TD 3 – Formalismes de représentation et raisonnement

1 Logiques de descriptions

Représentez les concepts et individus suivants sous la forme d'assertions en logique de descriptions :

- Dans le monde de Mario Kart, il y a des véhicules, qui peuvent être des motos ou des voitures.
- Les véhicules sont conduits par des pilotes. Un pilote qui conduit une voiture est un pilote de voiture. Un pilote qui conduit une moto est un pilote de moto.
- Les motos ont deux roues, les voitures en ont 4.
- Les motos ont une forte accélération.
- Il y a des véhicules rapides, qui ont une vitesse haute, et des véhicules lents qui ont une vitesse basse.
- Les véhicules rapides ont nécessairement une faible accélérations (mais les véhicule lents n'ont pas nécessairement une forte accélération).
- Un véhicule qui n'est pas rapide est un véhicule lent.
- On ne peut pas avoir à la fois une forte accélération et une faible accélération.

Montrez intuitivement que le concept de moto rapide est insatisfiable et du coup, que moto est un sous-concept de véhicule lent.

2 Programmation logique inductive

Nous allons utiliser Aleph pour trouver des règles qui s'appliquent aux véhicules de Mario Kart, d'après le jeu de données à `https://mdaquin.github.io/t/FRR_TD3_2022/mkbodies.csv` qui contient des valeurs (high, medium, low) pour la vitesse (speed), l'accélération (acceleration), le poids (weight) et la manoeuvrabilité (handling) des véhicules.

2.1 Installation et test d'Aleph

Installez SWI-Prolog sur votre machine. Vous devriez pouvoir le lancer en ligne de commande avec la commande `prolog`, `swprolog`, `swiprolog` ou `swipl`. Assurer vous qu'il s'agit bien d'une version récente (idéalement à 8.4.x).

Téléchargez ensuite le fichier `aleph.pl` (qui contient le module pour le système de programmation logique inductive Aleph) à partir de `https://mdaquin.github.io/t/FRR_TD3_2022/aleph.pl`.

Téléchargez ensuite les fichiers de connaissances, exemples positifs et exemples négatifs vus en cours à partir de :

- Connaissances et objectifs : fam1.b `https://mdaquin.github.io/t/FRR_TD3_2022/fam1.b`
- Exemples positifs : fam1.f `https://mdaquin.github.io/t/FRR_TD3_2022/fam1.f`
- Exemples négatifs : fam1.n `https://mdaquin.github.io/t/FRR_TD3_2022/fam1.n`

Placez les fichiers `aleph.pl`, `fam1.b`, `fam1.f` et `fam1.f` dans un même répertoire et lancez swi-prolog à partir de ce répertoire, puis :

- tapez `['aleph.pl'].` et entrer pour charger Aleph.
- tapez `read_all(fam1).` et entrer (deux fois) pour charger l'exemple.
- tapez `induce.` et entrer pour lancer l'apprentissage.

Inspectez le résultat pour voir si les règles attendues ont bien été trouvées.

Essayez aussi le deuxième exemple avec les fichiers :

- Connaissances et objectif : fam2.b `https://mdaquin.github.io/t/FRR_TD3_2022/fam2.b`
- Exemples positifs : fam2.f `https://mdaquin.github.io/t/FRR_TD3_2022/fam2.f`
- Exemples négatifs : fam2.n `https://mdaquin.github.io/t/FRR_TD3_2022/fam2.n`

En utilisant les observations dans le fichier `https://mdaquin.github.io/t/FRR_TD3_2022/mkbodies.csv`, l'objectif ici est de créer vos propres fichiers .b, .f et .n (un de chaque pour chaque question), de façon à induire des règles permettant de déduire :

1. l'accélération en fonction de la vitesse.
2. le poids en fonction de la manoeuvrabilité ou de l'accélération.
3. la relation "plus rapide que" (faster) entre karts (e.g. un kart avec une vitesse rapide (high) est plus rapide qu'un kart avec une vitesse moyenne (medium) ou lente (low)).
4. La relation "plus manoeuvrable que" (smoother) en fonction de la relation "plus rapide que" (il faut ajouter les règles obtenues à la question précédente).

Le manuel de Aleph se trouve à `http://www.di.ubi.pt/~jpaulo/competence/tutorials/aleph.pdf` pour vous aider.

Vous pouvez générer les exemples positifs et négatifs manuellement, ou automatiquement en utilisant des scripts tels que les suivants en python (requiert python 3, pandas et que le fichiers `mkbodies.csv` se trouve dans le même répertoire):

extract_predicate.py : pour extraire des exemples positifs ou des connaissances pour un prédicat (questions 1 à 4).

```
import pandas as pd
import sys
predicate = sys.argv[1]
df = pd.read_csv("mkbodies.csv")
def tovar(s):
    return s.lower().replace(" ", "").replace(".", "").replace("-", "")
for x in df.iloc:
    print(f"{predicate.lower()}({tovar(x['Vehicle'])}, {x[predicate]}).")
```

Utilisation :

```
$> python extract_predicate.py Speed
speed(standardkart, medium).
speed(pipeframe, low).
speed(mach8, medium).
...
speed(bonerattler, medium).
speed(inkstriker, medium).
speed(splatbuggy, medium).
```

extract_neg_predicate.py : pour extraire des exemples négatifs pour un prédicat (questions 1 et 2).

```
import pandas as pd
import sys
predicate = sys.argv[1]
df = pd.read_csv("mkbodies.csv")
def tovar(s):
    return s.lower().replace(" ", "").replace(".", "").replace("-", "")
for x in df.iloc:
    if x[predicate] == "low":
        print(f"{predicate.lower()}({tovar(x['Vehicle'])}, high).")
    elif x[predicate] == "high":
        print(f"{predicate.lower()}({tovar(x['Vehicle'])}, low).")
```

Utilisation :

```
$> python extract_neg_predicate.py Acceleration
acceleration(pipeframe, low).
acceleration(steeldriver, high).
acceleration(circuitspecial, high).
...
acceleration(citytripper, low).
acceleration(standardatv, high).
acceleration(bonerattler, high).
```

extract_comp.py : pour extraire des prédicats comparant les valeurs de couples de karts (questions 3 et 4).

```
import pandas as pd
import sys
predicate = sys.argv[1]
opred = sys.argv[2]
df = pd.read_csv("mkbodies.csv")
def tovar(s):
    return s.lower().replace(" ", "").replace(".", "").replace("-", "")
for x in df.iloc:
    for y in df.iloc:
        if x[predicate] == "high":
            if y[predicate] == "medium" or y[predicate] == "low":
                print(f"{opred.lower()}({tovar(x['Vehicle'])}, {tovar(y['Vehicle'])}).")
        if x[predicate] == "medium":
            if y[predicate] == "low":
                print(f"{opred.lower()}({tovar(x['Vehicle'])}, {tovar(y['Vehicle'])}).")
```

Utilisation :

```
$> python extract_comp.py Speed faster
faster(standardkart, pipeframe).
faster(standardkart, biddybuggy).
faster(standardkart, landship).
...
faster(splatbuggy, varmint).
faster(splatbuggy, mrscooty).
faster(splatbuggy, citytripper).oo
```

extract_neg_comp.py : pour extraire des prédicats négatifs comparant les valeus des couples de karts (questions 3 et 4).

```
import pandas as pd
import sys
predicate = sys.argv[1]
opred = sys.argv[2]
df = pd.read_csv("mkbodies.csv")
def tovar(s):
    return s.lower().replace(" ", "").replace(".", "").replace("-", "")
for x in df.iloc:
    for y in df.iloc:
        if x[predicate] == "low":
                print(f"{opred.lower()}({tovar(x['Vehicle'])}, {tovar(y['Vehicle'])}).")
        if x[predicate] == "medium":
            if y[predicate] == "high" or y[predicate] == "medium":
                print(f"{opred.lower()}({tovar(x['Vehicle'])}, {tovar(y['Vehicle'])}).")
        if x[predicate] == "high":
            if y[predicate] == "high":
                print(f"{opred.lower()}({tovar(x['Vehicle'])}, {tovar(y['Vehicle'])}).")
```

Utilisation :

```
$> python extract_neg_comp.py Handling smoother
smoother(standardkart, standardkart).
smoother(standardkart, pipeframe).
smoother(standardkart, mach8).
...
smoother(splatbuggy, teddybuggy).
smoother(splatbuggy, inkstriker).
smoother(splatbuggy, splatbuggy).
```

Le code de ces scripts (python 3 et pandas requis) et les autres fichiers nécessaires ici sont téléchargeables à partir de `https://mdaquin.github.io/t/FRR_TD3_2022/files.html`.