Departamento de Cs. e Ingeniería de la Computación
Universidad Nacional del Sur

Lógica para Ciencias de la Computación

Proyecto N^{o} 1

Computación en $\mathcal{L}$: Implementación en Prolog

Primer Cuatrimestre de 2020

Objetivos

El propósito principal del presente proyecto es doble: afianzar los conceptos teóricos acerca de la teoría $\mathcal{L}$, y al mismo tiempo, mejorar el dominio del lenguaje Prolog. Para ello se pedirá implementar en Prolog el mecanismo de computación para la teoría formal $\mathcal{L}$ visto en la materia, esto es, el procedimiento efectivo basado en *Eliminación de Literales Complementarios* que permite determinar si una *fórmula bien formada* (fbf) es o no *teorema* de la teoría. También se busca desarrollar la habilidad de documentar la solución implementada, por lo que se pedirá un informe asociado a la implementación, y se ejercite la práctica de trabajar en grupo, por lo que se alentará a que sea realizado en grupos de dos personas.

Representación de fbfs en Prolog

A continuación se presenta el fragmento de código Prolog, a ser incluído al comienzo del archivo .pl del proyecto, declarando los operadores lógicos que emplearemos para representar las fórmulas bien formadas de $\mathcal{L}$: `~` (negación), `/\` (conjunción), `\/` (disyunción), `=>` (implicación), y `<=>` (equivalencia).

```
:- op(300,fx,~).      % negación, prefija, no asociativa.
:- op(400,yfx,(/\)).      % conjunción, infija, asociativa a izquierda.
:- op(500,yfx,(\/)).    % disyunción, infija, asociativa a izquierda.
:- op(600,xfx,=>).      % implicación, infija, no asociativa.
:- op(650,xfx,<=>).   % equivalencia, infija, no asociativa.
```

La notación `:- p` es una forma de solicitar al SWI-Prolog que ejecute la consulta `p`, por única vez, al cargar el archivo .pl. Luego el fragmento de código anterior consiste de una secuencia de consultas al predicado predefinido `op/3` para declarar al intérprete nuevos operadores. El primer argumento de `op/3` establece la precedencia del operador, donde un menor valor significa una mayor precedencia. Para más información acerca de `op/3` ver la documentación del predicado en el manual de referencia del SWI.

Una vez declarados, el SWI-Prolog "reconocerá" estos operadores como estructuras especiales, de forma análoga a cómo considera las estructuras aritméticas +, -, *, etc., que pueden notarse de manera infija, más cómoda, y que cuentan con reglas de precedencia y asociatividad que permiten evitar paréntesis en algunas situaciones (ver transparencia 14 de la clase práctica 3).

Además de los operadores lógicos, adoptaremos el uso de dos constantes especiales: `top`, para denotar τ (o *verdadero*) y `bottom`, para denotar $\square$ (o *falso*). Los símbolos τ y $\square$ son introducidas en la sección 2.4 de [Dav89].

A continuación se muestran ejemplos de fbfs de $\mathcal{L}$ y su representación asociada en Prolog:

$\mathcal{L}$	Prolog
$\neg\neg a \rightarrow a$	`~(~a => a)`
$(a \rightarrow b) \equiv \neg(a \wedge \neg b)$	`(a => b) <=> ~(a /\ ~b)`
$(a \vee \neg b \vee \neg a) \wedge (\neg a \vee b) \wedge (c \vee \tau)$	`(a \/ ~b \/ ~a) /\ (~a \/ b) /\ (c \/ top)`

Requerimientos de implementación

Implementar un predicado teorema/1 que reciba como argumento una fbf de $\mathcal{L}$ y determine si ésta es o no teorema, empleando refutación por resolución. El predicado teorema/1 debe implementarse en términos de los siguientes predicados auxiliares:

- fncr/2, que reciba una fbf de $\mathcal{L}$ como primer argumento y retorne como segundo argumento su *forma normal conjuntiva reducida (fncr)*.
- refutable/1, que reciba una fbf en fncr y determine si ésta es o no refutable, es decir, si existe o no una derivación por resolución de bottom a partir de ella.

Ejemplos:

```
?- teorema( (a => (b => c)) => ((a => b) => (a => c)) ).
true

?- fncr( ~((a => (b => c)) => ((a => b) => (a => c))), FNCR ).
FNCR = (~a \/ ~b \/ c) /\ (~a \/ b) /\ a /\ ~c

?- refutable( (~a \/ ~b \/ c) /\ (~a \/ b) /\ a /\ ~c ).
true
```

Además se pide que cada predicado imprima **mensajes por consola** (usando write/1 y writeln/1) que permitan entender el proceso seguido por el predicado para obtener la respuesta final entregada.
Importante: es fundamental que se respete la signatura especificada anteriormente para los tres predicados que se requiere implementar, esto nos permitirá automatizar la corrida de casos de tests para corregir los proyectos.

Requerimientos de Documentación

Se deberá realizar un informe que **explique** claramente la **implementación** en Prolog realizada. Puede aprovechar el informe para destacar características positivas de la resolución, y documentar cualquier otra observación que considere pertinente. Además deberán incluir **casos de tests**, con sus respectivas respuestas, para los tres predicados que se requiere implementar.

Se recomienda estructurar el informe de manera top-down, comenzando con una descripción a alto nivel de la implementación.

Importante: en el desarrollo de software, la documentación de la implementación constituye un elemento fundamental. Es por esto que, para la evaluación del presente proyecto, se dará suma importancia a la calidad (claridad y completitud) del informe entregado.

Condiciones de Entrega

1. Las comisiones pueden estar conformadas por hasta 2 integrantes, y deben ser registradas en la página de la materia. A cada comisión se le asignará un integrante de la cátedra, quien hará el seguimiento y corregirá el proyecto de la comisión.

2. La fecha límite de entrega del presente proyecto se encuentra publicada en la página de la materia. Los proyectos entregados fuera de término recibirán una penalización en su calificación, la cual será proporcional al retraso incurrido.

3. La entrega del proyecto consiste del envío por mail de la resolución del proyecto y versión electrónica del informe.

   - Enviar por mail directamente al integrante de cátedra asignado a la comisión, con copia al asistente (en caso de no ser el asignado). Mails:
     - Federico: `fedeschmidt.10+LCC@gmail.com`
     - Diego: `dieorbe96+LCC@gmail.com`
     - Germán: `germang04+LCC@gmail.com`
     - Mauro (asistente): `mgomezlucero+LCC@gmail.com`
   - Asunto del mail: "Proyecto LCC - Comisión <`Ap.y Nom. Integrantes`>"
   - Adjunto: un .zip conteniendo 1) un archivo PR-1.pl con la implementación Prolog y 2) un pdf con la versión electrónica del informe.

Referencias

[Dav89] Davis, R. E. *Truth, Deduction and Computation: logic and semantics for computer science*. Computer Science Press, 1989.