

Lógica para la Computación

Convocatoria de junio. Primer parcial, 23/11/23

Nombre:

DNI:

PCEO ☐

Plan antiguo ☐

NOTA: Es necesario acumular un mínimo de 3 ptos (el 50% de la puntuación total) en todas las pruebas escritas para sumar las prácticas de la asignatura. La duración del examen es de 1 hora.

1. (1.5 ptos) Partiendo de la implementación del algoritmo de resolución SLD aplicada por el intérprete SWIProlog utilizado en las clases de prácticas:

- (a) (0.25 ptos) Reproducir el código de la negación por fallo (**not**). Explicar la semántica declarativa de cada una de las cláusulas utilizadas.

Ver la Sección 8.2 del fichero

Documentos y Enlaces/Material de Estudio/Prolog/prolog.pdf

de la entrada Moovi de la asignatura, y que ha servido de guía a las clases.

- (b) (0.5 ptos) Probar que la negación por fallo verifica la *Ley de la Negación* con los valores **true** y **fail**.

Ver el Ejemplo 43 del fichero

Documentos y Enlaces/Material de Estudio/Prolog/prologIA.pdf

de la entrada Moovi de la asignatura, y que ha servido de guía a las clases.

- (c) (0.75 ptos) Reproducir y explicar un universo de discurso (programa + pregunta) en el que la semántica declarativa de la negación lógica no coincida con la operativa de la negación por fallo. Proponer una solución y justificar su validez mediante el árbol de resolución correspondiente.

Ver el Ejemplo 63 del fichero

Documentos y Enlaces/Material de Estudio/Prolog/prologIA.pdf

de la entrada Moovi de la asignatura, y que ha servido de guía a las clases.

2. (1.5 ptos) Dado el programa PROLOG definido por las cláusulas:

```
conj([]).  
conj([Car|Cdr]):-member(Car,Cdr),!,fail.  
conj([_|Cdr]):-conj(Cdr).
```

Responder a las siguientes cuestiones:

(a) (0.5 ptos) Describir la semántica declarativa de cada una de las cláusulas del programa.

Ver la Subsubsección 10.3.1 del fichero

Documentos y Enlaces/Material de Estudio/Prolog/prolog.pdf

de la entrada Moovi de la asignatura, y que ha servido de guía a las clases.

(b) (1 pto) Proponer y justificar una reescritura del programa, que permita eliminar el *corte* (!).

Podemos considerar el siguiente conjunto de cláusulas:

```
conj([]).  
conj([Car|Cdr]):-not(member(Car,Cdr)), conj(Cdr).
```

cuya semántica declarativa es la misma que la del programa original, pero en la que la bifurcación que definía el *corte* se sustituye por la condición explícita `not(member(Car,Cdr))`. Ello requiere re-introducir el argumento `Car` en la última cláusula.