Nuevo Resumen de Teoría de IA

1-Tema

El campo de la IA no solamente pretende entender, sino que quiere construir entidades inteligentes artificiales.

Existen diferentes tipos de definición de IA:

> **Humano vs Racional**: se centra en cómo se compara la IA y sus capacidades cognitivas. La perspectiva "**humana**" busca replicar la inteligencia humana y sus capacidades cognitivas, mientras que la perspectiva "**racional**" se enfoca en crear sistemas inteligentes que puedan tomar decisiones óptimas y lógicas, independientemente de si se asemejan o no a la inteligencia humana.
>
> **Pensamiento vs Comportamiento**: se refiere a qué aspectos de la inteligencia se consideran más importantes. La perspectiva "**pensamiento**" se centra en la capacidad de una máquina para simular procesos mentales complejos (el razonamiento, la compresión del lenguaje natural y la resolución de problemas abstractos). La perspectiva "**comportamiento**" se enfoca en la capacidad de una máquina para llevar a cabo tareas específicas y producir resultados inteligentes.

El **machine learning** es el subcampo de las ciencias de la computación y una rama de la IA, cuyo objetivo es desarrollar técnicas que permitan que las computadoras aprendan. Permite a las máquinas aprender de manera autónoma a partir de los datos y mejorar su rendimiento con la experiencia.

El **Test de Turing** es una prueba propuesta por Alan Turing para evaluar la capacidad de una máquina de exhibir comportamiento inteligente indistinguible del de un ser humano. Consiste en la interacción de un evaluador con una máquina sin saber qué está interactuando con una máquina, de manera que si no es capaz de distinguir si se trata de una máquina o de una persona, se considera que la máquina posee IA.

El objetivo principal de **GPS** era resolver problemas complejos mediante el uso de un enfoque basado en reglas de producción. Utilizaba una representación de estados y operadores para modelar el problema y realizar operaciones de búsqueda y manipulación de datos para encontrar soluciones.

2-Tema

La **función de agente** define como un agente inteligente procesa sus percepciones para determinar las acciones que debe tomar en función de sus objetivos y conocimientos. Es el componente fundamental que permite a un agente interactuar con su entorno y lograr sus metas.

El **programa de agente** es la implementación concreta de la función de agente en un sistema de inteligencia artificial. Es el código o conjunto de instrucciones que define cómo el agente percibe su entorno, procesa la información, toma decisiones y actúa en consecuencia.

Un **Agente Inteligente** es un algoritmo que puede automáticamente llevar a cabo numerosas tareas basando sus decisiones en el entorno y sus experiencias.

El término **percepción** se refiere a las entradas recibidas en un instante dado. Una **secuencia de percepción** de un agente es la historia completa de todo lo que el agente ha percibido alguna vez. La elección de acción de un agente en un instante dado nunca puede depender de nada que **NO** haya percibido.

Un **Agente Racional** es el que hace las cosas bien, es decir, selecciona la acción de más éxito. Utiliza **su percepción y conocimiento** para tomar decisiones óptimas o cercanas a la óptima en función de sus objetivos y del entorno en el que se encuentra. Es mejor que el agente tenga objetivos relativos a como se debe comportar.

La definición de **Agente Racional** puede ser: para cada posible secuencia de percepción, un agente racional debe seleccionar una acción que se espera que maximice su medida de desempeño, dada la evidencia proporcionada por la secuencia de percepción y cualquier conocimiento incorporado que tenga el agente.

La **racionalidad** necesita de la recopilación de información para que un agente pueda tomar decisiones inteligentes y adaptarse a su entorno. La **exploración** es crucial para descubrir y obtener nueva información sobre el entorno. A través del **aprendizaje**, el agente puede adquirir conocimiento sobre lo que percibe y mejorar su capacidad para tomar decisiones informadas. Sin embargo, solo confiar en el conocimiento previo limita la autonomía del agente, ya que no puede adaptarse a situaciones nuevas o desconocidas. El aprendizaje permite que un agente racional se independice del conocimiento previo y tenga la capacidad de adquirir nuevos conocimientos, lo que le permite tomar decisiones más inteligentes y adaptarse a diferentes circunstancias.

Los entornos de la tarea son los problemas para los cuales un agente inteligente es la solución.

La descripción **PEAS (perfomance measure, environment, actuators, sensors)** describe de manera completa un entorno de tarea y agrupa todos los criterios que necesitamos para tener un agente que actúe de forma racional. Notación útil para describir un agente racional de una manera estandarizada y fácil de entender.

- **Performance Measure**: Es la métrica o criterio que se utiliza para evaluar qué tan bien está realizando su tarea el agente. Define el éxito o el fracaso en función de los objetivos que se le han asignado.
- **Environment**: Es el contexto o el espacio en el que opera el agente. Incluye todo lo que rodea al agente y con lo que interactúa, como el mundo físico, otros agentes, datos de entrada, etc.
- **Actuators**: Son los mecanismos o dispositivos a través de los cuales el agente actúa o realiza acciones en su entorno. Pueden ser motores, pantallas, altavoces, brazos robóticos, entre otros, dependiendo del tipo de agente y de las acciones que se le permita realizar.
- **Sensors**: Son los dispositivos o mecanismos que permiten al agente percibir o recibir información del entorno. Pueden ser cámaras, micrófonos, sensores táctiles, sensores de temperatura, entre otros, dependiendo de la forma en que el agente interactúa con su entorno y recopila información.

Entorno completamente observable vs parcialmente observable:

- Un entorno es **completamente observable** si los sensores del agente te dan toda la información relevante para tomar decisiones.

- Un entorno es **parcialmente observable** si los sensores del agente no te dan el estado completo o preciso del entorno.

Agente Único vs Multi-Agente:

- La diferencia fundamental radica en la cantidad de entidades inteligentes que participan en la interacción. El **agente único** es autónomo en el sentido de que puede operar y tomar decisiones de manera independiente sin la intervención de otros agentes, por otro lado, el **multi-agente**, consta de múltiples entidades inteligentes (agentes), que interactúan entre sí y con su entorno, para lograr objetivos comunes o resolver problemas complejos.

Determinista vs no determinista (se conocen o no las probabilidades):

- La diferencia entre un entorno de tarea **determinista** y uno **no determinista** radica en la previsibilidad y la certeza de los resultados de las acciones tomadas por un agente en ese entorno. En un **entorno determinista**, el resultado de las acciones del agente es completamente predecible, es decir, si el agente toma una acción específica en un estado dado, el resultado será siempre el mismo, por otro lado, en un **entorno no determinista**, el resultado de las acciones del agente puede ser incierto o variar en diferentes ejecuciones. Puede haber factores aleatorios o externos que influyan en los resultados de las acciones. Por ejemplo, si un agente está navegando en un entorno donde hay condiciones climáticas cambiantes, las consecuencias de sus acciones pueden ser afectadas por factores impredecibles como el viento o la lluvia.

Episódico vs secuencial:

- Es **episódico** si el próximo episodio no depende de otro. Cada episodio es considerado como un evento independiente, y el agente no retiene información de episodios anteriores para utilizarla en episodios futuros. Un ejemplo de un entorno episódico podría ser un juego de ajedrez donde se juegan partidas independientes, y cada partida se considera como una entidad separada sin relación con las partidas anteriores.
- Es **secuencial** si las decisiones actuales afectan a las decisiones futuras. Agente que trabaja en un ambiente en el cual las acciones y las percepciones son continuas y el agente retiene información de episodios anteriores para utilizarla en episodios futuros. Un ejemplo de un entorno secuencial podría ser un juego de ajedrez donde los movimientos realizados en un momento dado tienen un impacto directo en los movimientos futuros y en el resultado final del juego.

Estático vs dinámico:

- En un entorno **estático**, no hay cambios en el entorno una vez que el agente comienza a interactuar con él, es decir, las propiedades del entorno permanecen constantes a lo largo de la interacción del agente. Un ejemplo de un entorno estático podría ser un juego de mesa donde las reglas y el tablero no cambian a lo largo del juego. Por otro lado, en un entorno **dinámico**, el entorno puede cambiar durante la interacción con el agente, es decir, las propiedades del entorno pueden ser modificadas por eventos internos o externos pudiendo ser predecibles o impredecibles, afectando a las percepciones y las acciones del agente.

Discreto vs continuo:

- En un entorno **discreto**, las variables y acciones se representan mediante un conjunto finito y discreto de valores. Esto significa que hay un número limitado de estados y acciones posibles en el entorno. Por ejemplo, en un juego de ajedrez, el tablero está dividido en un número finito de casillas y cada pieza tiene movimientos discretos y limitados que se pueden realizar. Por otro lado, en un entorno **continuo**, las variables y acciones se representan mediante un conjunto infinito o continuo de valores. Esto significa que las variables pueden tener una amplia --ñgama de valores y las acciones pueden tener un rango continuo de posibilidades. Por ejemplo, en un simulador de vuelo, las variables como la velocidad, la altitud y el ángulo de inclinación pueden tomar valores continuos, y las acciones, como ajustar la dirección o la potencia del motor, pueden tener un rango continuo de ajustes posibles.

El **objetivo** de la IA es diseñar **un programa agente** que implemente la función de agente – mapeado de percepciones a acciones. Dicho programa se ejecuta en un **dispositivo de computación** con sensores y actuadores adecuados **(arquitectura)**.

Los agentes que se verán son:

- **Agentes reactivos simples**. Son simples, de limitada inteligencia. Si el entorno no es completamente observable se encuentra en problemas.
- **Agentes reactivos basados en modelos**. Un agente puede tomar decisiones correctas en un entorno parcialmente observable si almacena el estado de la parte del mundo que no se puede percibir al tomar la decisión. Tienen la capacidad de mantener un modelo interno del entorno y utilizarlo para planificar acciones y tomar decisiones.
- **Agentes basados en función de Utilidad**. Toman decisiones evaluando las diferentes opciones en función de su utilidad esperada. Asignan valores de utilidad a los estados del entorno y seleccionan la acción que maximiza la utilidad esperada, permitiéndoles tomar decisiones racionales y adaptativas, considerando las consecuencias y beneficios esperados de sus acciones en función de las preferencias y objetivos del agente. La utilidad es una medida de rendimiento que permite comparar diferentes conjuntos de estados. Son comúnmente utilizados en problemas de optimización, como la planificación financiera, la toma de decisiones bajo incertidumbre y la planificación estratégica.
- **Agentes que aprenden**. Se pueden desarrollar en entornos desconocidos.

El agente puede representar los estados del entorno de las siguientes maneras:

- **Atómica**: los estados son indivisibles y no tienen estructura interna. Cada estado se define por un conjunto de valores de atributos que describen el estado actual del entorno.
- **Factorizado**: cada estado tiene variables o atributos. Cada variable o atributo representa una parte del estado global y puede tener diferentes valores. Este enfoque se utiliza en áreas como la planificación, la lógica proposicional, el aprendizaje automático (machine learning) y las redes bayesianas.
- **Estructurado**: el agente describe el estado del entorno en términos de objetos y las relaciones entre ellos. Cada estado se compone de una estructura de datos que incluye los objetos presentes en el entorno y sus atributos, así como las relaciones y

conexiones entre los objetos. Esta representación se utiliza en lenguajes como el lenguaje natural y la lógica de primer orden.

4-Tema

Agentes de resolución de problemas basados en búsquedas.

Fases para la resolución de problemas por parte de un agente de resolución de problemas, en un entorno:

- **Formulación de objetivos**. Los objetivos o metas ayudan a organizar el comportamiento al limitar los objetivos que el agente está tratando de lograr y, por lo tanto, las acciones que debe considerar. El objetivo se da por cumplido cuando se dan un conjunto de estados deseado.
- **Formulación del problema**. Es el proceso de decidir qué acciones y estados considerar para dar por alcanzado el objetivo. Es un modelo abstracto de la parte relevante del mundo en el cual el agente se va a mover.
- **Búsqueda**. Antes de realizar cualquier acción en el mundo real, el agente simula secuencias de acciones en su modelo, buscando hasta que encuentra la secuencia de acciones necesarias para alcanzar el objetivo.
- **Ejecución**. El agente puede ejecutar las acciones encontradas en la búsqueda y que conducen al objetivo.

Si el entorno es observable, discreto, conocido y determinista, la solución es una secuencia fija de acciones.

Un problema de búsqueda se define por varios elementos que determinan su naturaleza y características:

- **Estado inicial del agente**: Es el estado desde el cual comienza el agente en el problema de búsqueda. Representa la configuración inicial del entorno en el que se desarrolla la búsqueda.
- **Acciones del agente**: Son las posibles acciones que el agente puede tomar en cada estado del problema de búsqueda. Son aplicables en determinados estados y permiten al agente cambiar de un estado a otro.
- **Modelo de transiciones**: Es la descripción de cómo las acciones afectan el estado del entorno. Define las condiciones bajo las cuales un agente puede tomar una acción y especifica los estados resultantes de esa acción. Este modelo de transiciones permite al agente planificar acciones futuras y anticipar los posibles resultados de sus acciones.
- **Espacio de estados del problema**: Es el conjunto de todos los posibles estados alcanzables a partir del estado inicial aplicando las acciones definidas en el problema de búsqueda. Se puede representar como un grafo dirigido en el que los nodos representan los estados y las transiciones representan las acciones que el agente puede tomar para cambiar de un estado a otro. Una ruta en este grafo es una secuencia de estados conectados por una secuencia de acciones.
- **Test de objetivo**: Se trata de una condición o criterio que determina si un estado alcanzado por el agente cumple con los objetivos del problema de búsqueda, verificando si se ha alcanzado la solución deseada.
- **Coste de la solución**: Es una medida de rendimiento que evalúa la calidad de una solución encontrada por el agente en el problema de búsqueda. El costo puede estar

relacionado con el tiempo, la eficiencia, la distancia, los recursos utilizados u otros factores relevantes para el problema en cuestión.

Una vez definido el problema de búsqueda, se procede a buscar una solución a través de un **árbol de búsqueda**. El **árbol de búsqueda** es una estructura que representa las diferentes configuraciones del problema y las posibles secuencias de acciones que se pueden tomar para alcanzar una solución.

Los algoritmos de búsqueda necesitan de una estructura de datos para seguir la pista del árbol de búsqueda. Por cada nodo podemos tener un estado, el padre de ese nodo, la acción de ese nodo y el coste de la ruta a ese nodo.

Los **nodos frontera** se introducen en una **cola** (**FIFO, LIFO (es una pila), de prioridad**) para saber cuál es el siguiente en expandir. El conjunto de exploración (estados ya explorados) puede estar en una tabla hash.

La efectividad de un algoritmo de búsqueda se puede evaluar mediante diferentes medidas, que incluyen:

- **Coste de la búsqueda**: Esta medida evalúa la complejidad del algoritmo de búsqueda, es decir, la cantidad de recursos computacionales y tiempo requeridos para encontrar una solución. El coste de la búsqueda puede estar relacionado con el número de nodos generados, el número de acciones aplicadas, el tiempo de ejecución o la cantidad de memoria utilizada.
- **Coste total**: Además del coste de la búsqueda, el coste total también tiene en cuenta el coste de la solución encontrada. Este coste puede estar relacionado con el tiempo, los recursos o cualquier otra medida relevante en el contexto del problema. El coste total se calcula sumando el coste de la búsqueda al coste de la solución obtenida.

Las estrategias de búsqueda no informadas (Búsqueda ciega), se refiere a las estrategias de búsqueda en las cuales el algoritmo no tiene información adicional sobre los estados más allá de la proporcionada en la definición del problema. El algoritmo genera sucesores a partir de un nodo padre y distingue entre un estado objetivo y un estado no objetivo. Sin embargo, no se tiene conocimiento previo sobre qué tan cerca o lejos se encuentra el nodo objetivo. Por lo tanto, las estrategias de búsqueda no informada se distinguen principalmente por el **orden** en que se **expanden** los nodos, sin utilizar ninguna heurística o información adicional para guiar la búsqueda.

Búsqueda en amplitud (anchura) primero (breadth-first search):

- Se caracteriza por expandir todos los nodos de un nivel antes de pasar al siguiente nivel en el grafo. Su funcionamiento es el siguiente:
    - Se selecciona un nodo inicial del grafo y se marca como visitado. Posteriormente se generan todos los sucesores o vecinos directos del nodo actual. Estos son los nodos adyacentes que aún no han sido visitados. Se expanden los sucesores del nodo actual, añadiéndolos a una estructura de datos llamada **cola FIFO,** asegurando que los nodos se visiten en orden de cercanía al nodo inicial. Cada nodo sucesor generado se marca como visitado y se encola. Una vez que se han expandido todos los sucesores del nivel actual, se pasa al siguiente nivel, por lo que se saca el siguiente nodo de la cola y se convierte en el nodo actual. Todo lo mencionado anteriormente se repite hasta

que se haya recorrido todo el árbol y se hayan visitado todos los nodos alcanzables desde el nodo inicial. El **test de objetivo conseguido** se aplica a cada nodo cuando se genera en lugar de cuando se selecciona para la expansión, realizándose una **comprobación temprana de objetivo**.

- El algoritmo **BFS** garantiza que, si existe una solución, se encontrará la solución óptima en término del número de pasos o distancia desde el nodo inicial. Sin embargo, no es óptimo en cuanto a encontrar la solución de menor costo dado que no tiene en cuenta el costo de las acciones o el camino recorrido hasta el nodo actual, siendo óptimo únicamente si todas las acciones tienen el mismo costo o es una función no decreciente de la profundidad del nodo. Por último, cabe destacar que es costoso en tiempo y espacio, teniendo una complejidad exponencial.

Búsqueda en coste uniforme:

- A diferencia de la **búsqueda en anchura (BFS)**, el **UCS** tiene en cuenta el costo de las acciones y selecciona las rutas de menor costo en lugar de simplemente expandir nodos en orden de cercanía. Su funcionamiento es el siguiente:
    - Se selecciona un nodo inicial del grafo y se marca como visitado. En este algoritmo se emplea una **cola de prioridad** para almacenar los nodos pendientes de ser visitados, ordenándolos en función del costo acumulado de llegar a ese nodo. Posteriormente, se extrae el nodo de menor costo de la cola de prioridad y se considera como el nodo actual, verificando si es el nodo objetivo. Si no lo es, se generan todos los sucesores del nodo actual, siendo los nodos adyacentes que aún no han sido visitados. Se calcula el costo acumulado para llegar a cada sucesor sumando el costo de la acción que lleva al sucesor al costo acumulado del nodo actual. Si un sucesor ya está en la cola de prioridad, se compara el nuevo costo acumulado con el costo acumulado previo. Si el nuevo costo es menor, se actualiza el costo acumulado en la cola de prioridad. Si el sucesor no está en la cola de prioridad, se agrega con su costo acumulado correspondiente. Si se encuentra el nodo objetivo, se reconstruye la ruta de menor costo desde el nodo inicial hasta el nodo objetivo utilizando la información almacenada en los nodos.
- El algoritmo **Uniform-Cost Search** garantiza encontrar una ruta de menor costo si existe una solución en el grafo. Al mantener una cola de prioridad ordenada por el costo acumulado, el **UCS** prioriza la exploración de las rutas de menor costo, lo que lo hace óptimo en términos de encontrar la solución de menor costo.

Búsqueda en profundidad primero (Depth-first search):

- A diferencia del algoritmo **Breadth-First Search (BFS)** que explora los nodos vecinos antes de avanzar a los nodos más distantes, el **DFS** se adentra en la estructura del grafo tanto como sea posible antes de retroceder. Su funcionamiento es el siguiente:
    - Se selecciona un nodo inicial del grafo y se marca como visitado. Se selecciona un nodo vecino no visitado del nodo actual y se avanza a él. Se marca el nodo actual como visitado para evitar volver a visitarlo en futuras iteraciones. Se vuelve a seleccionar un vecino no visitado y se repite lo anterior continuando así la exploración en profundidad. Si se alcanza un nodo que no tiene nodos vecinos no explorados, se retrocede al nodo anterior no visitado y se continúa la exploración a partir de ahí. Repitiéndose de nuevo todo lo anterior hasta que no haya más nodos por explorar.

- El **DFS** es eficiente en cuanto a uso de espacio, ya que solo necesita mantener una pila para almacenar los nodos pendientes de exploración. Sin embargo, no garantiza la solución óptima y puede quedar atrapado en ciclos si no se toman precauciones adecuadas.

Búsqueda bidireccional:

- Se inician dos búsquedas simultáneas, una desde el nodo inicial y otra desde el nodo objetivo. En cada iteración se expanden los nodos en ambas direcciones, verificando si los nodos generados en ambas búsquedas se intersecan o se encuentran en el mismo estado. Si se encuentra una intersección, se ha encontrado una solución y se termina el algoritmo.

Las estrategias de búsqueda informadas (heurística) utilizan conocimientos específicos del problema más allá de la definición básica del problema. El enfoque general utilizado en las estrategias de búsqueda informada es conocido como **"búsqueda del mejor primero" o "best-first search"**. En este enfoque, se utiliza una función de evaluación **f(h)** que estima el costo o la calidad de un nodo en términos de su proximidad a una solución deseada. A diferencia del algoritmo **Uniform-Cost Search** que utiliza la función **g** para evaluar el costo acumulado hasta un nodo, en el enfoque del **mejor primero** se utiliza la función **f**. La mayoría de los algoritmos de búsqueda del mejor primero incluyen como componente de la función **f** una **función heurística h(n)**, donde **h(n)** proporciona una estimación del costo o distancia desde el estado en el nodo n hasta un estado objetivo.

Búsqueda codiciosa del mejor primero (Greedy best-first search):

- Es una estrategia de búsqueda informada que se basa en la heurística para guiar la exploración hacia los nodos que parecen ser más prometedores según la estimación de la función heurística. Su funcionamiento es el siguiente:
    - Se elige el nodo inicial del problema y se aplica la **función heurística h(n)** a cada nodo sucesor del nodo actual para obtener una estimación de su proximidad al objetivo. Después se selecciona el nodo sucesor que tenga la estimación heurística más baja, es decir, el que se considera más cercano al objetivo según la función heurística. Se repite todo lo anterior hasta que se alcance el objetivo o no haya más nodos por explorar.
- El algoritmo no garantiza encontrar la solución óptima, ya que puede quedar atrapado en un mínimo local o tomar decisiones basadas únicamente en información heurística sin considerar el costo acumulado hasta el momento.

Búsqueda A*:

- Combina la exploración de nodos con la información de la función heurística para encontrar una solución óptima en un árbol. Su funcionamiento es el siguiente:
    - Se selecciona el nodo inicial y se establece su valor de costo acumulado en cero. Después se calculan dos valores para cada nodo: **g(n) y h(n).** El valor **g(n)** representa el costo acumulado desde el nodo inicial hasta el nodo n, y el valor **h(n)** es una estimación del costo restante desde el nodo n hasta el objetivo, según la función heurística. En cada iteración, se selecciona el nodo con el valor **f(n)** más bajo, donde **f(n) = g(n) + h(n).** Este nodo se expande y se generan sus nodos sucesores. Para cada sucesor generado, se actualizan los valores de **g(n) y h(n)** y se establece el nodo actual como su padre. En cada

iteración, se verifica si el nodo expandido es el objetivo. Si es así, se ha encontrado una solución óptima y se termina el algoritmo.

- El algoritmo **A*** requiere el uso de una heurística **admisible y consistente** para garantizar su correcto funcionamiento. Una heurística es considerada **admisible** si nunca sobreestima el costo real desde un nodo hasta el objetivo, y es **consistente**, si para un nodo n y todos sus sucesores n' generados por cualquier acción a, el costo estimado de llegar al objetivo desde n nunca es mayor que el costo estimado de llegar al objetivo desde n' más el costo real de realizar la acción a. La consistencia garantiza que la estimación heurística no "contradiga" el costo real de las acciones tomadas, lo que significa que la heurística proporciona una estimación coherente y confiable del costo restante.

5-Tema

Los **agentes basados en conocimiento** utilizan un proceso de razonamiento sobre una representación interna del conocimiento para decidir sobre los próximos pasos a dar. Representan el estado del entorno de manera factorizado. Pueden aceptar nuevas tareas en forma de metas descritas explícitamente, tienen la capacidad de adquirir nuevos conocimientos y aprender de su entorno y tienen la capacidad de actualizar su base de conocimientos para reflejar los cambios en su entorno, permitiéndoles seguir siendo efectivos incluso en entornos dinámicos y en constante cambio.

Estos agentes se fundamentan en una **Base de Conocimiento (KB)** compuesta por sentencias escritas en un lenguaje de representación del conocimiento, que representan aserciones sobre el mundo en el que opera el agente. Las sentencias en la **KB** pueden ser **axiomas**, lo que significa que no se derivan de otras sentencias, sino que se consideran como verdades fundamentales.

El agente también puede hacer preguntas o consultas a la **KB** utilizando la operación **ASK**. Esto implica buscar información específica o verificar la veracidad de una afirmación en base a lo que se conoce en la **KB**. Es importante destacar que las respuestas proporcionadas por la operación **ASK** deben ser necesariamente consecuencia lógica de lo que se ha proporcionado previamente al sistema a través de la operación **TELL**. El agente no puede inventar información que no esté respaldada por la **KB**. Tanto la operación **TELL** como la operación **ASK** pueden involucrar inferencias, es decir, la capacidad del agente para derivar nuevas sentencias a partir de las que ya están presentes en la **KB**. Estas inferencias permiten al agente realizar un razonamiento lógico y ampliar su base de conocimientos. Es importante tener en cuenta que un agente basado en conocimiento puede tener un conocimiento previo del mundo, denominado **conocimiento previo** (**Background knowledge**).

En el contexto de la **lógica**, dos conceptos fundamentales son:

- La **sintaxis** se refiere a las reglas que gobiernan la formación y la estructura de las sentencias o axiomas en un lenguaje lógico. Estas reglas determinan cómo se combinan los símbolos y los operadores lógicos para construir expresiones válidas. La sintaxis define la gramática del lenguaje y establece las pautas para la correcta construcción de las sentencias. Por ejemplo, en la lógica proposicional, se pueden tener sentencias formadas por proposiciones atómicas y conectivas lógicas, como la conjunción, la disyunción y la negación, siguiendo ciertas reglas de combinación.

- La **semántica** se refiere al significado o interpretación de las sentencias en relación con un modelo o posible mundo. Establece cómo se asigna un valor de verdad (verdadero o falso) a una sentencia en función de las condiciones que deben cumplirse en el modelo. Un **modelo** es una interpretación de las proposiciones y los operadores lógicos de acuerdo con un conjunto de reglas y restricciones. Si una sentencia es verdadera en un modelo dado, decimos que el modelo **satisface** la sentencia.

En lógica, la notación "**α ⊨ β**" se utiliza para indicar que la sentencia α **implica lógicamente** la sentencia β. Esto significa que, si α es verdadera, entonces necesariamente β también debe ser verdadera.

En términos semánticos, podemos decir que α implica β **si y solo si** los modelos en los que α es verdadera están contenidos en los modelos en los que β es verdadera. Es decir, si todos los posibles mundos en los que α es verdadera **también** hacen que β sea verdadera, entonces α implica β.

Imaginemos que el conjunto de todas las consecuencias derivadas de la base de conocimiento (**KB**) es como un gran pajar, y la sentencia α que queremos probar es como una aguja perdida en ese pajar. Una implicación α ⊨ β es como encontrar esa aguja en el pajar, y la inferencia KB ⊢ α es el proceso de buscar y encontrar esa aguja.

Un algoritmo de inferencia se utiliza para derivar sentencias a partir de la **KB**, es decir, buscar en el pajar en busca de la aguja α. Si el algoritmo de inferencia es **sólido** (sound), significa que solo deriva sentencias que son verdaderas, preservando la verdad de la KB. Esto asegura que si el algoritmo de inferencia encuentra la implicación α ⊨ β, podemos confiar en que realmente se cumple.

Por otro lado, si el algoritmo de inferencia es **completo**, significa que puede derivar cualquier sentencia que esté implícita en la KB. Esto implica que, si la aguja α está realmente en el pajar, el algoritmo de inferencia la encontrará eventualmente.

El proceso de **inferencia lógica** implica examinar todos los posibles modelos basados en la información que tenemos, verificar si satisfacen nuestra base de conocimiento y determinar si nuestra base de conocimiento implica una conclusión específica. Es una forma de razonamiento lógico que nos ayuda a sacar conclusiones basadas en la información disponible.

Decir que "**a es formalmente deducible de KB**" significa que es posible derivar o probar lógicamente la afirmación o sentencia "a" utilizando el conjunto de conocimientos o la base de conocimientos "KB" como premisas. En otras palabras, a se puede obtener como una conclusión lógica utilizando las reglas de inferencia y los axiomas que están definidos en KB.

La deducción formal implica seguir una secuencia de pasos lógicos y aplicar reglas de inferencia válidas para llegar a la conclusión deseada. Si se puede demostrar que “a” es formalmente deducible de KB, significa que existe una prueba lógica válida que muestra que “a” se sigue necesariamente de las premisas en KB.

El **model checking** es una técnica utilizada en el contexto de la lógica proposicional para verificar la validez de propiedades en sistemas formales. Consiste en verificar exhaustivamente si un modelo, que representa un sistema en particular, cumple ciertas propiedades especificadas en lógica proposicional. Implica generar sistemáticamente todos los posibles

estados del sistema y verificar si las propiedades especificadas se cumplen en cada estado (tablas de verdad).

La equivalencia lógica: dos axiomas α y β son lógicamente equivalentes si son verdaderos en el mismo conjunto de modelos

Theorem proving en el algoritmo 1 en resolución por búsqueda (aplicable a cualquier algoritmo de resolución por búsqueda del tema anterior):

- **Estado Inicial**: Es la base de conocimientos inicial que tenemos, que contiene las premisas y reglas de inferencia que conocemos.
- **Acciones**: El conjunto de acciones consiste en todas las reglas de inferencia que podemos aplicar a las oraciones en nuestra base de conocimientos.
- **Resultado**: El resultado de una acción es agregar una nueva oración a nuestra base de conocimientos, que es el axioma que se encuentra en la mitad inferior de la regla de inferencia.
- **Objetivo**: Es un estado que contiene el axioma que estamos tratando de demostrar. Queremos encontrar una secuencia de acciones que nos lleve desde el estado inicial hasta el objetivo, demostrando así el teorema que estamos buscando.

El proceso de **theorem proving** mediante **búsqueda** es más eficiente que el **model-checking**, ya que puede ignorar proposiciones irrelevantes y enfocarse únicamente en las oraciones relevantes para demostrar el teorema. Además, este enfoque se basa en la propiedad de que un sistema lógico es **monotónico**, lo que significa que agregar más conocimiento nunca invalidará una implicación previamente demostrada.

Theorem proving en el algoritmo 2 en **prueba por resolución**, empleado para demostrar teoremas y deducir conclusiones:

- La **Resolución** es una única regla de inferencia que se aplica a cláusulas, que son disyunciones literales. Produce un algoritmo de inferencia **completo** cuando se combina con cualquier algoritmo de búsqueda completo. Nos permite combinar cláusulas para obtener nuevas cláusulas.
- Para utilizar la **resolución** de manera efectiva, es importante expresar todo el conocimiento en forma de cláusulas. Cada sentencia de lógica proposicional se puede convertir en una conjunción de cláusulas equivalentes. Esto se conoce como la **Conjuntive Normal Form (CNF).**
- El objetivo de la prueba por resolución es demostrar que una afirmación α se sigue lógicamente de una base de conocimientos (KB). Para hacer esto, probamos que la conjunción de la base de conocimientos y la negación de α (KB ∧ ¬α) es **insatisfacible**, lo que significa que no existe un modelo que haga que **todas** las cláusulas sean verdaderas al mismo tiempo.
- Hay dos posibles resultados al aplicar la resolución:
    - No se pueden agregar nuevas cláusulas, lo que significa que **KB ∧ ¬α** es **insatisfacible**. Esto implica que α se sigue lógicamente de la base de conocimientos (KB) y se denota como **KB ⊨ α**.
    - Se obtiene la cláusula vacía (∅), lo que significa que no se puede satisfacer KB ∧ ¬α. En este caso, se concluye que KB no implica α y se denota como KB ⊭ α.

La **lógica proposicional** es un lenguaje formal que nos permite representar proposiciones o afirmaciones sobre el mundo utilizando variables, conectores lógicos y operadores.

Los agentes basados en lógica proposicional la utilizan para representar el conocimiento y razonar sobre él. Tienen una base de conocimientos formada por sentencias en lógica proposicional, las cuales representan afirmaciones sobre el estado del mundo en el que operan los agentes. Utilizan reglas de inferencia y algoritmos de razonamiento para procesar la información contenida en su base de conocimientos y tomar decisiones y pueden realizar deducciones lógicas, hacer inferencias y responder a consultas sobre el estado del mundo en base a su conocimiento.

La lógica proposicional, aunque útil para representar conocimiento en entornos simples, puede resultar limitada cuando se trata de representar conocimiento más complejo de manera concisa. Es por eso por lo que se utiliza **la lógica de primer orden**, también conocida como lógica de predicados, que es más expresiva y flexible.

La lógica de primer orden es una extensión de la **lógica proposicional** que combina los aspectos de la lógica proposicional con elementos del lenguaje natural, como objetos y relaciones, para proporcionar una representación más rica y expresiva del conocimiento. En **la lógica de primer orden,** se introducen variables para representar objetos y se definen **predicados** para representar relaciones entre estos objetos. Estos **predicados** pueden tener argumentos que corresponden a las variables y representan las diferentes posiciones en las que se pueden encontrar los objetos en una relación. Además de las variables y predicados, la lógica de primer orden también introduce **cuantificadores**, como **el cuantificador universal (∀)** y **el cuantificador existencial (∃),** que permiten hacer afirmaciones sobre todos los objetos o al menos un objeto que cumpla ciertas condiciones; y permite el uso de funciones, que asignan valores a variables y se utilizan para representar operaciones y cálculos sobre los objetos. Estas funciones pueden tener argumentos y retornar un valor.

En **la lógica de primer orden**, cada **modelo** se utiliza para establecer una correspondencia entre el vocabulario utilizado en las sentencias lógicas y los elementos del mundo posible. Esto permite determinar la verdad o falsedad de cualquier oración dentro de ese modelo.

El **dominio** de un modelo se refiere al conjunto de objetos o elementos de dominio que contiene. Es importante destacar que el dominio no puede estar vacío, debe contener al menos un objeto. Lo que importa en un modelo no es qué objetos específicos están incluidos, sino cuántos objetos hay en total.

Los objetos en un modelo pueden estar relacionados entre sí de diferentes maneras. Por ejemplo, puede haber **una relación binaria de hermandad, una relación binaria más general y relaciones unarias** (representan propiedades o características de un objeto en particular).

La **lógica de primer orden** permite el uso de **funciones**. Una **función** es una relación que, dado un objeto, se relaciona con otro objeto específico. Por ejemplo, se puede tener una función que, dada una persona, se relacione con su ciudad de residencia. Cabe destacar que las funciones en la lógica de primer orden deben ser **totales**, lo que significa que debe haber un valor asignado a cada posible combinación de objetos de entrada.

Los elementos sintácticos básicos son los **símbolos** que representan objetos, relaciones y funciones. Estos símbolos se utilizan para construir oraciones lógicas y expresar proposiciones sobre el mundo.

Los símbolos constantes se utilizan para representar objetos individuales, como nombres de personas o lugares.

Los símbolos de predicado se utilizan para representar relaciones entre objetos.

Los símbolos de funciones se utilizan para representar operaciones que toman uno o más argumentos y devuelven un resultado.

Cada modelo en la **lógica de primer orden** proporciona la información necesaria para determinar si una sentencia dada es verdadera o falsa. Esto se logra mediante la interpretación (I) del modelo, que especifica a qué objetos, relaciones y funciones se refieren los símbolos constantes, de predicado y de función.

Un **modelo en la lógica de primer orden** consiste en un conjunto de objetos y una interpretación que asigna los símbolos constantes a objetos específicos, los símbolos de predicado a relaciones en esos objetos, y los símbolos de función a funciones que operan sobre esos objetos.

Un **término** es una expresión lógica que se refiere a un objeto. Los términos pueden ser simples, como los símbolos constantes que representan objetos específicos, o pueden ser complejos, como las funciones que toman argumentos y devuelven un resultado.

Una **sentencia atómica** expresa hechos sobre el mundo. Está formada por un símbolo de predicado seguido opcionalmente de una lista de términos entre paréntesis. **Una oración atómica** es verdadera en un modelo si la relación a la que hace referencia el símbolo de predicado se cumple entre los objetos a los que se refieren los argumentos.

Se pueden construir **sentencias complejas** utilizando conectores lógicos, al igual que en la lógica proposicional.

Las sentencias se añaden a la base de conocimiento mediante **aserciones** usando **TELL**.

Preguntamos a la base de conocimiento **queries** o **goals** mediante aserciones usando **ASK**.

Los **axiomas** son enunciados asumidos como verdaderos sin necesidad de demostración, mientras que los **teoremas** son enunciados que se derivan lógicamente de los axiomas y otras proposiciones demostradas. Los **axiomas** establecen las reglas básicas del sistema lógico o matemático, mientras que los **teoremas** amplían el conocimiento al demostrar nuevas proposiciones basadas en esas reglas.

6-Tema

Los agentes necesitan manejar la **incertidumbre** debido a varias razones:

- Solo puede tener acceso a información parcial sobre su entorno. Debe basar sus acciones en la información disponible, lo que puede llevar a cierta incertidumbre sobre la situación actual.
- Sus acciones pueden no tener resultados predecibles o deterministas.
- Pueden no tener conocimiento completo sobre su propio estado.

Para ser capaces de manejar la incertidumbre en su toma de decisiones utilizan técnicas y modelos que permitan razonar y actuar en esas situaciones:

- Pueden asignar **probabilidades** a diferentes estados o resultados posibles. Pueden utilizar técnicas como **la teoría de la probabilidad** para evaluar las acciones más prometedoras en función de su probabilidad de éxito.
- Pueden utilizar modelos formales para representar y razonar sobre la incertidumbre. Por ejemplo, pueden utilizar **redes bayesianas** o **procesos de decisión de Markov** para modelar y planificar en entornos inciertos.
- Pueden establecer metas claras y utilizar **algoritmos de planificación** para generar secuencias de acciones que maximicen la probabilidad de lograr esas metas, incluso en entornos inciertos.

La **Teoría de la Utilidad** es un concepto fundamental en la toma de decisiones de los agentes inteligentes. Según esta teoría, cada estado en el que puede encontrarse un agente tiene asociado un grado de utilidad, que representa la preferencia o valor que el agente asigna a ese estado en particular. El agente tomará decisiones que le permitan alcanzar los estados que considera más deseables.

La **Teoría de la Utilidad** se combina con la **teoría de la probabilidad** en la teoría general de decisiones racionales, conocida como **Teoría de la Decisión**. Esta teoría establece que la toma de decisiones racionales implica considerar tanto las probabilidades de que ocurran ciertos resultados como las utilidades asociadas a esos resultados. Por lo tanto, la **Teoría de la Decisión** abarca tanto la incertidumbre como las preferencias del agente.

Teoría de la Decisión = Teoría de Probabilidad + Teoría de la Utilidad

El **espacio muestral** es el conjunto de todos los posibles resultados o mundos posibles en un experimento o situación.

Los eventos son **mutuamente exclusivos** cuando no pueden ocurrir al mismo tiempo. En otras palabras, si se cumple uno de los eventos, los otros eventos **mutuamente exclusivos** no pueden ocurrir.

Los eventos son **exhaustivos** cuando, en conjunto, cubren todos los posibles resultados en el espacio muestral. Esto significa que al menos uno de los eventos debe ocurrir.

Un **modelo de probabilidad completamente especificado** asigna una probabilidad numérica ($P(\omega)$) a cada elemento del espacio muestral. Esto significa que se asigna una probabilidad a cada posible resultado o mundo posible.

Un **evento** es un conjunto de mundos posibles sobre los que se realiza una aserción probabilística o una consulta. En la notación de probabilidad clásica, un evento se describe como una proposición en lenguaje formal.

Las **proposiciones** que describen conjuntos de mundos posibles (**eventos**) están escritas en una notación que combina elementos de lógica proposicional y notación de satisfacción de restricciones - representación factorizada.

Las **distribuciones de probabilidad** son una forma de describir cómo se distribuyen las probabilidades de los diferentes resultados en un conjunto de posibles eventos. Proporcionan información sobre las probabilidades asociadas a cada posible resultado o evento en un espacio muestral. Son importantes en la estadística y la teoría de la probabilidad, ya que permiten modelar y analizar una amplia gama de fenómenos y eventos aleatorios.

La función de densidad de probabilidad (PDF) es una herramienta utilizada en la teoría de la probabilidad y la estadística para describir la distribución de probabilidad de una variable aleatoria continua. Proporciona una descripción matemática de cómo se distribuyen las probabilidades a lo largo del rango de valores posibles de la variable aleatoria continua. No indica la probabilidad de obtener un valor específico, sino la probabilidad relativa de que la variable aleatoria caiga dentro de un intervalo dado.

Una **distribución de probabilidad conjunta completa** proporciona información detallada sobre la relación entre todas las variables aleatorias en un experimento o situación.

La probabilidad marginal es la tarea común de extraer la distribución sobre algún subconjunto de variables o una variable. Se refiere a la probabilidad de un evento o variable aleatoria específica sin tener en cuenta las demás variables aleatorias.

La inferencia probabilística es un proceso que nos permite obtener conclusiones o hacer predicciones basadas en la información proporcionada por las distribuciones de probabilidad. En particular, cuando se trabaja con distribuciones conjuntas completas, podemos realizar inferencias más precisas y detalladas sobre las variables aleatorias involucradas. También nos permite realizar predicciones sobre eventos futuros o valores desconocidos. Utilizando la distribución conjunta completa, podemos calcular la probabilidad de diferentes escenarios y evaluar la incertidumbre asociada a estas predicciones.

La Regla de Bayes es un concepto fundamental en la teoría de la probabilidad y estadística que nos permite actualizar nuestras creencias o probabilidades iniciales en función de nueva evidencia o información observada.

La fórmula de **la Regla de Bayes** se expresa de la siguiente manera:

$$P(A|B) = (P(B|A) * P(A)) / P(B)$$

Donde:

- **P(A|B)** es la probabilidad condicional de que el evento A ocurra dado que el evento B ha ocurrido.
- **P(B|A)** es la probabilidad condicional de que el evento B ocurra dado que el evento A ha ocurrido.
- **P(A) y P(B)** son las probabilidades marginales de los eventos A y B, respectivamente.

Algún uso común de **la Regla de Bayes** es en **los diagnósticos médicos**, se utiliza para evaluar la probabilidad de que un paciente tenga una enfermedad en función de los síntomas observados y la prevalencia de la enfermedad en la población, o en la **predicción meteorológica**, para ajustar las probabilidades de los eventos climáticos en función de las observaciones y los datos históricos.

Una **red bayesiana** es un modelo gráfico probabilístico que combina teoría de probabilidad y teoría de grafos para representar las relaciones causales y dependencias condicionales entre variables.

La representación de **incertidumbre** en una **red bayesiana** se logra mediante el uso de nodos que representan las variables y arcos dirigidos que indican las relaciones de dependencia entre ellas. Cada nodo en la red representa una variable y está asociado con una distribución de probabilidad condicional que describe cómo la variable depende de sus padres en el grafo. La

ventaja de utilizar una **red bayesiana** es que se puede representar la distribución de probabilidad conjunta completa de manera compacta y eficiente

Método para construir redes bayesianas:

1. Identifica las variables relevantes en el dominio que estás modelando. Estas variables representarán los nodos de la red bayesiana.
2. Determina las relaciones de dependencia entre las variables. Cómo se conectan los nodos entre sí mediante arcos dirigidos.
3. Para cada nodo en la red, asigna una DPC que describe cómo la variable depende de sus padres en la red. Estas DPCs especifican las probabilidades condicionales asociadas con cada combinación de valores de los padres (Regla de la Cadena).
4. Asegúrate de que las DPCs sean consistentes y no generen ciclos en la red. Puedes realizar pruebas de consistencia para verificar que las DPCs cumplan con las reglas de probabilidad.
5. Utiliza la red bayesiana para realizar inferencias probabilísticas. Puedes calcular probabilidades condicionales, actualizar creencias en función de nueva evidencia o realizar predicciones.

La relación de independencia condicional en una red bayesiana se refiere a la independencia de un nodo con respecto a otros nodos en la red.

Un nodo en una red bayesiana es **condicionalmente independiente** de sus **no descendientes**, es decir, de otros nodos que no son sus hijos ni están conectados directamente a él, dado el conocimiento de sus padres. La probabilidad de que ocurra un evento en el nodo en consideración no se ve afectada por la presencia o ausencia de eventos en los no descendientes, una vez que se conocen los valores de los padres.

El **manto de Markov** de un nodo en una red bayesiana se refiere al conjunto de nodos que están directamente conectados al nodo en cuestión o son descendientes de sus padres. Se dice que un nodo está relacionado con los nodos dentro de su **manto de Markov**, lo que implica que la información necesaria para determinar la probabilidad de ese nodo se encuentra dentro de su **manto de Markov**, y no se requiere información adicional de nodos más distantes.

La inferencia exacta en redes bayesianas consiste en determinar la probabilidad de un evento dado el conocimiento de valores observados en otros nodos de la red. Se basa en la regla de Bayes y utiliza las distribuciones de probabilidad conjunta y condicional definidas en la red bayesiana. Existen varios algoritmos para llevar a cabo la **inferencia exacta en redes bayesianas**:

1. El **algoritmo de eliminación de variables** aprovecha la estructura de la red bayesiana para eliminar variables de manera progresiva y calcular las probabilidades condicionales de interés de manera eficiente. Aprovecha el **principio de independencia condicional en redes bayesianas** para eliminar las variables que no afectan directamente a la variable de consulta. Al eliminar estas variables, se reduce el tamaño del problema y se simplifican los cálculos necesarios para obtener las probabilidades deseadas.

2. **La inferencia por enumeración** se basa en la idea de enumerar todos los posibles estados de los nodos en la red y calcular las probabilidades condicionales a partir de las distribuciones de probabilidad definidas en la red.
   El **proceso de inferencia por enumeración** consiste en seguir los siguientes pasos:
   1. Seleccionar el evento de interés y establecer los valores observados en los nodos relevantes de la red.
   2. Enumerar todos los posibles estados de los nodos en la red, es decir, todas las combinaciones de valores que pueden tomar los nodos.
   3. Calcular la probabilidad de cada estado enumerado, multiplicando las probabilidades de los nodos de acuerdo con las distribuciones de probabilidad condicional definidas en la red.
   4. Sumar las probabilidades de los estados que son consistentes con el evento de interés. Esto proporciona la probabilidad marginal del evento.

/* **Describe el comportamiento del algoritmo de Best-First-Search y explica qué cambios serían necesarios introducir en él para que se comportara como Breath-First search y como un Depth-First search.** Para hacer que el algoritmo Best-First Search se comporte como Breadth-First Search (BFS), se requiere modificar la estrategia de expansión de los nodos. El algoritmo BFS expande los nodos en el orden en que fueron descubiertos, de modo que todos los nodos en el mismo nivel se exploran antes de pasar al siguiente nivel. Para lograr esto en Best-First Search, se debe modificar la función de evaluación que guía la expansión de los nodos para que solo tenga en cuenta el orden de descubrimiento de los nodos, sin considerar ningún otro factor como la heurística o el costo del camino.

En cuanto a hacer que el algoritmo Best-First Search se comporte como Depth-First Search (DFS), nuevamente es necesario modificar la estrategia de expansión de los nodos. DFS explora los nodos hasta que alcanza un nodo hoja antes de retroceder y explorar otras ramas. Para lograr esto en Best-First Search, se debe modificar la función de evaluación de manera que dé prioridad a los nodos más profundos en lugar de los nodos más prometedores según la heurística o el costo del camino. Esto permitirá que el algoritmo explore en profundidad antes de expandir los nodos más cercanos a la solución.

En resumen, para hacer que Best-First Search se comporte como Breadth-First Search se debe modificar la función de evaluación para tener en cuenta el orden de descubrimiento de los nodos, y para hacer que se comporte como Depth-First Search se debe modificar la función de evaluación para dar prioridad a los nodos más profundos en lugar de los más prometedores.

¿Cómo podemos especificar de forma completa un entorno en el que se desenvuelve un Agente racional? Describe el significado de las siglas PEAS.

¿Qué es una base de conocimiento? ¿De qué manera un agente lógico puede utilizarla para extraer conclusiones? ¿Qué tipo de operaciones se permiten en una base de conocimiento?

Describe brevemente qué caracteriza a la lógica proposicional y en qué se diferencia de la lógica de primer orden. ¿Es suficientemente expresiva la lógica proposicional? ¿Por qué?

¿Qué es el teorema de Bayes? ¿Puedes nombrar alguna situación en la que pueda resultarnos de utilidad? Pon un ejemplo y razónalo.

¿Qué ventaja nos aporta la independencia de variables aleatorias respecto a la probabilidad condicionada? La independencia de variables aleatorias simplifica los cálculos (el cálculo de la probabilidad conjunta y la probabilidad condicionada se simplifica), reduce la dependencia de información (la información sobre una variable no proporciona ninguna información adicional sobre la otra variable), proporciona flexibilidad en el modelado y permite una interpretación intuitiva de las relaciones entre variables**. ¿Podemos estar siempre convencidos de la independencia de variables aleatorias?** No podemos estar siempre convencidos de la independencia de variables aleatorias sin una justificación o evidencia adecuada. La independencia de variables aleatorias es una propiedad que debe ser evaluada y respaldada por análisis estadísticos o conocimiento del dominio. **¿Qué relación tiene la independencia de variables aleatorias con las redes bayesianas?** La independencia de variables aleatorias en el contexto de las redes bayesianas implica que ciertas variables son independientes de otras dadas ciertas variables intermedias. Esto se refleja en la estructura del grafo de la red bayesiana, donde los nodos representan variables aleatorias y las aristas representan las dependencias condicionales. La independencia de variables aleatorias en una red bayesiana es utilizada para simplificar y reducir el número de cálculos necesarios en la inferencia

probabilística. Si dos variables aleatorias son independientes dadas ciertas variables intermedias, podemos evitar realizar cálculos redundantes y reducir la complejidad computacional de la inferencia.

¿Qué significa la expresión b es consecuencia lógica de a? ¿Cómo comprobaríamos que b es una consecuencia lógica de a utilizando tablas de verdad?

Construir una tabla de verdad que contenga todas las variables proposicionales presentes en las fórmulas a y b, así como las fórmulas a y b mismas.

Asignar valores de verdad a todas las variables proposicionales en la tabla de verdad, considerando todas las combinaciones posibles de valores de verdad.

Calcular los valores de verdad para las fórmulas a y b en cada fila de la tabla de verdad, utilizando las reglas de la lógica proposicional.

Verificar si en todas las filas donde la fórmula a es verdadera, la fórmula b también es verdadera. Si esto se cumple, podemos concluir que b es una consecuencia lógica de a.

¿Define el concepto de satisfacibilidad? La **satisfacibilidad** es un concepto utilizado en lógica y teoría de la computación para determinar si una fórmula lógica es verdadera o tiene una interpretación que la hace verdadera. Se refiere a la capacidad de encontrar una asignación de valores de verdad a las variables de una fórmula lógica que haga que la fórmula sea verdadera. En lógica proposicional, una fórmula es satisfacible si existe al menos una asignación de valores de verdad a las variables que hace que la fórmula sea verdadera. En lógica de primer orden, la **satisfacibilidad** se refiere a la capacidad de encontrar una interpretación o modelo en el cual todas las fórmulas de un conjunto dado sean verdaderas. **¿De qué manera sencilla podríamos comprobar la satisfacibiliad de una sentencia respecto a otra?** Una manera sencilla de comprobar la **satisfacibilidad** de una sentencia respecto a otra es mediante la construcción de una tabla de verdad. Se enumeran todas las posibles combinaciones de valores de verdad para las variables involucradas en ambas sentencias. Luego, se evalúa cada sentencia en cada una de estas combinaciones de valores y se compara si ambas sentencias son verdaderas o no.

Si en todas las combinaciones de valores de verdad ambas sentencias son verdaderas, entonces se puede concluir que la primera sentencia es satisfacible respecto a la segunda. De lo contrario, si existe al menos una combinación en la cual la primera sentencia es falsa mientras que la segunda es verdadera, entonces la primera sentencia no es satisfacible respecto a la segunda.

¿Para qué se utilizan las CNF? Defínelas brevemente. Las CNF (Conjunctive Normal Form) se utilizan principalmente en lógica proposicional y en programación lógica para representar y analizar problemas y expresiones lógicas de una manera estandarizada.

Algunas de las aplicaciones y usos de las CNF son:

Resolución de problemas de satisfacibilidad: Las CNF son una forma estándar para representar problemas de satisfacibilidad booleana (SAT) y se utilizan en algoritmos de resolución de SAT para determinar si existe una asignación de valores de verdad que satisfaga una fórmula lógica.

Razonamiento automático: Las CNF son utilizadas en sistemas de razonamiento automático, como los demostradores de teoremas, para representar y analizar conocimiento lógico y realizar inferencias sobre él.

Optimización de circuitos booleanos: Las CNF se utilizan en el diseño y optimización de circuitos lógicos, donde se busca encontrar la forma más eficiente y compacta de implementar una función booleana.

Representación de reglas y restricciones: Las CNF se utilizan en sistemas basados en reglas y en programación lógica para representar reglas y restricciones que deben cumplirse en un dominio específico.

*/

La **resolución de problemas de satisfacibilidad**, también conocidos como **SAT (Satisfiability**), se refiere a la tarea de determinar si una fórmula lógica puede ser satisfecha, es decir, si existe una asignación de valores de verdad a las variables proposicionales que haga que la fórmula sea verdadera.

El problema **SAT** es de gran importancia en el campo de la informática y la inteligencia artificial, ya que muchos problemas pueden ser formulados en términos de **satisfacibilidad** de fórmulas lógicas. Por ejemplo, la planificación de tareas, la verificación de circuitos electrónicos y la programación de horarios son ejemplos de problemas que se pueden reducir al problema SAT.

La resolución de problemas SAT implica encontrar una asignación de valores de verdad a las variables que haga que la fórmula sea verdadera, o demostrar que no existe tal asignación, en cuyo caso se dice que la fórmula es insatisfacible.

Existen diferentes enfoques y algoritmos para resolver problemas SAT. Uno de los algoritmos más utilizados es el Método de Resolución, que combina inferencia lógica y búsqueda en un espacio de posibles asignaciones de valores de verdad.

El theorem proving (demostración de teoremas) y el model checking son dos enfoques diferentes para abordar problemas de verificación en sistemas formales. A continuación, se presentan algunas diferencias clave entre ellos:

Objetivo:

Theorem proving: El objetivo es demostrar si una afirmación o teorema es verdadero a partir de un conjunto de axiomas y reglas de inferencia.

Model checking: El objetivo es verificar si un modelo de sistema cumple ciertas propiedades o especificaciones.

Representación:

Theorem proving: Se basa en la manipulación de símbolos y reglas de inferencia lógica para derivar conclusiones lógicas.

Model checking: Utiliza estructuras de modelos finitos, como grafos o árboles de estados, para representar el comportamiento del sistema y verificar propiedades en cada estado.

Complejidad:

Theorem proving: Es generalmente más complejo y requiere técnicas avanzadas de lógica y matemáticas para demostrar teoremas.

Model checking: Es más eficiente y escalable, especialmente para sistemas grandes y complejos, ya que utiliza técnicas de exploración sistemática de estados.

Alcance:

Theorem proving: Puede abordar una amplia gama de problemas de demostración de teoremas y razonamiento lógico, incluyendo lógica proposicional y de primer orden.

Model checking: Se utiliza principalmente para verificar propiedades específicas y finitas en sistemas concurrentes o sistemas basados en eventos, como protocolos de comunicación o circuitos digitales.

En resumen, el theorem proving se centra en demostrar la validez de afirmaciones lógicas utilizando reglas de inferencia, mientras que el model checking se enfoca en verificar propiedades específicas en modelos de sistemas finitos mediante la exploración exhaustiva de estados. Ambos enfoques tienen sus ventajas y se utilizan en diferentes contextos dependiendo del problema a abordar.