Intervalles d'Allen

Nous allons implémenter dans ce TME le calcul des relations obtenues par composition de relations, ainsi que la création d'un graphe temporel grâce à l'algorithme de propagation d'Allen.

Il est conseillé de faire ce TME en Python, mais il est possible d'utiliser d'autres langages. En Python, voici quelques (rapides) rappels qui peuvent être utiles :

`import definitions as d` permet d'importer le fichier defitions.py s'il est dans le même dossier. Il est ensuite possible d'utiliser le dictionnaire `transpose` (par exemple) avec `d.transpose`

`['a','b','c']` est une **liste**

`list.pop()` enlève de la liste le dernier élément et le retourne

`dic={'k1':'a','k2':['z','e'],('k3','k4'):5}` est un **dictionnaire**, indexé par des clés (un tuple peut servir de clé). `dic['k1']` renvoie `'a'`

`dict.keys()` retourne les différentes clés du dictionnaire

`s=set(['a','z','a'])` **ou** `s={'a','z','a'}` crée un **ensemble ("set")** qui contient `{'a','z'}` (les doublons n'apparaissent qu'une fois). La structure fonctionne à peu près de la même manière que les listes, mais ne contient qu'un unique exemplaire de chaque objet. Voir la doc Python sur la structure : on peut en particulier calculer l'union, l'intersection, la différence... entre deux sets.

`set()` crée une collection vide de type "set" ({} crée un dictionnaire vide)

Les fichiers Python, **commentés**, sont à envoyer pour le compte rendu.

Exercice 1 Implémentation de la composition

1. Récupérer sur moodle le fichier `definition.py` qui définit :
   - un dictionnaire `transpose` : `transpose[r]` fournit la transposition de la relation `r`.
   - un dictionnaire `symetrie` : `symetrie[r]` fournit la symetrie de la relation `r`.
   - un dictionnaire `compositionBase` dont les clés sont des couples de relations : `compositionBase[(r1,r2)]` est le set des relations obtenues lorsqu'on compose la relation `r1` avec la relation `r2`. Ce dictionnaire représente la table de composition donnée en cours et rappelée dans l'annexe du TD.
2. Définir les fonctions `transposeSet` et `symetrieSet` telles que `transposeSet(S)` (resp. `symetrieSet(S)`) retourne l'ensemble des relations transposées (resp. symétriques) des relations qui apparaissent dans `S`.
3. Définir la fonction `compose` telle que `compose(r1, r2)` retourne l'ensemble des relations que l'on peut obtenir par composition des relations `r1` et `r2`.

   Il s'agit donc d'utiliser :
   - la table de composition directement
   - la transposition : $r_1 \circ r_2 = (r_2^t \circ r_1^t)^t$
   - la symétrie : $r_1 \circ r_2 = (r_1^s \circ r_2^s)^s$
   - les deux transformations simultanément : $r_1 \circ r_2 = (r_2^{st} \circ r_1^{st})^{ts}$

   Remarque : il peut être utile de tester si un élément fait partie des clés d'un dictionnaire avec `(k1,k2) in dict.keys()`.

   Pour tester l'implémentation, vérifier par exemple que
   - $= \circ \mathrm{d} = \{\mathrm{d}\}$
   - $\mathrm{m} \circ \mathrm{d} = \{\mathrm{d}, \mathrm{o}, \mathrm{s}\}$
   - $\mathrm{ot} \circ > = \{>\}$

- $> \circ e = \{>\}$
- $ot \circ m = \{dt,et,o\}$

4. Définir la fonction `compositionSet` telle que `compositionSet(S1, S2)` retourne l'ensemble des relations que l'on peut obtenir par composition des relations qui apparaissent dans l'ensemble `S1` avec les relations qui apparaissent dans l'ensemble `S2`.

   Il s'agit de généraliser la fonction précédente à des ensembles.

Exercice 2 Gestion du graphe temporel

Nous allons ici implémenter l'algorithme d'Allen. L'implémentation de cette partie peut ne pas suivre exactement les indications si une autre solution vous parait plus facile. Il est toutefois conseillé de représenter un graphe par :

- l'ensemble de ses noeuds
- l'ensemble des relations entre ses noeuds, par exemple à l'aide d'un dictionnaire ayant pour clés des couples de noeuds, et pour valeurs les ensembles des relations correspondantes.

Un graphe avec deux noeuds A et B et A{o,e}B sera donc représenté par :

- ses noeuds {'A','B'}
- ses relations {('A','B') :{'o','e'} }

1. Créer une classe `Graphe` ayant pour attributs `noeuds` et `relations`, ainsi qu'une fonction `getRelations`. `graphe.getRelations(i,j)` doit retourner l'ensemble des relations entre les noeuds `i` et `j` (attention si c'est le cas transposé qui est stocké dans `graphe.relations`, ou si il n'existe pas de relations stockées entre les noeuds `i` et `j` !)
2. Créer une fonction `propagation` qui prend en entrée un graphe et deux noeuds de ce graphe, et utilise l'algorithme d'Allen pour propager la relation entre ces deux noeuds dans le reste du graphe (algorithme donné en TD). On suppose ici que les deux noeuds existent déjà, on ne fait que propager la relation en question.
3. Créer une fonction `ajouter` qui prend en entrée un graphe et une relation entre deux noeuds, et ajoute cette relation au graphe. On peut avoir les deux noeuds qui appartiennent déjà au graphe, l'un d'eux uniquement, ou aucun.

   Pour construire un graphe cohérent, il suffit maintenant de créer un graphe vide puis d'ajouter les relations une par une.
4. Tester la création de graphe avec des cas simples de graphes à trois noeuds A, B, C :
   - un graphe avec les relations A{<}B et A{>}C
   - un graphe avec les relations A{<}B et A{<}C
   - dans les deux cas, propager la relation B{=}C
5. On reprend l'exemple du cours :

   > La lampe s'est allumée pendant ou juste après que j'ai appuyé sur l'interrupteur. John n'était pas dans la pièce lorsque j'ai appuyé sur l'interrupteur.

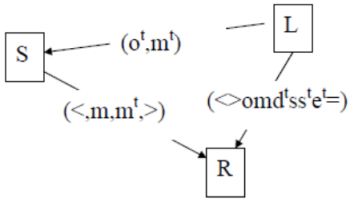
   On note :

   S l'instant où je touche l'interrupteur

   L le temps durant lequel la lampe est allumée

   R le temps durant lequel John est dans la pièce

   Créer le graphe correspondant, uniquement avec les relations données dans l'énoncé (entre L et S, et entre R et S). La relation entre L et R doit être créée automatiquement. Le graphe après propagation doit être :



6. On apprend que John était dans la pièce lorsque la lumière a été éteinte. Propager cette nouvelle contrainte dans le graphe. Quel est le résultat ?
7. Résoudre l'exercice 2 du TD, sur le petit-déjeuner d'Alfred.