallelProcess`).
3. Spécifier la marge de manoeuvre (`slack`) pour les instances de `Iteration`.
4. Créer une classe nommée `BehindScheduleProcess` dont les instances seront les instances de `process` ayant une marge de manoeuvre négative (`slack`). Spécifier les instances de cette classe.

   Indice : `slack some xsd:decimal[< 0]`

   reprendre celui-là pour regarder ce que donne slack < 0
5. Définir une classe `ProjectManager` et une ObjectProperty `Manage` de domaine `ProjectManager` et de co-domaine `Process` telle que si ?x manage ?y alors ?x manage tous les sous-process de ?y.
6. Définir une classe `LoopProcess` dont les instances seront des processus séquentiels contenant une boucle dans leur succession de sous-processus.