Representación del conocimiento
Tarea 1

Angel García Báez
Alumno de la Maestría en Inteligencia Artificial

IIIA Instituto de Investigaciones en Inteligencia Artificial
Universidad Veracruzana
Campus Sur, Calle Paseo Lote II, Sección 2a, No 112
Nuevo Xalapa, Xalapa, Ver., México 91097

ZS24019400@estudiantes.uv.mx

23 de marzo de 2025

1. Cuestionario en Socrative

Cree una cuenta en https://www.socrative.com para ingresar como **estudiante** en el salón **2025RC**. Ahí encontrará un test de opción múltiple a resolver (rc-01). El test puede resolverse en cualquier orden y estará abierto hasta el día de entrega de la tarea. El único medio de entrega es por esta vía. [20/100]

El cuestionario de Socrative fue respondido bajo el nombre/alias de **Angel GB**.

2. Articulo de Yoav Shoham

Lea el artículo de Yoav Shoham titulado: "Why Knowledge Representation Matters, A personal story: From philosophy to software". Responda las siguientes preguntas de manera breve y concisa para su discusión en clase.[30/100]

a) ¿En qué sentido la atención que recibe actualmente la IA es diferente de la atención que recibía en los 90s?

   R: La IA recibía especial atención en el enfoque lógico (IA simbólica) donde brillaban los sistemas expertos y lenguajes de programación como Prolog y Lisp. En la actualidad (para el contexto del articulo Shoham (2015) y sigue siendo valido en 2025) la IA actual se a enfocado principalmente en el machine learning basado en algoritmos estadísticos y el procesamiento de grandes volúmenes de información de todo tipo.

b) De un ejemplo de aplicación donde el enfoque actual de la IA es muy exitoso.

   R: Uno de estos casos de aplicación donde el enfoque actual de la IA es exitoso, es el caso del reconocimiento y procesamiento de imágenes.

c) Explique por qué Shoham considera el caso considerado como *filosofía aplicada*.

   R: Se considera así porque llevo conceptos filosóficos (teóricos) hacia un caso de aplicación (el desarrollo del programa) de manera inteligente y fue un caso de éxito.

d) Explique qué significa ese término en el contexto del artículo.

   R: Lo maneja como la representación de nociones de sentido común con un énfasis en la claridad semántica.

e) ¿Qué significado tiene la abreviatura AGM en el artículo?

   R: Dentro del contexto del artículo, AGM es la abreviatura del modelo Alchourrón, Gärdenfors y Makinson para la revisión de creencias Huber (2013).

f) Explique en qué consiste el problema de mantenimiento de creencias e intenciones.

R: En el marco de trabajo inicial, los agentes únicamente contaban con una sola base de datos que era la responsable de almacenar sus creencias y también de garantizar su consistencia. Después el marco fue expandido o enriquecido para que ahora sean 2 bases de datos, una para creencias y la otra para las intenciones del agente, el problema ahora es más grande, debido a que es necesario que cada una de estas 2 bases de datos sean consistentes individualmente y además, que tengan consistencia mutuamente.

g) ¿Qué es un PTA?

R: Un PTA es la abreviatura de Personal Assistant Time y tiene la tarea de ayudar a manejar el tiempo y las tareas del usuario.

h) ¿Cuáles son los tres pilares en la construcción de Timeful 1.0?

R: El primer y más destacable de los pilares fue permitir la representación de las tareas a los usuarios de una forma natural (casi que platicada) debido a que esta funciona principalmente gracias a la representación del conocimiento. El segundo pilar fue la aplicación de machine learning junto con otros algoritmos para resolver el problema de la optimización de los tiempos. El tercer pilar fue la ciencia del comportamiento con la cual, se pudo construir un entorno que permitiera corregir errores naturales en la gestión del tiempo, como lo podían ser eventos no cumplidos en tiempo (procrastinación) o la sobre estimación de tiempo.

i) ¿Qué significa que citas, eventos, etc. sean intenciones?

R: Haciendo una breve simplificación de la realidad, las citas, los eventos, las actividades recreativas (deporte, entretenimiento, etc..) y el descanso comparten la característica de que hacen uso del recurso más finito y valioso que tenemos, el tiempo. Pese a que sucedan bajo diferentes contextos se necesita disponer de tiempo para participar o realizar cualquiera de las actividades.

j) Explique el concepto de IO.

R: IO u Objeto intencional, es un modelo de datos donde se representa mediante un vector de atributos, en donde se guarda la descripción textual, los atributos relacionados al tiempo (cuando va a ocurrir, cuando debería ser, la duración y toda especificación respecto al tiempo con diferentes grados de precisión), las condiciones para que sea llevada a cabo la intención como el lugar donde va a ser, entre otros atributos.

k) ¿Qué clases de intenciones se consideraban en la versión de 2015?

R: Hasta la versión del 2015 se tenían 4 tipos de clases para las intenciones: La clase de eventos (como reuniones), tareas (como hacer una llamada telefónica o mandar un correo), los hábitos (como salir a trotar x cantidad de veces por semana) y proyectos (tales como escribir un reporte extenso.

l) Explique uno de los mejores ejemplos de como la filosofía influyó las decisiones en el diseño de Timeful.

R: En el primer ejemplo, se explica cómo la marca de realizado.[es]taba implementada únicamente para las tareas, pero no para los eventos ni otras clases de IO. Antes de decidir agregar esta checkmark a todas las demás clases (ya que, al ser IO, podían hacerlo con relativa facilidad), se detuvieron a reflexionar sobre el verdadero propósito de la función. Más allá de la estética y la comodidad, se encontraron con una cuestión filosófica: ¿realmente tenía sentido? Una tarea implica una intención (lo que se debe hacer) y un compromiso (llevarla a cabo), lo que conlleva un seguimiento hasta su conclusión, momento en el que se marca como realizada. En cambio, los eventos simplemente se programan, ocurren bajo ciertas condiciones y se asiste a ellos, sin un seguimiento o compromiso en el mismo sentido que una tarea. Por esta razón, decidieron incluir la checkmark en todos los IO, excepto en los eventos, respetando la relación entre intención, compromiso y seguimiento.

m) ¿De qué manera contribuyó la representación de conocimiento en el desarrollo de Timeful?

R: A lo largo del artículo, se destaca la importancia de la representación del conocimiento, señalando que su mayor fortaleza radica en la forma en que lograron diseñar una estructura de datos capaz de establecer una comunicación directa y amigable con el usuario en sus actividades. Como menciona el autor al final del texto, el verdadero

éxito del producto se debió a la manera en que se pensó y diseñó su arquitectura principal los IO que representan las cosas por hacer, aplicando técnicas de representación del conocimiento (KR) y realizando un análisis filosófico de sus fundamentos y propósitos.

n) ¿Qué problemas hay al querer usar aprendizaje automático en una aplicación como Timeful?

R: Si se intentara manejar desde un enfoque más orientado al machine learning y las técnicas actuales, surgirían varios problemas. El primero radica en la representación de los eventos como datos de entrada, lo que crearía una barrera significativa entre el producto y el usuario, ya que implicaría definir rígidamente la incorporación de su IO al sistema bajo un formato de trabajo estandarizado. Además, el procesamiento de los IOs sería más complejo, dado que los enfoques de machine learning dependen de grandes volúmenes de datos para generar estadísticas y ofrecer resultados plausibles, lo que, a su vez, conllevaría un alto costo computacional. Por lo que su enfoque desde esta perspectiva en la actualidad, podría ser abordado pero probablemente sea más complicado.

ñ) ¿Cuál es su conclusión tras la lectura del artículo?

R: En conclusión, destaco la importancia de conocer distintos paradigmas de programación, así como sus fortalezas y debilidades, para desarrollar herramientas según las necesidades de cada caso. Además, comprender la representación del conocimiento es fundamental para comprender, diseñar y estructurar adecuadamente todo el proceso lógico de la información que se manipulará, permitiendo así crear programas limpios y fáciles de manejar.

3. Realizar una demostración

Pruebe que $\neg p \vee q \vdash p \implies q$. Demuestre que también son sintácticamente equivalentes. [20/100]

Para la realización de esta demostración, se hizo uso del conocimiento adquirido durante las clases, así como de las notas largas Guerra-Hernández (2025) a base de prueba y error, como si se intentara armar un rompecabezas, tratando de entender las reglas fundamentales para poder mover las piezas.

3.1. Primera demostración

Primero, se hizo la demostración usando la parte izquierda $\neg p \vee q$ para probar que de ahí se sigue $p \implies q$:

1.	$\neg p \vee q$	premisa
2.	$\neg p$	supuesto
3.	p	supuesto
4.	$\perp$	contradicción con 2 y 3
5.	q	(por eliminación de la contradicción 4)
6.	$p \implies q$	introducción de la implicación (3-5)
7.	q	supuesto
8.	p	supuesto
9.	q	copia de 7
10.	$p \implies q$	por introducción de la implicación de 8 y 9
11.	$p \implies q$	eliminación de la disyunción sobre la premisa 1

3.2. Segunda demostración

A continuación se hizo la demostración usando la parte derecha $p \implies q$ para probar que de ahi se sigue $\neg p \vee q$:

1.	$p \implies q$	Premisa
2.	$p \vee \neg p$	Ley del medio excluido (LEM)
3.	p	Supuesto
4.	q	por Modus Ponens de 1 y 3
5.	$\neg p \vee q$	Introducción de la disyunción, segunda variante
6.	$\neg p$	Supuesto
7.	$\neg p \vee q$	Introducción de la disyunción, primera variante
8.	$\neg p \vee q$	eliminación de la disyunción sobre la premisa 2

4. Inciso 4

Descargue el repositorio

`https://github.com/flijnzaad/natural-natural-deduction`

que implementa un demostrador de teoremas basado en deducción natural, implementado como una búsqueda en Prolog. Prepare un reporte respondiendo a las siguientes preguntas. Las respuestas se discutirán en clase.

- ¿Cómo se representan las preguntas al sistema (*queries*)?
- ¿Cómo se representan las demostraciones?
- De un ejemplo de como funciona el predicado `connectives/3`.
- ¿Qué hace el predicado `provesWrap/3`?
- ¿Qué tipo de búsqueda implementa el predicado `provesIDS/7`?
- ¿Qué implementa el predicado `proves/7`?
- ¿Cómo se implementa la introducción de la conjunción?
- ¿Qué heurística usa la búsqueda de este demostrador?
- ¿Donde se implementan las sub-pruebas (cajitas)?

4.1. Reporte

El programa "Natural.[es] un demostrador automático basado en las reglas de deducción natural de la lógica. Está implementado en Prolog y cuenta con una interfaz que permite manipular los resultados de las consultas de Prolog desde Python, generando un reporte en formato TeX con los resultados de la demostración. Por defecto, el programa incluye 38 consultas en forma de reglas Prolog, las cuales se utilizan para probar su funcionamiento, realizar las demostraciones y generar el reporte final en formato PDF con Python.

Las consultas implementadas siguen una estructura definida de reglas, donde la cabeza, que es un predicado, toma la variable X. Esta construcción se denota mediante el número de la consulta, como $q_1, q_2, \ldots, q_n$. El cuerpo de cada regla se compone de tres partes principales: la premisa, que contiene los hechos que queremos probar; la conclusión, que almacena el resultado encontrado; y el predicado `provesWrap/3`, que se encarga de darle el formato adecuado a la salida.

En el programa, las demostraciones se representan como una lista en la que se van almacenando los pasos válidos que el programa sigue, comenzando con la primera línea de premisa. A medida que avanza, se van construyendo las pruebas y subpruebas necesarias hasta llegar a la conclusión, respetando el índice o número de fila de cada paso, tal como se hace en las demostraciones manuales.

El programa implementa un predicado llamado connectives/3, cuya tarea es analizar las fórmulas lógicas que se van construyendo para extraer los conectivos (como and, or, not, if, etc.) que aparecen en ellas. Esto resulta especialmente útil, ya que al identificar estos conectivos, el programa puede seleccionar adecuadamente las reglas de inferencia que se deben aplicar según corresponda. Para lograr esto, el predicado utiliza una estructura recursiva por casos, explorando las premisas y conclusiones hasta extraer todos los conectivos y devolver una lista vacía como resultado.

De forma muy simple, si se tienen las premisas $and(p, q)$ y $if(r, s)$ y las conclusiones $or(p, not(s))$, el código debería devolver una lista con los operadores presentes como se muestra en el ejemplo:

```
?- connectives([and(p,q),if(r,s)], or(p,not(s)), L)
L = [and, if, or, not]
```

Por otro lado, el predicado `provesWrap/3` funciona como una envoltura del predicado principal que se desea demostrar, recopila información sobre los conectivos presentes en las premisas y conclusión, permite estandarizar el flujo de trabajo inicial y delega el trabajo a `provesWrap/5` que es una

versión más completa del predicado que toma en cuenta futuras adiciones a las premisas .

El predicado encargado de construir el árbol de búsqueda en Prolog es `provesIDS/7`. Este predicado implementa un algoritmo de búsqueda "profunda"(o "deep search") de manera iterativa. Su función es tomar las premisas disponibles y buscar formas lógicas que satisfagan correctamente la demostración. Si no logra encontrar una solución en la profundidad inicial, el algoritmo aumenta gradualmente la profundidad de búsqueda hasta encontrar una demostración válida. Una vez que la encuentra, hace el corte y continúa con el siguiente paso.

Por otro lado, es `proves/7` el encargado de implementar las reglas de deducción natural que permiten introducir o eliminar elementos de las formulas lógicas de forma recursiva por casos para que `provesIDS/7` sea el que induzca la búsqueda.

Dentro del abanico de reglas para introducir y eliminar elementos de las formulas, llama la atención la forma en que se implementa la introducción de la conjunción, la regla se fija si ya se ha demostrado X (en la linea $N1$) y Y (en la linea $N2$), si se cumplen ambas condiciones, entonces puede realizar la introducción de la conjunción $X \wedge Y$.

Parece que el demostrador implementa una heurística que busca minimizar la cantidad de líneas necesarias para probar las demostraciones, dado el modo en que se induce el árbol de búsqueda dentro del programa. Esto se evidencia principalmente en la búsqueda IDS, que intenta reducir al mínimo el número de pruebas posibles. Esto se debe a que un aumento excesivo en la profundidad de la búsqueda incrementa el espacio de búsqueda, lo que complica la demostración y aumenta tanto el tiempo como los recursos computacionales necesarios.

Finalmente, las premisas no se demuestran directamente, es necesario ir construyendo sub pruebas que prueban los argumentos a utilizar, para ello, dentro de la lógica del programa se manda a llamar al predicado `subproof/12` dentro de los casos de la función `proves/7`.

A modo de probar el potencial de dicho programa, se agrego al archivo de *queries.pl* la siguiente regla que representa la demostración del inciso 3.2 probado anteriormente a mano:

```
q40(X) :-
Premises = [line(1, or(neg(p), q), premise, 0)],
Concl    = line(_, if(p, q), _, _),
provesWrap(Premises, Concl, X).
```

El resultado arrojado por el programa en formato latex es el siguiente:

Proof 40

1. $(\neg P \vee Q)$	
2. P	
3. $\neg P$	
4. $\bot$	$\bot$ Intro: 2, 3
5. Q	$\bot$ Elim: 4
6. Q	
7. Q	Reit: 6
8. Q	$\vee$ Elim: 1, 3–5, 6–7
9. $(P \rightarrow Q)$	$\rightarrow$ Intro: 2–8

Figura 1: Demostración de $\neg P \vee Q \vdash (P \implies Q)$

El resultado es prácticamente el mismo, salvo que tiene la notación más formal y bonita.

Es importante señalar que se intentó realizar el complemento de la demostración $(p \implies q)$, pero el árbol de búsqueda creció rápidamente, alcanzando una profundidad de 12 sin generar ningún resultado. Se sospecha que, debido a esta limitación, una posible área de mejora del programa sería incorporar la ley del medio excluido, la cual, como se mostró en el inciso 3.1, es fundamental para la correcta resolución de la demostración.

5. Referencias

Referencias

Guerra-Hernández, A. (2025). Deducción natural. Notas de clase, Capítulo 3, Curso de Representación del Conocimiento, Universidad Veracruzana.

Huber, F. (2013). Belief revision i: The agm theory. *Philosophy Compass*, 8(7):604–612.

Shoham, Y. (2015). Why knowledge representation matters. *Communications of the ACM*, 59(1):47–49.