Facultad de Ciencias - UNAM
Lógica Computacional 2023-2
Práctica 5: Algoritmo de `Hao Wang`

Javier Enríquez Mendoza
Ramón Arenas Ayala
Óscar Fernando Millán Pimentel
Kevin Axel Prestegui Ramos

22 de Abril de 2023
Fecha de entrega: 7 de Mayo de 2023

1. Introducción

`Prolog` es un lenguaje de programación lógico que pertenece al estilo declarativo. En esta práctica continuaremos trabajando en Prolog, ya que las aplicaciones de este son muy amplias en el estudio de la lógica computacional.

Para esta práctica utilizaremos un algoritmo de lógica computacional que trabaja con preposiciones lógicas, el algoritmo de Hao Wang, que puede ser modelado mediante el uso de `Prolog`.

2. Objetivo

El objetivo de esta práctica, es modelar el algoritmo de Hao Wang mediante los conocimientos adquiridos en el laboratorio y de la práctica pasada de `Prolog`, así como reforzar sus conocimientos de lógica de predicados.

3. Justificación

El motivo de ser de esta práctica, es el hecho de indagar más sobre la lógica de predicados, mediante la implementación de distintos algoritmos, uno de los cuales será el Algoritmo de Hao Wang, ya que presenta una interesante forma de abordar la lógica de predicados.

Como lo hemos mencionado, `Prolog` será una herramienta de trabajo durante el curso, ya que nos permite una forma más directa para trabajar con fórmulas proposicionales y lógicas.

4. Preparativos

4.1. Instalación de SWI - Prolog

Antes de comenzar a escribir programas en cualquier lenguaje de programación, es necesario instalar, ya sea, un compilador o un intérprete. Para el caso de `Prolog`, existen diversos compiladores, uno de los más conocidos y el cual se usa en esta sección es `SWI-Prolog`.

Para el caso de Windows y MacOS existen archivos binarios que se pueden descargar en `https://www.swi-prolog.org/download/stable`. En el caso de Linux, los detalles de la instalación se encuentran en `https://www.swi-prolog.org/build/unix.html`.

Una vez concluida la instalación, se puede verificar que esta haya sido exitosa ejecutando el comando `swipl -version` , lo que mostrará la versión instalada.

`SWI-Prolog` también funciona como un ambiente interactivo, el cual puede iniciarse ejecutando el comando `swipl` , lo que mostrará un mensaje de bienvenida.

Entre los comandos más útiles para interactuar con el ambiente se tienen:

- `help(X).`; que sirve para mostrar más información acerca de X.
- `halt.`; detiene la ejecución de `SWI-Prolog`.
- `[archivo1, archivo2, ...].`; que sirve para cargar las definiciones de archivos y poder usarlas en el ambiente. Alternativamente, también puede cargar el programa pasando el nombre del archivo como parámetro a `SWI-Prolog: swipl -s program.pl.`
- `make.`; para actualizar las definiciones de algún archivo previamente cargado en el ambiente.

Es importante destacar que el punto al final de cada comando es importante, de esta manera `Prolog` sabe que ahí termina el comando.

5. Desarrollo de la práctica

5.1. Algorimo de Hao Wang

Para esta practica vamos a implementar un probador de teoremas para fórmulas de lógica de predicados sin variables y cuantificadores $\neg$, $\wedge$, $\vee$, como los únicos conectores proposicionales. Tal fórmula libre de cuantificadores y variables phi es válida si el secuente $\Rightarrow \phi$ es válido. Ahora, para la validez de las secuencias ϕ_1 ... $\phi_n \Rightarrow \psi_1$ ... ϕ_m que consisten en fórmulas libres de cuantificadores y libres de variables.

La validez de estas secuencias está dictaminada por reglas. Las reglas de este cálculo tenían la útil propiedad de que conservaban la validez en ambos sentidos. Interpretados en este sentido conducen a enunciados verdaderos. Estas reglas podremos ver en la siguiente sección.

5.2. Instrucciones

Haciendo uso de lo visto durante la clase, se realizará la implementación de un demostrador de lógica de predicados siguiendo el algoritmo Hao Wang utilizando el lenguaje de programación del paradigma lógico Prolog.

Para esto solo tomaremos en cuenta proposiciones cuyos únicos conectivos son la negación, el and y el or,los cuales representaremos en el lenguaje con la notación neg(A), and(A,B) y or(A,B). Además de esto para los conjutos de fórmulas los representaremos por medio de listas.

Las reglas que se implementaran son las siguientes:

1. Se dice que $\Gamma \vdash \Delta$ es válido si Γ y Δ tienen al menos un elemento en común.
2. Γ_1, $\neg\phi$, $\Gamma_2 \vdash \Delta$ es válido si Γ_1, $\Gamma_2 \vdash \phi$, Δ es válido.
3. $\Gamma \vdash \Delta_1$,$\neg\phi$, Δ_2 es válido si ϕ, m $\Gamma \vdash \Delta_1$, Δ_2 es válido.
4. Γ_1, $(\phi \wedge \psi)$, $\Gamma \vdash \Delta$ es válido si ϕ, ψ, Γ_1, $\Gamma_2 \vdash \Delta$ es válido.
5. $\Gamma \vdash \Delta$, $(\phi \vee \psi)$, Δ_2 es válido si $\Gamma \vdash \phi$, ψ, Δ_1, Δ_2 es válido.
6. Γ_1, $(\phi \vee \psi)$, $\Gamma_2 \vdash \Delta$ es válido si tanto ϕ, Γ_1, $\Gamma_2 \vdash \Delta$ como ψ, Γ_1, $\Gamma_2 \vdash \Delta$ son válidos.

5.3. Ejercicios

NOTA: Para esta práctica no se puede utilizar ninguna Clausula ya definida en prolog. Solo se pueden definir cosas nuevas y utilizar los operadores de «AND» (,), «OR» (;) y «NOT» (not).

1. (1 punto) Escribe una regla que nos diga si el predicado elem(X,M) es verdadero. Este debe ser verdadero y X pertenece a la lista M.

   Ejemplos:

   - `elem(3,[1,2,3])`
     `true`
   - `elem(A,[1,2,3])`
     `A=1`
     `A=2`
     `A=3`
     `false`
   - `elem(4,[1,2,3])`
     `false`
   - `elem(B,[25])`
     `A=25`
     `false`

2. (1 puntos) Escribe una regla que nos diga si el predicado intersect(M,N) sea verdadero. Para esto se recomienda utilizar el predicado elem. Este debe ser verdadero si M y N tienen al menos un elemento en común.

   Ejemplos:

   - `intersect([1],[1,2,3])`
     `true`

- `intersect([8,6,7,2],[1,2,3])`
  `true`
- `intersect([4,5],[1,2,3])`
  `false`
- `intersect(2,[1,2,3])`
  `false`

3. (1 punto) Escribe una regla que nos diga si el predicado delete(X,L,NewL) es verdadero. Donde X es un elemento de una lista,L y NewL listas, y el predicado es verdadero si NewL es igual L pero eliminadole X (solo la primera aparición).

   Ejemplos:

   - `delete(2,[1,2,3],[1,3])`
     `true`
   - `delete(2,[1,2,3],NL)`
     `NL=[1,3]`
     `false`
   - `delete(2,[1,2,3,2,4],NL)`
     `NL=[1,3,2,4]`
     `false`

4. (7 puntos) Ecribe un conjunto de reglas para saber si el predicado wang(G,D) es verdadero. Este conjunto de reglas debe ser correspondiente a las reglas enumeradas en la primera parte de esta práctica.

   Hints:

   - *Utiliza los predicados que se realizaron en los puntos anteriores.*
   - *Recuerda que se pueden utilizar variables en prolog (representadas con un string que comienza con una letra mayuscula) y al poner esas variables en un predicado Prolog te da posibles valores de esa variable para que el predicado se cumpla.*

5. (1 punto) Escribe una regla que nos diga si el predicado valid(F) es verdadero. Esta regla debera utilizar el predicado wang.

   Hint:

   - *Razona sobre que conjuntos de fórmulas se debe verificar Wang para saber si F es válido.*

6. Especificaciones de entrega

- **GitHub Classroom:** Para realizar la entrega de la práctica se utilizará la plataforma de GitHub Classroom. `https://classroom.github.com/a/6509fEZB`
- **Equipos:** La práctica puede realizarse en equipos de hasta tres personas. Esto siempre y cuando se respete la política de trabajo colaborativo y podamos ver que ambos miembros del

equipo contribuyen en la elaboración de la práctica. En caso contrario, las prácticas serán exclusivamente individuales.

- **Fecha de entrega:** La fecha de entrega será la indicada al principio de este documento. No se recibirá ninguna práctica en fechas posteriores a la fecha indicada, al menos que el profesor indique una prórroga.
- **Flujo de trabajo:** Se recomienda llevar un flujo de trabajo adecuado, en el cual se realicen commits constantemente y no hacer todo al final y en un solo commit. Esto ya que de esta manera podemos comprobar la colaboración de todos los miembros del equipo, dar feedback a lo largo del tiempo de trabajo para mejorar la entrega de las prácticas, facilitar la resolución de dudas, etc.
- **Evaluación:** Para la evaluación la práctica se someterá a un conjunto de pruebas, y además se realizará una revisión del código. Comprobándose que se cumplan las condiciones indicadas para cada ejercicio.
- **Compilado:** Cualquier práctica que al ser descargada no compile, **será evaluada con una calificación de 0**.
- **Datos personales:** Se debe agregar un documento `README.md` en el directorio raíz de sus repositorios, con sus datos personales. En caso de no ser indicados, la calificación no podrá ser asignada.
- **Limpieza y estructura:** en caso de no tener una estructura clara y limpieza dentro de sus repositorios, la calificación se verá afectada negativamente.

7. Sugerencias y Notas

- **Dudas**: Pueden preguntar sus dudas a través de telegram, o correo por el canal que ustedes deseen. Pero les recomendamos preguntar a través del grupo de telegram para que sus demás compañeros también puedan aclarar dudas similares a la suya.
- **Limitaciones**: Pueden utilizar funciones predefinidas siempre y cuando su utilización no derive en que se pierda el objetivo académico del ejercicio que se está realizando. Si se tiene duda acerca del poder utilizar algo, pueden preguntarlo.

Buena suerte a todos! ☺☺☺