**Menciones importantes:**

Investigar greedy search algorithm.

Cuando la función heurística depende solo de h(n) (el futuro), el algoritmo de búsqueda toma el nombre de greedy search algorithm. Se utiliza generalmente para resolver problemas de optimización.

<http://elvex.ugr.es/decsai/algorithms/slides/4%20Greedy.pdf>

Resolution:

Es una regla de inferencia sólida y correcta, por lo que es completa.

Unification:

Es un algoritmo de unificación, que permite reducir 2 expresiones para ver si son equivalentes. Al decir equivalentes, quiere decir que son idénticas sintácticamente y lógicamente.

Inference engine.

Knowledge acquisition component of an expert system.

Sistemas inteligentes: Los sistemas inteligentes dependen del conocimiento involucrado, de la representación, de los procedimientos o algoritmos para el procesamiento, de la interfaz del usuario, del lenguaje empleado, etc.

Los sistemas expertos son programas capaces de solucionar problemas que para ello normalmente se requiere la intervención humana especializada (inteligencia).

El experto de área revela información colapsada sobre el conocimiento.

El ingeniero de conocimiento da forma simbólica y automáticamente manipulable a la información (conocimiento) dada por el experto del área.

Experto: Persona que cada vez sabe más cosas sobre menos cosas. Es una persona que ha dejado de pensar, sabe.

Sistema experto: Software que simula el razonamiento que realiza el ser humano.

Es un sabio, práctico, hábil, que tiene experiencia en una ciencia o arte.

Las técnicas de IA se usan para problemas que no tengan bien definido el problema y la solución.

La inteligencia artificial no busca simular inteligencia humana, sino reproducir los efectos de un comportamiento inteligente.

1950-1965 Classic Period:

Best-first search:

Es un algoritmo que utiliza 2 listas con un valor heurístico.

En la lista Open se incorporan todos los nodos a explorar con un valor heurístico asignado que me indica el orden.

En la lista Close se incorporan todos los nodos explorados que no tienen la solución como valor heurístico asignado. A futuro pueden ser un mejor valor si se cambia su valor heurístico, cuando cambia el valor heurístico por uno mejor lo pongo en Open.

Estrategias para búsqueda en espacio de estados:

Data-driven search (forward chainning):

In data-driven search, or forward chaining, the problem solve begins with the given facts of the problem and a set of legal moves or rules for changing state.

Search proceeds by applying rules to facts to produce new facts, which are in turn used by the rules to generate more new facts.

Busca una hipótesis que represente esos datos.

Goal driven (backward chainnig):

In goal-driven, or backward-driven: Take the goal that we want to solve. See what rules or legal moves could be used to generate this goal and determine what conditions must be true to use them. These conditions become the new goals, or subgoals, for the search. Search continues, working backward through successive subgoals until it works back to the facts of the problem.

Cuando usar goal driven y cuando usar data driven:

Goal driven: Cuando no hay muchos datos y el objetivo es claro.

Data driven:

Desventajas de generate and test:

No guarda un historial de pruebas anteriores.

Puedes evaluar la misma solución varias veces sin darte cuenta.

Comparación entre breath y Depth first:

Desventajas de depth first: no da la solución más corta necesariamente, y toma más tiempo.

Backtrack no es un algoritmo de búsqueda, es un algoritmo que te hace el trace de la solución.

Resolution: Es una regla completa, que ingresa R en una base de conocimiento y permite inferir nuevo conocimiento.

Es una regla de inferencia utilizada sobre cierto tipo de proposiciones lógicas y especialmente utilizada para los demostradores automatizados de teoremas.

Completo: Además de tener nuevas reglas sólidas me permite generar nuevo conocimiento.

Vamos a utilizar símbolos y estructuras, para razonar de manera artificial.

<https://es.wikipedia.org/wiki/Resoluci%C3%B3n_(l%C3%B3gica)>

El razonamiento artificial usando la lógica matemática: a través de símbolos y estructuras

usábamos unificación para verificar si las expresiones son sintácticamente equivalentes.

Pasos para que

1 Eliminar el entonces, cambiando lo por negación de a o b, no eliminar todavía los parenteis

2 Eliminamos el ámbito de la negación, reducimos la negación.

3 Estandarizamos las variables, cada cuantificador debe tener un solo tipo de variables.

4 Movemos todos los cuantificadores hacia la izquierda, manteniendo siempre el mismo orden.

5 Eliminar el existencial, usamos skolemization.

6 Eliminamos los cuantificadores universales.

7 Convertir las expresiones en conjunción de disyunciones, en las expresiones no debo tener ands.

8 Llamamos a cada término en la cláusula cómo cláusulas separadas

9 Vuelvo a estandarizar las variables, cada cláusula con diferentes variables.

**Resumen 1er parcial:**

***Definición de inteligencia artificial:***

**Técnicas para solucionar problemas de inteligencia artificial:**

La inteligencia artificial es una parte de las ciencias computacionales, enfocada a la creación de hardware y software con comportamiento inteligente.

**Objetivos a largo plazo:**

Desarrollar sistemas que alcancen el nivel de inteligencia similar/comparable/mejor que los humanos.

Esto es posible que ocurra en los próximos 10 a 15 años.

**Objetivos a corto plazo:**

En tareas específicas: Desarrollar sistemas que alcancen el nivel de inteligencia similar/comparable/mejor que los humanos.

La inteligencia artificial busca reproducir los efectos de un comportamiento inteligente.

**Las ciencias que contribuyeron al desarrollo de la inteligencia artificial son:**

Filosofía

Medicina

Matemáticas

Psicología

Lingüística

Ciencias computacionales e ingeniería.

**2 propósitos de la inteligencia artificial:**

Usar recursos computacionales para el empoderamiento del pensamiento humano y el desarrollo. Por ejemplo, la robótica y los sistemas expertos.

Para entender cómo piensan los humanos, no solo en la figura de qué es, sino también en cómo está hecho.

***Técnicas para solucionar problemas de inteligencia artificial:***

**Características de las técnicas usadas en la inteligencia artificial:**

Búsqueda: Provee un mecanismo para solucionar problemas donde no hay un método directo.

Uso del conocimiento: Provee un mecanismo para solucionar problemas complejos usando las estructuras existentes e información de los objetos involucrados.

Abstracción: Provee mecanismos para concentrarse en aquellos aspectos que son importantes para resolver el problema y separar aquellos que no son importantes y podrían hacer que el proceso colapse.

**Propiedades indeseables del conocimiento:**

Es voluminoso.

Es difícil de caracterizar.

Cambia.

Es comúnmente organizado de la misma forma que será usado.

**Las técnicas para solucionar problemas en IA son métodos que usan la representación del conocimiento:**

Representa generalizaciones, es decir, no es necesario representar cada situación particular.

Tiene que ser entendido por los proveedores de conocimiento.

Puede ser modificado para reflejar los cambios en el mundo real.

Puede ser usado en situaciones diferentes, por lo que puede ser incompleto o vago.

Puede ser usado para superar sus propias inexactitudes y reducir el rango de posibilidades que se deben considerar.

Las técnicas de IA generalmente incorporan procesos o métodos de búsqueda, que pueden ser considerados como árboles de búsqueda, en los cuales cada nodo representa un estado del problema y cada enlace representa la relación entre los estados enlazados.

En general, hay reglas que determinan como navegar por el árbol, y partes del árbol pueden ser generadas o expandidas como las reglas disparadas. Hay direcciones asociadas con los enlaces, en la dirección de la búsqueda (forward or backward search). También hay una estrategia de control para seleccionar las reglas que pueden ser aplicadas.

**Pasos necesarios para resolver un problema:**

Definir formalmente el problema.

Analizar el problema.

Representar el conocimiento necesario para solucionar el problema.

Seleccionar la mejor técnica para resolver el problema e implementarla.

**Para definir formalmente el problema:**

Primero creamos una descripción formal del problema, y lo reducimos en piezas manejables.

Los problemas artificiales están bien estructurados, y son fáciles de especificar.

Los problemas del mundo real, como el entendimiento del lenguaje natural, y otros problemas no son fáciles de especificar.

**Para crear una especificación formal de un problema, necesitamos definir:**

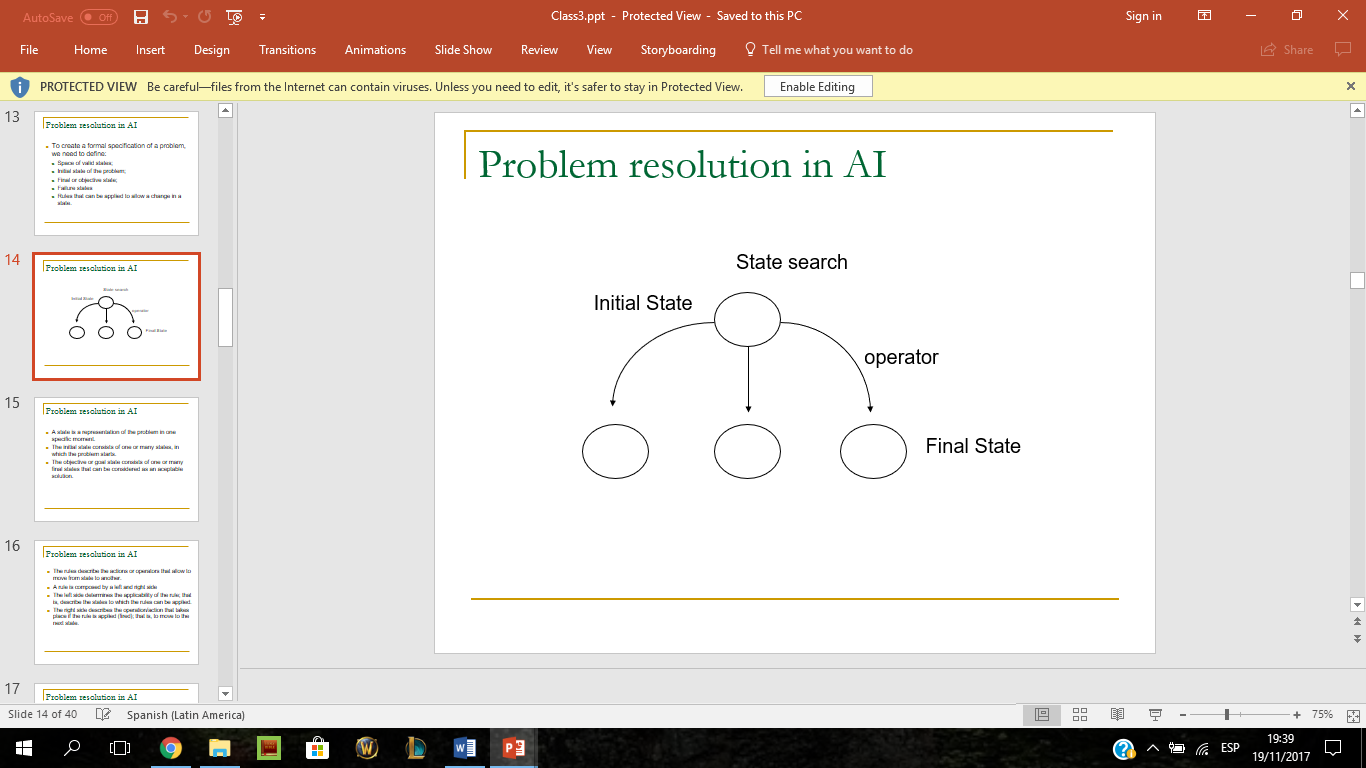
Espacio de estados válidos.

Estado inicial del problema.

Estado final u objetivo.

Estados falla.

Reglas que pueden ser aplicadas para permitir cambios de estado.



Un estado es una representación de un problema en un momento específico.

El estado inicial consiste en uno o muchos estados, en los que el problema empieza.

El objetivo o estado objetivo consiste en uno o muchos estados finales que pueden ser considerados como soluciones aceptables.

Las reglas describen las acciones u operadores que permiten moverse de un estado a otro.

Una regla está compuesta por una parte izquierda y una parte derecha.

La parte izquierda determina la aplicabilidad de la regla, esto es, describe los estados con los cuales las reglas pueden ser aplicadas.

La parte derecha describe la acción que toma lugar si la regla es aplicada, esto es, moverse al siguiente estado.

**Estrategias para la búsqueda de espacios estados:**

Goal driven - backward chaining: Toma el objetivo a ser solucionado, mira que reglas o movimientos legales pueden ser usados para generar este objetivo y determina qué condiciones deben ser verdaderas para usarlas. Estas nuevas condiciones se convierten en nuevos objetivos para la búsqueda, trabaja hacia atrás de los factores del problema. Encuentra la cadena de movimientos o reglas desde los datos hasta el objetivo, pero hacia atrás.

Data driven – forward chaining: La búsqueda empieza con los factores dados del problema y un conjunto de movimientos legales o reglas para cambiar de estado. Procedimientos de búsqueda aplicando reglas a los factores para producir nuevos factores, hasta que genera una ruta que satisface una condición objetivo.

**Tipos de búsqueda basados en estrategias de control:**

Algorítmica:

Tenemos información y modelos matemáticos sobre las operaciones a ser desarrolladas.

Búsqueda exhaustiva:

Explora el árbol de búsqueda sistemáticamente, pero sin información.

Búsqueda heurística:

Reduce la búsqueda con información sobre el dominio del problema.

**Estrategias para la búsqueda ciega:**

Generate and test.

Breadth first.

Depth first.

Uniform cost search.

Limited depth.

Interactive depth.

Bidirectional search.

**Generate and test:**

Genera una posible solución (estado o ruta).

Evalúa para ver si esta es la solución. Compara con los elementos de un conjunto soluciones aceptables.

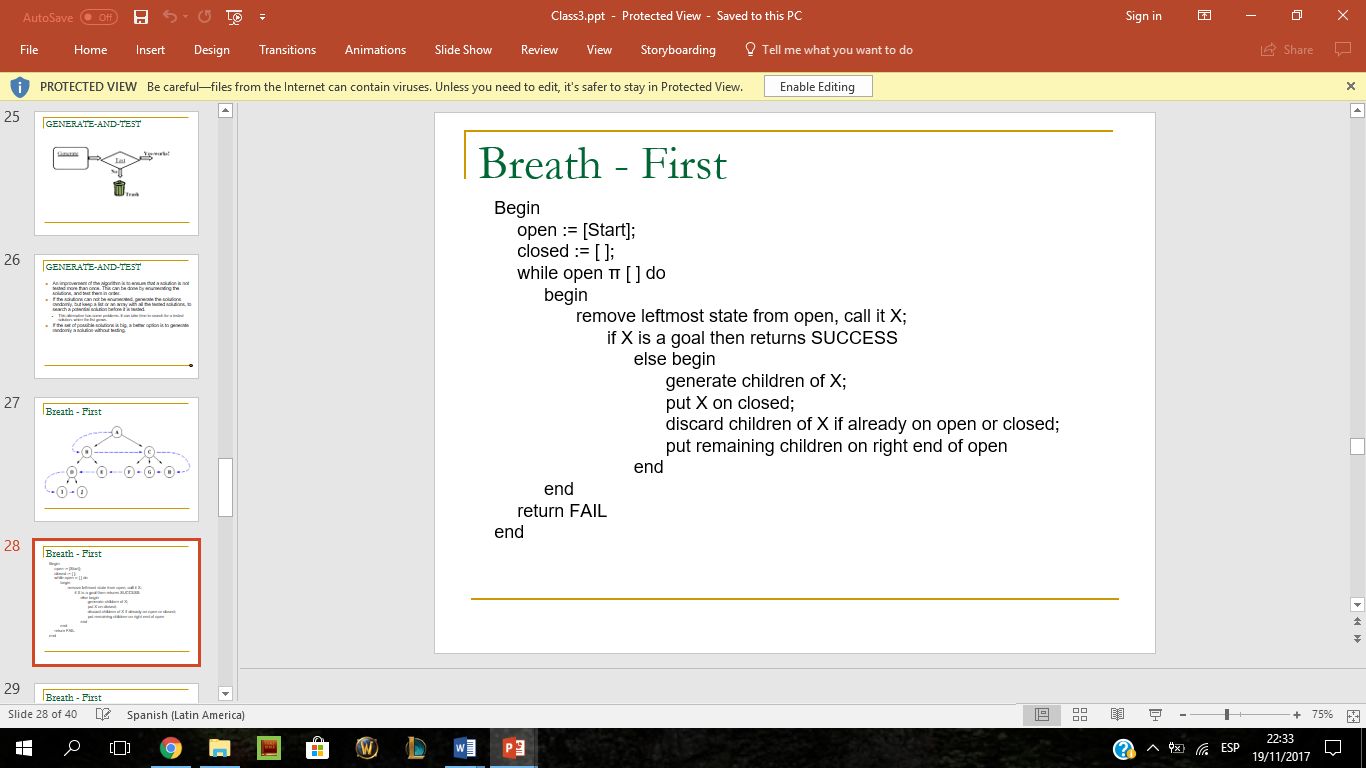
Si la solución ha sido encontrada, retorna. Si no, ve al paso 1.

Una mejora de este algoritmo es asegurarse que la solución no fue evaluada más de una vez. Esto puede ser hecho al enumerar las soluciones, y evaluarlas en orden.

Si las soluciones no pueden ser enumeradas, genera las soluciones de forma aleatoria, pero lleva un registro de la lista o arreglo con todas las soluciones evaluadas, para buscar una solución potencial antes de que esta sea evaluada. Un problema que puede ocurrir aquí es que puede tomar tiempo para alcanzar a evaluar una solución, cuando la lista crece.

Si el conjunto de soluciones posibles es grande, una mejor opción es generar aleatoriamente las soluciones sin evaluarla.

**Breadth first:**



Es completo, incluyendo para árboles finitos.

Evita caer en ciclos y repetir los nodos visitados.

Siempre encuentra la ruta más corta.

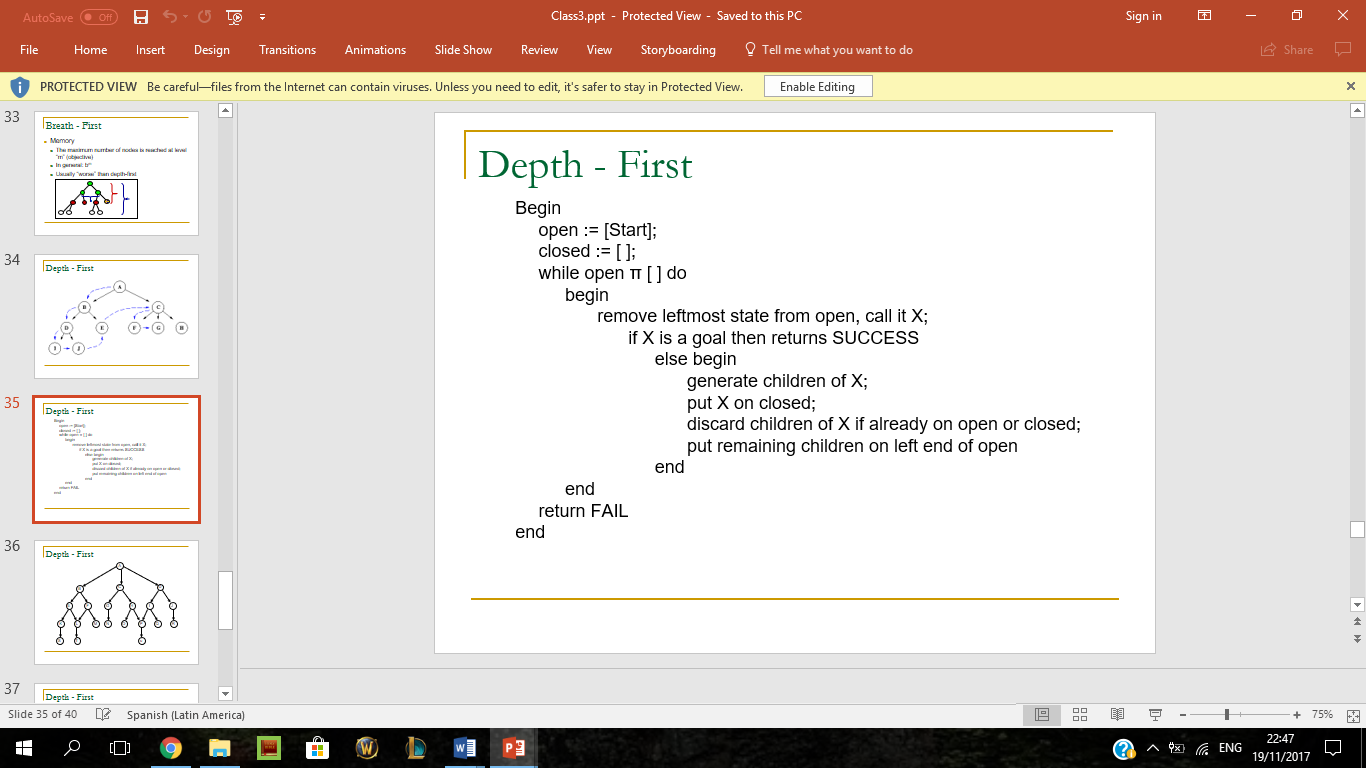
Tiempo: O(bm).

Memoria: El máximo número de nodos es alcanzado en el nodo “m”.

Memoria en general: bm.

Usualmente peor que Depth first.

**Depth first:**



Es completo, árboles infinitos.

Tiempo en el peor escenario: O(bd).

El estado final podría ser la última rama del árbol.

No da la ruta más corta necesariamente, y toma más tiempo.

Memoria: En general O(d\*b). Es eficiente en memoria.

***Búsqueda heurística:***

La información aparece como una función que vamos a maximizar o minimizar dependiendo del objetivo, la función determina la experiencia.

Los problemas en la búsqueda heurística:

Si el objetivo no es claro.

No garantiza la solución.

Si la función heurística está mal, todo el proceso está mal.

**Hill Climbing:**

Es una versión improvisada del algoritmo generate and test.

Para el proceso de evaluación, hay una retroalimentación que ayuda a decidir al generador de soluciones, en qué dirección ir después y moverse en el espacio de soluciones. En este proceso, la búsqueda finaliza si no existe un estado o nodo siguiente razonable para moverse.

El algoritmo hill climbing es local, porque decide que hacer después, basado solamente en las consecuencias de sus acciones inmediatas. Pero podría no alcanzar o encontrar la mejor solución, o quedarse atrapado en un nodo o nodos que no son objetivos.

**Problemas de hill climbing:**

Localidad máxima: Es el mejor estado o nodo comparado con sus vecinos, pero no es el mejor sobre los otros, que podría estar adelante en el espacio solución. Para solucionarlo, retornamos el mejor estado anterior y exploramos tomando una diferente solución.

Meseta o mesa: Es un estado en la búsqueda donde un conjunto de estados vecinos tienen valores similares. Para darle solución saltamos en una dirección e intentamos encontrar una nueva sección del espacio solución.

Acantilado: Es un tipo especial de máximo local, que es imposible de cruzar con movimientos simples. Aplicamos 2 o más reglas antes de que el nuevo nodo o estado sea evaluado, esto es equivalente a moverse en diferentes direcciones a la vez en el espacio solución.

**Simulated Annealing:**

Es una versión mejorada del algoritmo Hill Climbing. Al inicio, este algoritmo explora una buena parte del espacio solución, de tal manera que la solución final podría no resultar afectada desde el estado inicial, y como consecuencia, la probabilidad de quedarse atrapado en una localidad máxima, o mesa o acantilado, es pequeña. Es un proceso computacional que refleja los pasos establecidos en un proceso de tratamiento físico térmico usado con materiales. Por causa de su naturaleza, este algoritmo puede ser considerado como “descendiendo a un valle”.

**Ventajas de Simulated Annealing frente a Hill Climbing:**

Los movimientos a estados peores, que el estado actual, son aceptados.

El estado actual y el mejor estado desde el pasado, necesita ser almacenado, de esta forma, si el estado actual o estados futuros se ponen peores, entonces siempre hay la posibilidad de retornar al mejor estado desde el pasado.

**Algoritmo tipo A:**

Un algoritmo tipo A es toda función heurística de la forma f(n) = g(n) + h(n).

Un algoritmo tipo A es un algoritmo admisible, esto es, cuando el algoritmo garantiza una solución óptima, si esta existe.

**Best-first search:**

Este algoritmo combina las ventajas de breadth and depth first search algorithms.

Este sigue una ruta a la vez, pero esta puede cambiar la ruta de búsqueda, basado en el mejor camino prometedor que la ruta actual. El algoritmo usa 2 listas de nodos y una función heurística para estimar los méritos de cada nodo a ser explorado.

La lista Open son todos los nodos a explorar con un valor heurístico asignado que da el orden en dicha lista.

La lista Close son todos los nodos explorados que no tienen la solución con un valor heurístico asignado, pero a futuro pueden ser un mejor valor si se cambia su valor heurístico, entonces los saco y pongo en open.

La función heurística es: f(n) = g(n) + h(n).

f(n): Es la función heurística.

g(n): Es lo que me costó llegar desde el nodo inicial hasta n.

h(n): Es lo que me costará llegar desde n hasta el objetivo.

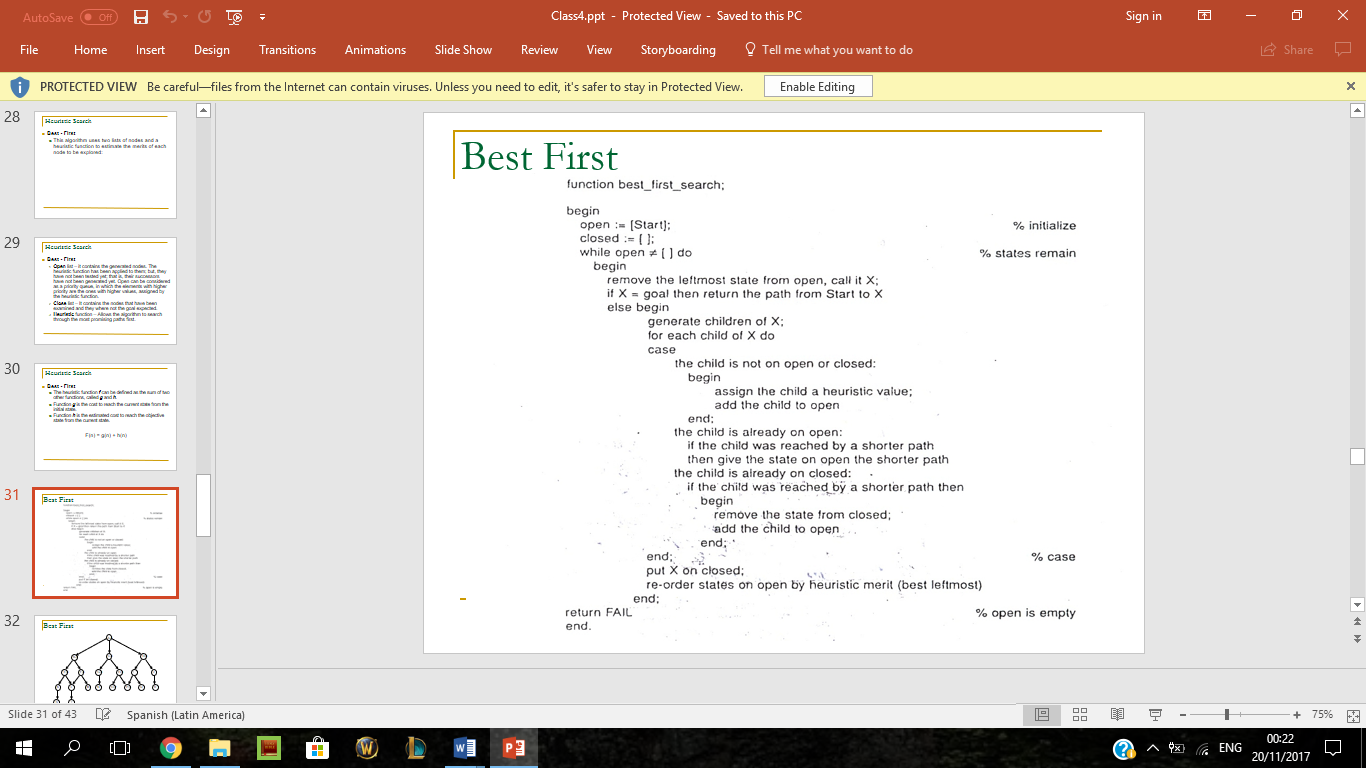
Best-first search es un algoritmo tipo A.

**Casos especiales de la función heurística:**

Cuando f(n) = h(n) se llama greedy search.

Cuando la función heurística depende solo de h(n) (el futuro), el algoritmo de búsqueda toma el nombre de greedy search algorithm. Se utiliza generalmente para resolver problemas de optimización.

Cuando f(n) = g(n) se llama uniform cost search.



**Teorema de admisibilidad de A\*:**

Si consideramos:

f\*(n) = g\*(n) + h\*(n).

Donde:

g\*(n): Costo de la ruta más corta desde el estado inicial hasta n.

h\*(n): Costo de la ruta más corta desde n hasta el objetivo.

Un algoritmo A\*, que usa h(n), donde h(n) ≤ h\*(n), para todo n, garantizará encontrar una ruta óptima desde n hacia el objetivo, si la ruta existe.

Breadth-first puede ser categorizado como un algoritmo A\*, en la cual f(n) = g(n) + 0.

**Backtracking:**

Es la ruta a la solución, no es un algoritmo de búsqueda.

**Representación del conocimiento:**

Debe ser:

Expresivo:

Permitir representar todo lo necesario.

Proveer esquemas naturales para representar el conocimiento requerido.

Eficiente:

Dar soporte a una ejecución eficiente de código de computación.

En general para permitir problemas cuantitativos y cualitativos.

Para organizar grandes y diferentes tipos de conocimiento, e implementar un algoritmo.

**Esquemas de la representación del conocimiento:**

Representación lógica:

Este usa expresiones formales lógicas para representar el conocimiento.

Cálculo de predicados en primer orden.

Ejemplo de representación lógica:

If all mobile objects are blue, then all non mobile objects are green.

(∀x mobile (x) ⇒ blue (x)) ⇒ (∀y ¬ mobile (y) ⇒ green (y))

Representación lógica:

El conocimiento es representado como un conjunto de instrucciones.

Tipos de reglas IF – THEN.

Ejemplo de representación procedural:

*If* temperature > 100o

*and* pressure < 75 dpi

*Then* free freezing liquid

Representación de red:

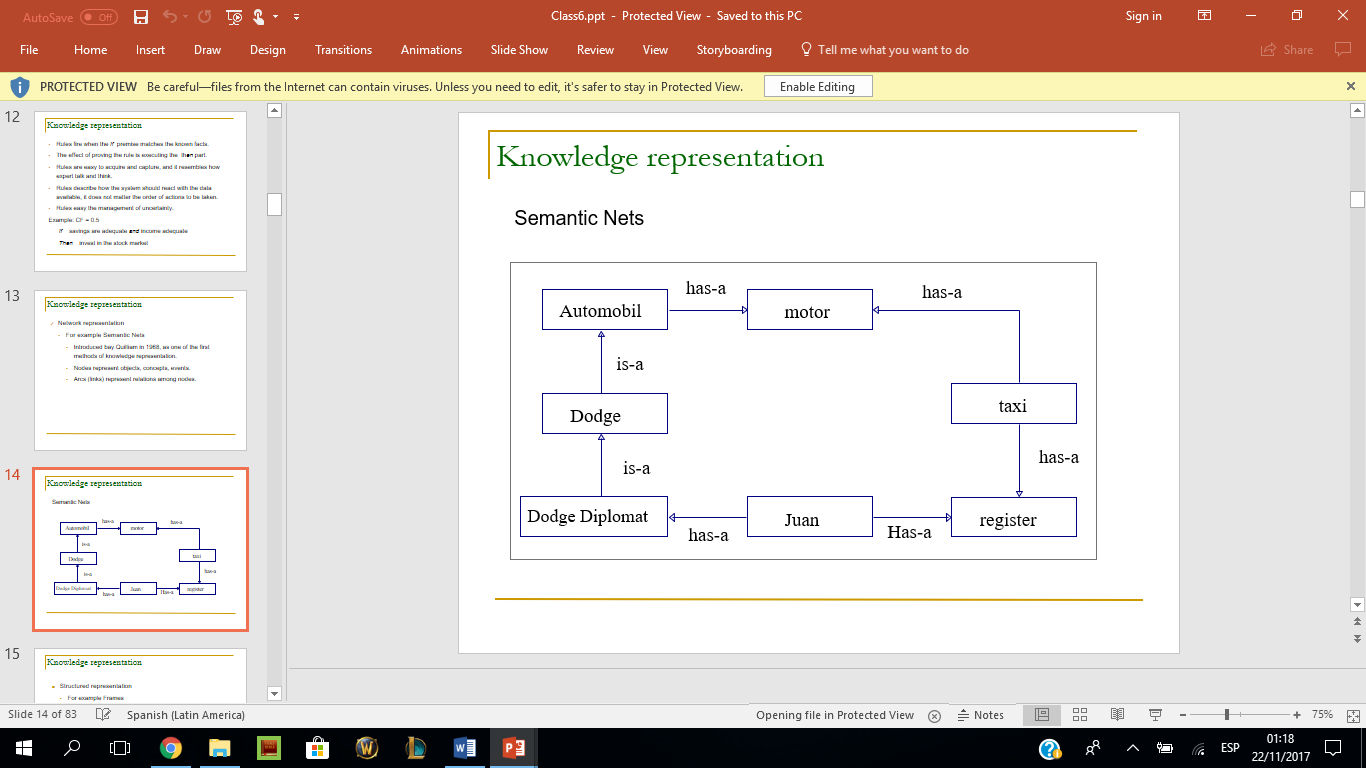
El conocimiento es capturado en forma de grafo.

Los nodos representan objetos o conceptos y los enlaces representan relaciones o asociaciones entre ellos.

Redes semánticas, grafos conceptuales.

Ejemplo de representaciones de red:

Redes semánticas.



Representación estructurada:

Esta es una extensión de la representación por redes.

Los nodos son estructuras complejas, que contienen slots con valores asociados.

Scripts, frames, objetcs.

Ejemplo de representación estructurada:

Syntactic structure of a Frame:

Frame: Name of a frame

Parent: Name of the father-1

. . .

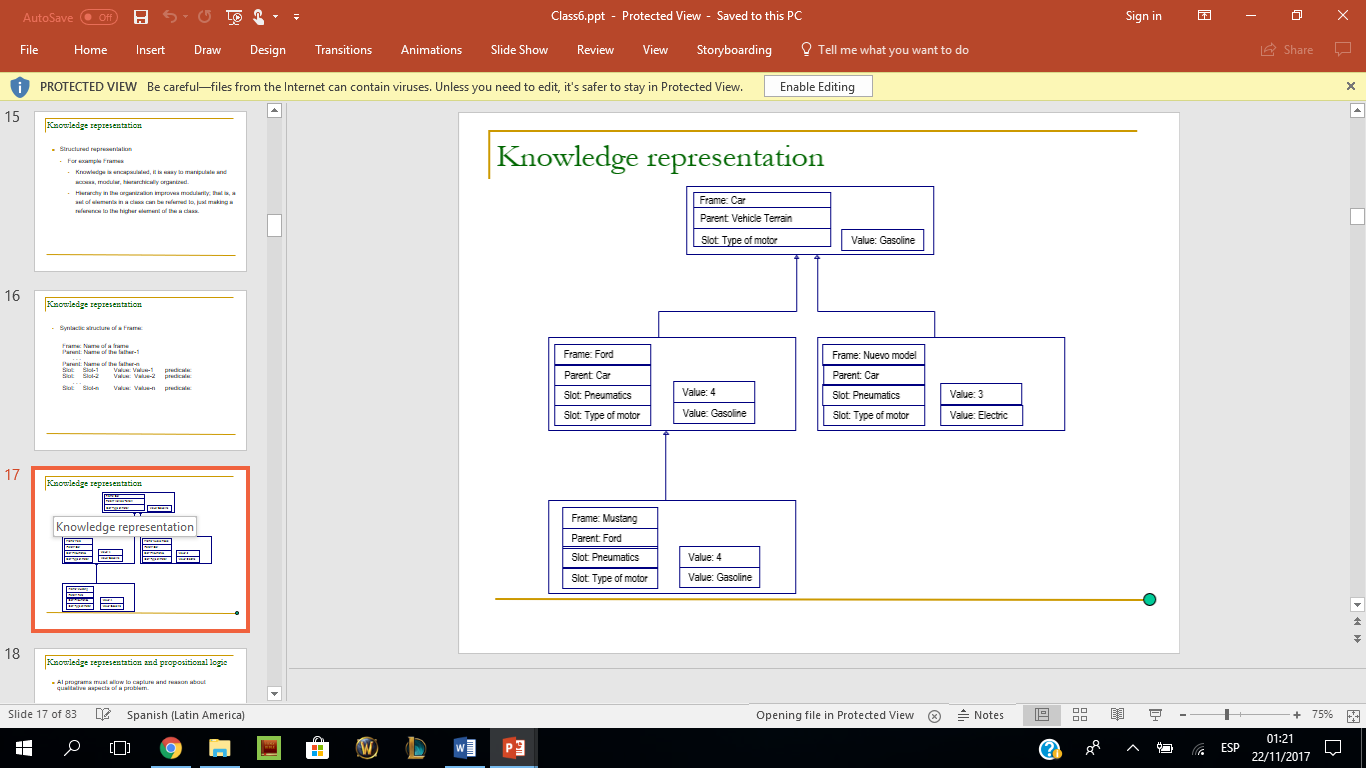
Parent: Name of the father-n

Slot: Slot-1 Value: Value-1 predicate:

Slot: Slot-2 Value: Value-2 predicate:

. . .

Slot: Slot-n Value: Value-n predicate:



Cálculo proposicional:

Palabras, frases o sentencias (proposiciones) en proposiciones y cálculo de predicados, pueden ser usados para representar y razonar sobre las propiedades y relaciones de los objetos en el mundo real.

Símbolos proposicionales:

P, Q, R, S, T, …

Símbolos de verdad:

True, False.

Concectores:

∧, ∨, ¬, →, =

**Predicados:{\displaystyle \lor }**

El cálculo de predicados provee el significado para acceder a los componentes individuales de una proposición.

Ejemplo:

Today is sunny.

Con el cálculo de predicados el clima puede describir la relación entre fecha y clima.

weather(today, sunny).

**Sintaxis:**

Los símbolos en el cálculo de predicados son elementos sintácticos y no pueden ser reducidos.

Letras (mayúsculas y minúsculas).

Dígitos del 0 al 9.

El símbolo “\_”.

Los símbolos en el cálculo de predicados empiezan con una letra.

Carlos, XYZ, father\_of.

Los argumentos de un predicado pueden incluir variables o funciones.

friends(father(david), father(andres))

friends(Xyz, manuel)

El cálculo de predicados permite cuantificar las variables.

Si el dominio de una variable es infinito, la expresión es llamada irreducible.

**Cálculo de predicados en primer orden:**

Esto permite a las variables cuantificadas hacer referencia a los objetos en el dominio del problema y no a otros predicados o funciones.

If does not rain tomorrow, John will go to the mountain:

¬ weather (rain, tomorrow) → go (jhon, mountain)

All basketball players are tall:

∀X (basketball\_player (X) → tall (X))

Nobody like taxes:

¬∃ X likes (X, taxes).

El método de representación de conocimiento debe:

Permitir inferir nuevo conocimiento empezando desde factores básicos.

Permitir representar principios generales tanto como situaciones específicas.

Capturar el significado semántico complejo.

Permitir meta-conocimiento.

**Reglas de inferencia:**

* La semántica en el cálculo de predicados provee las bases para una teoría de inferencia formal en lógica.
* La habilidad de inferir nuevas expresiones desde un conjunto de afirmaciones verdadera es la llave en el cálculo de predicados.
* Las nuevas expresiones son correctas porque son consistentes con la interpretación original.
* Cuando una interpretación hace verdadera una afirmación, se dice que la interpretación satisface la afirmación.

Una regla de inferencia en el cálculo de predicados es un mecanismo para producir nuevas afirmaciones a partir de otras afirmaciones.

Cuando cada afirmación X es producida por una regla de inferencia R, aplicada para un conjunto S de expresiones lógicas, la regla de inferencia es sólida y correcta.

Si cada regla de inferencia está en capacidad de producir nuevas expresiones que lógicamente siguen el conjunto S, entonces se dice que la regla es completa.

Ejemplo:

La regla de resolución.

**Unificación:**

Para inferir una nueva regla de un conjunto de reglas, necesitamos determinar si las expresiones son equivalentes.

2 expresiones son equivalentes si y solo si son sintácticamente iguales.

Unificación es un algoritmo que permite determinar las sustituciones necesarias para hacer 2 expresiones sintácticamente equivalentes.

Para permitir la libertad en las sustituciones, todas las variables deben estar universalmente cuantificadas.

Las variables cuantificadas existencialmente pueden ser eliminadas remplazándolas con valores constantes que hacen las expresiones verdaderas.

∀ X ∃ Y mother (Y, X)

Y depende del valor de X.

Skolemization: Es una acción que remplaza cada variable cuantificada existencialmente con una función que retorna el valor constante apropiado desde rango, como una función de otra variable.

∀ X mother (*f* (X), X).

La notación Y/X indica que, Y ha sido remplazada por una variable X, en la expresión original.

Una variable no puede ser unificada con un término que contenga la misma variable.

Ejemplo:

X no puede ser remplazada por g(X), porque produce la expresión: g(g(g(g(...X)...).

Si una variable X1 es remplazada por otra variable X2, y luego es remplazada por una constante, entonces X1 refleja el último cambio.

El completo conjunto de sustituciones usadas en el proceso de inferencia es importante porque podría contener la respuesta a la pregunta original.

El unificador debe ser lo más general posible.

**Modus ponens:**

Si tenemos una expresión de la forma p → q, y otra expresión p, que son verdaderas, entonces podemos inferir que otra expresión q también es verdadera.

**Modus tolens:**

Si tenemos una expresión de la forma p → q, que se conoce es verdadera, y se conoce que la expresión q es falsa, entonces inferimos que otra expresión p es falsa.

**AND elimination:**

Si tenemos una expresión de la forma p ∧ q, que se conoce es verdadera, podemos inferir que p y q, son ambas verdaderas.

Si tenemos 2 expresiones p, q que son verdaderas, entonces podemos inferir que p ∧ q también es verdadera.

**Universal instantiation:**

Si, cualquier variable cuantificada universalmente es una afirmación verdadera, es remplazada por otro termino desde el dominio, la afirmación resultante también es verdadera.

**Existencial instantiation:**

Si, cualquier variable cuantificada existencialmente, en una afirmación verdadera, es remplazada por un término apropiado del dominio, la afirmación resultante es también verdadera.

**Teorema de resolución:**

El principio de resolución es una técnica para probar teoremas usando cálculo de predicados.

Es un método de inferencia sólido, correcto y completo. Provee una forma para encontrar contradicción en una base de cláusulas con un mínimo conjunto de sustituciones.

Es una regla completa, que ingresa R en una base de conocimiento y permite inferir nuevo conocimiento.

Es una regla de inferencia utilizada sobre cierto tipo de proposiciones lógicas y especialmente utilizada para los demostradores automatizados de teoremas.

Completo: Además de tener nuevas reglas sólidas me permite generar nuevo conocimiento.

Vamos a utilizar símbolos y estructuras, para razonar de manera artificial.

En un proceso de refutación, un teorema puede ser probado negando la hipótesis, y añadiéndola, en forma de cláusula, para el conjunto de axiomas que son conocidos como verdaderos.

Resolución, como una regla de inferencia, tiene que probar que termina en una contradicción.

Una vez que la contradicción es encontrada, al negar la hipótesis, entonces se deduce que el objetivo original fue cierto y consistente con el conjunto original de axiomas.

**El proceso de resolución involucra los siguientes pasos:**

Convertir las afirmaciones en cláusulas.

Añadir la negación de lo que va a ser probado (la hipótesis), en forma de cláusula, al conjunto original de axiomas.

Resolver las cláusulas, produciendo nuevas cláusulas que lógicamente vienen del conjunto original.

Produce o encuentra una contradicción, genera la cláusula vacía.

Extrae los resultados de las sustituciones.

**Convertir a forma de cláusulas:**

1. Eliminar → usando su forma equivalente: ¬, ∧.
2. Reducir el alcance de la negación. Usando las transformaciones:
3. Estandarizar las variables, cada cuantificador debe dar soporte únicamente un tipo de variable.
4. Mover todos los cuantificadores hacia la izquierda, sin cambiar el orden.
5. Eliminar los cuantificadores existenciales, usando skolemization.
6. Eliminar los cuantificadores universales.
7. Convertir la expresión en una conjunción de disyunciones usando las propiedades de v y ∧.
8. Llamar cada término en la conjunción como cláusulas separadas.
9. Estandarizar las variables.

**2 métodos para Extraer los resultados en resolución:**

Mantener la conclusión original o hipótesis, y rastrear, cada una de las sustituciones hechas durante el proceso de resolución.

Añadir a la negación de la hipótesis, un elemento con igual número de variables que la hipótesis, y llevar un registro de todas las sustituciones hechas durante el proceso de resolución, en estas variables.