Máster Universitario en
Lógica, Computación e Inteligencia Artificial
Aprendizaje Automático

Apellidos:.....

Nombre :.....

Vamos a realizar ejercicios de Programación Lógica inductiva con FOIL, para ello:

- Descarga e instala SWI-Prolog http://www.swi-prolog.org/
- En el directorio, además de `foil.pl`, tenéis disponible `mi_foil.pl` con pequeñas modificaciones sobre la información al usuario.
- El fichero original de `foil.pl` está disponible en la web, p.e., http://www.eecs.wsu.edu/~holder/courses/ai2pro/code/learning/foil.pl

Empezamos lanzando Prolog

```
=========================================================
naranjo@torcal:~$ prolog
Welcome to SWI-Prolog (Multi-threaded, 64 bits, Version 5.10.4)
Copyright (c) 1990-2011 University of Amsterdam, VU Amsterdam
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to redistribute it under certain conditions.
Please visit http://www.swi-prolog.org for details.

For help, use ?- help(Topic). or ?- apropos(Word).

?-
=========================================================
```

A continuación, cargamos `mi_foil.pl` o `foil.pl`.

```
=========================================================
?- [mi_foil].
Load an example data file: [filename]
Start Foil                : foil(Predicate/Arity)
Start Foil measure time   : foil_time(Predicate/Arity)
% mi_foil compiled 0.01 sec, 45,576 bytes
true.
=========================================================
```

Dependiendo del sistema operativo, quizá te salga algún `Warning` relativo a la codificación. Ignóralo.

A continuación, cargamos el fichero con el problema de aprendizaje. En este caso, empezamos con `hija.pl`. Vemos que el fichero contiene cuatro parámetros:

- `foil_predicates`, donde indicamos la lista de pares `predicado/aridad` que podemos usar en las reglas que construye FOIL.

- `foil_cwa`. CWA son las siglas de *Closed World Assumption*, la hipótesis del mundo cerrado. En este parámetro indicamos si la usamos o no.

- `foil_use_negations`. Indica si queremos que FOIL use o no literales negados en el cuerpo de las reglas.

- `foil_det_lit_bound`. Indica el límite de profundidad de la búsqueda de literales determinados. Un literal L_m es *determinado* para la cláusula parcial $A \leftarrow L_1, \ldots, L_{m-1}$ si L_m contiene nuevas variables que tienen sólo una posible unificación para cada ejemplo positivo.

Si estamos usando `mi_foil`, además incluimos el parámetro `verbosity(N)`, donde

- `verbosity(0)` Devuelve sólo la ganancia de información

- `verbosity(1)` Devuelve el número de ejemplos positivos y negativos cubiertos

- `verbosity(2)` Devuelve también la lista de ejemplos positivos y negativos cubiertos.

Si usamos `foil.pl`, tenemos que omitir `verbosity/1` en el fichero de entrada.

El fichero incluye también los ejemplos positivos, negativos y el conocimiento base. Tras cargar `mi_foil.pl`, cargamos también `hija.pl`

```
?- [hija].
% hija compiled 0.00 sec, 4,192 bytes
true.
```

Ahora lanzamos `foil`. Para ello debemos decirle el nombre y la aridad del predicado que queremos aprender.

```
?-  foil(hija/2).
```

Dependiendo del valor del parámetro `verbosity`, obtenemos más o menos información de la ejecución del algoritmo. Por ejemplo, en este caso, al extender `hija(A,B):-` con `progenitor(B,A)`, obtenemos los siguientes valores.

```
Regla: hija(A, B):-progenitor(B, A)
Número de ejemplos positivos: 2
Ejemplos positivos: [[maria, ana], [eva, tomas]]
Número de ejemplos negativos: 1
Ejemplos negativos: [[tomas, ana]]
Número de pos. cubiertos en la cláusula ampliada: 2
Positivos cubiertos en la cláusula ampliada: [hija(maria, ana), hija(eva, tomas)]
Ganancia: 1.47393 Regla: hija(A, B):-progenitor(B, A)
```

Finalmente, FOIL devuelve la definición de `hija`

```
Definición encontrada:
hija(B, A) :- progenitor(A, B), mujer(B).
```

FOIL también nos puede dar información del tiempo que ha necesitado para aprender, mediante el predicado `foil_time(Predicado/Aridad)`.

```
?- foil_time(hija/2).

...

Definición encontrada:
hija(B, A) :- progenitor(A, B), mujer(B).

Run Time = 0.01 sec.
true.
```

Ejercicio 1. Prueba con el predicado `hija.pl` modificando su contenido.

- ¿Qué ocurre si cambiamos el parámetro `verbosity`?
- ¿Crees que cambia el tiempo de ejecución si eliminamos `'='/2` de los predicados permitidos? ¿Y si prohibimos el uso de literales negados? ¿Por qué crees eso? Pruébalo.
- ¿Y si ponemos más o menos ejemplos?

Ejercicio 2. El fichero `ejercicio_abuelo.pl` contiene ejemplos positivos para este concepto y las definicionoes de `madre/2`, `padre/2` y `progenitor/2`. Completa el fichero con los parámetros que faltan para realizar aprendizaje con FOIL usando la hipótesis del mundo cerrado. ¿Qué ocurre si eliminamos la definición de progenitor del conocimiento base?

Ejercicio 3. En el fichero `ejercicio_grafo.pl` están los ejemplos positivos y el conocimiento base para el problema de aprendizaje del predicado *camino* visto en clase. Añade los parámetros necesarios para poder realizar aprendizaje con FOIL usando la hipótesis del mundo cerrado. ¿Es necesario incluir el predicado `camino/2` en el parámetro `foil_predicates`?

Ejercicio 4. En el fichero `trenes.pl` tienes una descripción en modo texto de los trenes de Michalski (al final del fichero viene información sobre los trenes que van al este y al oeste). Adapta la información del fichero para que FOIL pueda encontrar el criterio que define a los trenes que van al este.

Ejercicio 5. Escribe el fichero de datos para aprender el predicado `par/1` con FOIL, tomando como ejemplos positivos 0, 2, 4 y 6 y como ejemplos negativos 1, 3 y 5. Usa `es_cero(0)` entre los predicados del conocimiento base.

- Prueba primero usando como conocimiento base el predicado `menos_2/2`.
- Usa después `menos_1/2` en el conocimiento base. ¿Es necesario cambiar el parámetro `foil_det_lit_bound`?