# LTL et $\omega$-automates

14 février 2017

L'objectif de ce TP est de vous familiariser avec la logique LTL et les $\omega$-automates. On utilise pour cela Spot, une bibliothèque de manipulation de… formules LTL et d'$\omega$-automates développée au LRDE.
Demain vous utiliserez cette même bibliothèque pour vérifier un vrai système.

## 1 Se connecter à l'interface IPython de Spot

1. Rendez-vous sur `http://spot-sandbox.lrde.epita.fr/`
2. Créez un nouveau Notebook Python 3 à l'aide du boutton "New"
3. Changez le titre du Notebook : au lieu de `Untitled`, mettre votre login, ou nom, ou sha1, bref, quelque chose de différent des voisins.

Notes :
- vous êtes tous en train d'exécuter du code sur le même compte de la même machine, merci donc d'éviter les boucles infinies ou autres bétises pas drôles pour les copains.
- la machine `spot-sandbox` est remise à zéro toutes les matins (et parfois plus souvent) ; si vous souhaitez conserver votre travail, téléchargez le notebook d'ici ce soir.
- si vous avez une machine sous Debian unstable, vous pouvez installer cet environment sur votre propre machine pour ce TP et le suivant :

```
echo 'deb http://www.lrde.epita.fr/repo/debian/ unstable/' >> /etc/apt/sources.list
apt-get update
apt-get install spot python3-spot jupyter-notebook python3-ipykernel graphviz divine-ltsmin
```

et le lancer avec

```
jupyter notebook
```

## 2 initialiser Spot

Tapez le code Python qui suit dans la première cellule, et validez avec Shift-Enter.

```
import spot
spot.setup(show_default='.ab')
```

Le `show_default` utilisé ici sert à faire en sorte que les conditions d'acceptation soient montrées pour tous les automates.

## 3 Traduire une formule LTL en automate

A toutes fins utiles, les transparents du cours, qui définissent les opérateurs de base de LTL sont ici : `https://www.lrde.epita.fr/~adl/ens/mc/2017/intro-1.pdf`
Tapez le code Python qui suit et validez de la même façon :

```
spot.translate('a U b')
```

dans ce cas, vous avez obtenu un automate de Büchi, avec condition d'acceptation portant sur les états. Fair porter la condition d'acceptation sur les transitions n'apporterait rien de mieux (= pas plus petit).
Avec

```
spot.translate('G(req -> F resp)')
```

on a par défaut un automate avec condition d'acceptation sur les transitions. Spot préfère en général des automates avec conditions sur les transitions, sauf pour certaines formules simples où l'on sait qu'utiliser les états suffit. Si l'on tient absolument à avoir un automate de Büchi avec acceptation sur les états, on peut faire :

```
spot.translate('G(req -> F resp)', 'ba')
```

Comparez en particulier

```
spot.translate('(G F a) & (G F b)')
```

avec

```
spot.translate('(G F a) & (G F b)', 'ba')
```

# 4 Tester l'équivalence de deux formules LTL

Définissez les deux fonctions suivantes :

```
def implies(f1, f2):
   f1 = spot.formula(f1)
   f2 = spot.formula_Not(spot.formula(f2))
   return spot.product(spot.translate(f1), spot.translate(f2)).is_empty()

def equiv(f1, f2):
   return implies(f1, f2) and implies(f2, f1)
```

Maintenant on peut tester l'équivalence de deux formules LTL :

```
equiv('X!a', '!Xa')
```

# 5 Exercices

## 5.1 Réécriture de formules

Trouvez comment réécrire les formules suivantes en utilisant seulement `U` et `X` comme opérateurs temporels (vous pouvez aussi utiliser tous les opérateurs booléens). Par exemple la formule `Fa` peut se réécrire `true U a`.
Comment réécririez-vous les formules qui suivent ?

- `G a`
- `a W b`
- `a R b`
- `{a*; b*; c}!` (Le point d'exclamation final est important ! Ce n'est pas une formule LTL, mais Spot comprendra.)

Aidez-vous de la représentation sous forme d'automate pour deviner le sens des opérateurs dont je n'ai pas parlé. Utilisez `equiv()` pour vérifier votre réécriture.

## 5.2 Écriture de spécifications

L'exercice de traduire une spécification du français vers LTL n'est pas toujours facile. Dans ce qui suit, vérifiez que les automates produits par vos formules correspondent bien au scénario décrit en français.

- Écrivez une formule LTL pour spécifier le fait que la variable `light_on` est toujours vraie lorsque `door_open` est vraie.
- Écrivez une formule LTL pour spécifier qu'un feu peut être orange (`!v&o&!r`) pendant plusieurs instants, à condition que cela soit entre un instant où il est vert (`v&!o&!r`) et un instant celui où il est rouge (`!v&!o&r`).
- Quand on regarde l'automate de `G(req -> F resp)`, on se rend bien compte qu'on peut accepter des réponses qui ne suivent aucune demande. Comment pourrait-on spécifier qu'il y a exactement une réponse qui suit chaque requête ?
- Supposons que la variable `call_1` passe à vrai chaque fois que l'ascenseur est appelé au premier étage, et que `open_1` est vrai chaque fois que l'ascenseur est ouvert au premier étage. Écrivez une formule qui dit que si une l'ascenseur est appelé une infinité fois au premier étage, alors il sera ouvert une infinité de fois au premier. (Notez bien que cette spécification n'impose pas à l'ascenseur de passer au premier au temps de fois qu'il a été appelé.)