Universidad de Costa Rica | Escuela de Ciencias de la Computacion e Informatica

**CI-1441 Paradigmas computacionales** **Examen Final**

**II Semestre 2018**

Profesor: Álvaro de la Ossa Osegueda, Dr.rer.nat. Entrega: 9 de diciembre de 2018

Estudiantes: Mayrene Raquel Anchía Quesada - B40371

Patrick Loney Rojas - B43834

1. **En el capítulo 16 del libro de Luger y Stubblefield (2009), que trata la integración de la programación lógica y la programación funcional, se construye un intérprete Prolog en Lisp, cuya función central es solve, que recibe una meta a probar y un conjunto de sustituciones, y devuelve todas las soluciones consistentes con la base de conocimientos. En el caso general (la meta es una conjunción), se invoca a una función llamada filter-through-conj-goals, que recorre la lista de submetas, y en el caso trivial (la meta es simple), se invoca a la función infer.**

**a. 15 pts. ¿Qué información produce infer sobre la meta que se desea probar con el intérprete?**

*Infer* toma una meta y un conjunto de sustituciones y encuentra todas las soluciones en la base de conocimientos , que es el tercer parámetro de *infer*, una base de expresiones lógicas. Cuando *solve* llama a *infer*, le pasa la base de conocimientos (*kb*) contenida en la variable global "*assertions*". *Infer* realiza una búsqueda secuencial en la base de conocimientos, probando la *meta* contra cada uno de los hechos o reglas de conclusión.
La implementación recursiva de *infer* construye la búsqueda de encadenamiento hacia atrás típica de Prolog. Primero verifica, si *kb* está vacía, devuelve un flujo vacío. De lo contrario, enlaza el primer elemento en la *kb* al símbolo "*assertion*" utilizando un bloque let*. También define la coincidencia de la variable: si "*assertion*" es una regla, let* inicializa a "*match*" como las sustituciones requeridas para unificar la meta con el primer elemento de la base de conocimientos, *infer* prueba si esta unificación tuvo éxito. Si no fue así, *infer* se llama recursivamente. Pero si "assertion" no es una regla, es decir, es un hecho, infer agrega la solución enlazada a "match" a aquellas dadas por el resto de la base de conocimientos.

Referencias:

- Luger, G., Stubblefiled, W. (2009). AI Algorithms, Data Structures, and Idioms in Prolog, Lisp, and Java. (p. 213-214).

**b. 15 pts. ¿Que efectos tiene la ejecución de infer en la base de conocimientos (por ejemplo, ¿se añaden o se eliminan expresiones de la base?).**

La ejecución de infer agrega nuevas expresiones a la base de conocimientos. Como se explicó anteriormente, si "assertion" es un hecho, infer agrega la solución enlazada a "match" a aquellas dadas por el resto de la base de conocimientos. Ese necesario notar que una vez que la meta se unifica con un hecho, queda resuelta y esto termina la búsqueda.

Referencias:


- Luger, G., Stubblefiled, W. (2009). AI Algorithms, Data Structures, and Idioms in Prolog, Lisp, and Java. (p. 213-214).


2. **10 pts. En la Teoría de la Complejidad Computacional, ¿que quiere decir que un problema sea tratable?**

En la Teoría de la Complejidad Computacional un problema 'P' es tratable si existe un algoritmo 'A' que lo resuelve haciendo un uso a lo más polinomial de los recursos, es decir un problema es tratable si se puede resolver en tiempo polinómico $O(n^k)$ para algún k. Estos recursos son:

- Tiempo de procesadores, es decir es número de pasos base de ejecución de un algoritmo para resolver un problema.
- Espacio de representación del problema. es decir la cantidad de memoria utilizada para resolver un problema.
- Comunicación: mensajes entre procesos.

Referencias:


- Alseco, R. (2014). Complejidad de problemas. Recuperado de: https://www.slideshare.net/rodrigoalseco/complejidad-de-problemas.
- de la Ossa, A. (2018). *INTRODUCCIÓN A LAS TEORÍAS DE LA COMPUTABILIDAD Y LA COMPLEJIDAD COMPUTACIONAL* (p. 4).
- Sanchis, A., Ledezma, A., Iglesias, J., García, B., & Alonso, J. (2018). Complejidad computacional. Recuperado de: http://ocw.uc3m.es/ingenieria-informatica/teoria-de-automatas-y-lenguajes-formales/material-de-clase-1/tema-8-complejidad-computacional


3. **10 pts. En el marco de trabajo de la Inteligencia Artificial y las Ciencias Cognoscitivas estudiado en el curso, ¿que quiere decir que un modelo de inteligencia artificial sea psicológicamente plausible?**

Que un modelo de inteligencia artificial sea psicológicamente plausible significa que cumple con la siguiente condición:
Dada una población de sujetos humanos a quienes se les pide resolver una tarea particular, una simulación del modelo resolviendo la misma tarea debe exhibir un comportamiento similar al del conjunto de sujetos. Esto se debe cumplir al menos en dos tipos de métrica: tiempo de recuperación y precisión de la respuesta.
También se puede decir que un modelo es psicológicamente plausible si responde de manera afirmativa a la siguiente pregunta: ¿hay experiencias psicológicas que permitan evaluar si dicho entendimiento puede mecanizarse?

Referencias:


- de la Ossa, A. (2018). El marco de trabajo de las Ciencias Cognoscitivas y el paradigma de Razonamiento Basado en Casos Archivo (p. 24).

- González, R. (2016). El entendimiento lingüístico en la Inteligencia Artificial: Una relación ambivalente con Descartes (p. 24).

4. **10 pts. Explique las medidas de trabajo y eficiencia de la complejidad computacional de algoritmos paralelos.**

   El trabajo y la eficiencia son medidas de rendimiento en complejidad computacional de algoritmos paralelos.

   - La eficiencia E es una medida del uso de los recursos, y se define como la relación entre la aceleración y la cantidad de procesadores utilizados.
     Se puede calcular como E = S/p. Donde S es la aceleración y p es la cantidad de procesadores utilizados. Si E = 1, la eficiencia es lineal.
   - Por otro lado, el trabajo W es el tiempo total de ejecución.

   Referencias:

   - de la Ossa, A. (2018). *INTRODUCCIÓN A LAS TEORÍAS DE LA COMPUTABILIDAD Y LA COMPLEJIDAD COMPUTACIONAL* (p. 44 y 45).

5. **A continuación se describen un sistema multiagente con tres tipos de agente. Seleccione dos de ellos y responda las preguntas que se presentan al final.**

   **Se tiene un SMA de monitoreo de una parcela agrícola. Su objetivo principal es detectar en forma temprana y alertar al productor agrícola de la presencia de plagas. Las clases de agente son: detectores de plagas (computacionales), asesores (computacionales), y productores agrícolas (humanos); el entorno de los agentes está conformado por la parcela, la base de datos de plagas, y una lista de recomendaciones del asesor.**

   **Agente detector de plagas. Este tipo de agente toma en intervalos de 4 horas imágenes de la parcela y las procesa para determinar la presencia de plagas (p.ej., nematodos, ácaros, hongos). Diariamente produce 6 registros en la base de datos. Utiliza métodos de procesamiento de señales y de reconocimiento de patrones para determinar si hay presencia de alguna plaga, y al concluir el análisis, añade al registro esa información.**

   **Agente asesor. Este tipo de agente posee una lista de todos los tipos de acción que es posible llevar a cabo para atacar las plagas. Actúa una vez al día. Para cada uno de los seis últimos registros del detector de plagas, selecciona las acciones aplicables de la lista general, las instancia en cada caso (es decir, las especifica o las ajusta a las condiciones informadas por el agente detector de plagas), y se las envía al agente productor.**

   **Agente productor. Este agente revisa una vez al día la lista de recomendaciones producidas por el agente asesor. El objetivo del productor es maximizar su**

**ganancia futura derivada de la venta de sus productos, por lo que ordena la lista de recomendaciones producidas por el agente asesor, en orden descendente de su probabilidad de maximizar la destrucción de la plaga o de minimizar el daño a los productos en la parcela.**

**a) 10 pts. c/u De las arquitecturas de agente descritas por Russell y Norvig (2010) en el capítulo 2 de su libro, ¿Cuál es más apropiada para construir el agente, y por qué?**

Agentes elegidos:

- Agente detector de plagas: Consideramos que la arquitectura más apropiada para construir este tipo de agente es la de "Agentes reflejo simple". Ya que un agente detector de plagas lo que hace es tomar imágenes de la parcela en intervalos de tiempo, procesarlas para determinar la presencia de plagas y cuando concluye guarda esta información. Y los agentes reflejo simple seleccionan acciones en base a la percepción actual, ignorando el resto del historial de percepción. No es necesario que el agente detector de plagas maneje un historial de percepción, ya que el solo necesita basarse en lo que puede observar en su entorno cuando toma las imágenes. Es por eso que una arquitectura de agente reflejo simple sería viable para implementar este tipo de agente detector.
- Agente productor: Para este agente, consideramos que la arquitectura más apropiada para construirlo es la de "Agentes Basados en Utilidad", ya que, los agentes productores deben tomar las recomendaciones de los agentes asesores y su objetivo es maximizar su ganancia futura de la venta de sus productos, y para ello ordena descendentemente las recomendaciones del asesor.
  Un agente basado en la utilidad tiene muchas ventajas en términos de flexibilidad y aprendizaje. Un agente basado en utilidad puede tomar decisiones racionales, por ejemplo, si un agente puede lograr varios objetivos pero el nivel de certeza de que dichos objetivos se alcance es incierta, la utilidad provee una forma en la que la importancia de los objetivos puede compararse con la probabilidad de éxito. Un agente basado en la utilidad racional elige la acción que maximice la utilidad esperada de los resultados de la acción.

  Referencias**:**
  
  - Russell, S., Norving, P. (2010). Artificial Intelligence A Modern Approach. (p. 46-55).
  

**b) 10 pts. c/u De los paradigmas y métodos de la lista siguiente, ¿cuál o qué combinación de ellos es más apropiada para implementar el agente, y por qué?**

**a. Sistemas de reglas de producción**
**b. Arboles de decision (ID3)**
**c. Aprendizaje analítico (EBL)**
**d. Redes neuronales artificiales (RNA)**

**e. Computación evolutiva**
**f. Otro (especifique)**

- Para el agente detector de plagas: El paradigma de redes neuronales artificiales sería muy apropiado, ya que el detector de plagas necesita reconocer patrones y esto es precisamente lo que realiza una red neuronal artificial. Una red de neuronal está organizada por capas y es capaz de aprender una distribución de niveles de activación de las neuronas tal que, cada vez que la red recibe como entrada una representación del patrón reconocible, la salida es cercana a 1, y es cercana a 0 para cualquier otro patrón de entrada.

  Además quizá podría combinarse una RNA con el método ID3, mediante la red neuronal entrenada se extraen árboles de decisión y con el método ID3 se construyen reglas de clasificación a partir de conjuntos extensos de ejemplos y contraejemplos. De esta manera, utilizando las imágenes de la parcela como entrada, podría determinarse la presencia de plagas.
- Para el agente productor: el paradigmas de Computación evolutiva puede ser el más apropiado para la implementación del agente, esto debido a que, como se mencionó en la arquitectura apropiada para el agente productor, maximizar la utilidad de los resultados esperados (la ganancia futura de la venta de los productos) es su objetivo principal. Dado esto cabe destacar que la computación evolutiva sirve como método de optimización, es decir, dada una función objetivo a maximizar, es posible crear un conjunto de soluciones candidatas; también, la computación evolutiva resuelve el problema de seleccionar de un conjunto de reglas, el conjunto que mejor ayuda a predecir un tipo de evento y problemas que requieren de la adaptación del proceso de solución a lo largo del tiempo.

  Referencias:
  
  - de la Ossa, A. (2018). *Aprendizaje 1/2: aprendizaje simbólico supervisado.* (p. 8 a 15).
  - de la Ossa, A. (2018). *Aprendizaje 2/2: Redes neuronales artificiales (RNA).* (p. 11 a 16).
  - de la Ossa, A. (2018). *Computación Evolutiva.* (p. 1 a 7).
  

### 2.4.2 Agentes reflejo simple

El tipo más simple de agente es el simple agente de reflejo. Estos agentes seleccionan acciones en base al de la percepción actual, ignorando el resto del historial de percepción. Por ejemplo, el agente de vacío cuya función de agente está tabulada en la Figura 2.3 es un simple agente reflejo, porque su decisión se basa solo en la ubicación actual y en si esa ubicación contiene suciedad. Un programa de agente para este agente se muestra en la Figura 2.8.

Observe que el programa de agente de vacío es muy pequeño en comparación con la tabla correspondiente. La reducción más obvia proviene de ignorar el historial de percepción, lo que reduce el número de posibilidades de 4T a solo 4. Una pequeña reducción adicional proviene del hecho de que cuando el cuadro actual está sucio, la acción no depende de la ubicación. Los comportamientos reflejos simples se producen incluso en entornos más complejos. Imagínate a ti mismo como el conductor del taxi automatizado. Si el automóvil en los frenos delanteros y sus luces de freno se encienden, debería notar esto e iniciar el frenado. En otras palabras, se realiza algún procesamiento en la entrada visual para establecer la condición que llamamos "El automóvil que está en frente frena". Luego, esto activa una conexión establecida en el programa del agente a la acción "iniciar el frenado". conexión una regla de condición-acción, 5 escrita como if car-in-front-is-braking then initiate-braking.

Los humanos también tienen muchas conexiones de este tipo, algunas de las cuales son respuestas aprendidas (como para conducir) y otras son reflejos innatos (como parpadear cuando algo se acerca al ojo). En el curso del libro, mostramos varias formas diferentes en que se pueden aprender e implementar dichas conexiones.

El programa de la Figura 2.8 es específico para un entorno de vacío particular. Un enfoque más general y flexible es primero construir un intérprete de propósito general para reglas de condición-acción y luego crear conjuntos de reglas para entornos de tareas específicas. La figura 2.9 muestra la estructura de este programa general en forma esquemática, que muestra cómo las reglas de condición-acción permiten que el agente realice la conexión de percepción a acción. (No se preocupe si esto parece trivial, se pondrá más interesante en breve).

Usamos rectángulos para denotar el estado interno actual del proceso de decisión del agente, y óvalos para representar la información de fondo utilizada en el proceso. El programa del agente, que también es muy simple, se muestra en la Figura 2.10. La función INTERPRET-INPUT genera una descripción abstracta del estado actual a partir de la percepción, y la función RULE-MATCH devuelve la primera regla en el conjunto de reglas que coincide con la descripción del estado dado. Tenga en cuenta que la descripción en términos de "reglas" y "coincidencia" es puramente conceptual; Las implementaciones reales pueden ser tan simples como una colección de compuertas lógicas que implementan un circuito booleano.

Los agentes reflejos simples tienen la admirable propiedad de ser simples, pero resultan ser de inteligencia limitada. El agente en la Figura 2.10 funcionará solo si la decisión correcta puede tomarse sobre la base de la percepción actual, es decir, solo si el entorno es completamente observable. Incluso un poco de inobservabilidad puede causar serios problemas. Por ejemplo, la regla de frenado dada anteriormente asume que la condición del auto en el frente es el frenado se puede determinar a partir de la percepción actual, un solo cuadro de video. Esto funciona si el auto en frente tiene una luz de freno montada centralmente. Desafortunadamente, los modelos más antiguos tienen diferentes configuraciones de luces traseras, , y no siempre es posible decir de una sola imagen si el automóvil está frenando. Un simple agente reflejo que conduce detrás de un automóvil de este tipo frenaría de forma continua e innecesaria, o, peor aún, nunca frenaría en absoluto.

2.4.3 Agentes reflejo basados en modelos

La forma más efectiva de manejar la observabilidad parcial es que el agente realice un seguimiento de la parte del mundo que no puede ver ahora. Es decir, el agente debe mantener algún tipo de estado interno que dependa del historial de percepción y, por lo tanto, refleje al menos algunos de los aspectos no observados del estado actual. Para el problema de frenado, el estado interno no es demasiado extenso: solo el cuadro anterior de la cámara, lo que permite al agente detectar cuando dos luces rojas en el borde del vehículo se encienden o apagan simultáneamente. Para otras tareas de manejo, como cambiar de carril, el agente necesita realizar un seguimiento de dónde están los otros autos si no puede verlos todos a la vez.

Y para que cualquier conducción sea posible, el agente necesita realizar un seguimiento de dónde están sus llaves.

La actualización de esta información de estado interno a medida que pasa el tiempo requiere que se codifiquen dos tipos de conocimiento en el programa del agente. Primero, necesitamos información acerca de cómo evoluciona el mundo independientemente del agente, por ejemplo, que generalmente un automóvil que va a adelantar estará más cerca que hace un momento. En segundo lugar, necesitamos cierta información sobre cómo las propias acciones del agente afectan al mundo; por ejemplo, cuando el agente gira el volante en el sentido de las agujas del reloj, el automóvil gira a la derecha o después de conducir durante cinco minutos hacia el norte en la autopista, uno estará por lo general, a unas cinco millas al norte de donde se encontraba hace cinco minutos. Este conocimiento sobre "cómo funciona el mundo", ya sea implementado en circuitos booleanos simples o en teorías científicas completas, se denomina modelo del mundo. Un agente que utiliza un modelo de este tipo se denomina agente basado en modelos.

2.4.4 Agentes basados en metas

Saber algo sobre el estado actual del medio ambiente no siempre es suficiente para decidir qué hacer. Por ejemplo, en un cruce de carreteras, el taxi puede girar a la izquierda, girar a la derecha o seguir recto. La decisión correcta depende de dónde intenta llegar el taxi. En otras palabras, además de una descripción del estado actual, el agente necesita algún tipo de información de meta que describa las situaciones que son deseables, por ejemplo, estar en el destino del pasajero. El programa del agente puede combinar esto con el modelo (la misma información que se usó en el agente reflejo basado en modelos) para elegir acciones que alcancen la meta.

A veces, la selección de la acción basada en objetivos es sencilla, por ejemplo, cuando la satisfacción de los objetivos resulta inmediatamente de una sola acción. A veces será más complicado, por ejemplo, cuando el agente tiene que considerar largas secuencias de giros y vueltas para encontrar una manera de lograr el objetivo.

Observe que la toma de decisiones de este tipo es fundamentalmente diferente de las reglas de condición-acción descritas anteriormente, en que involucra la consideración del futuro, tanto

"¿Qué pasará si hago tal o cual cosa?" Y "¿Eso me hará feliz? ? ”En los diseños de los agentes reflejos, esta información no se representa explícitamente, porque las reglas incorporadas se asignan directamente de las percepciones a las acciones. El agente reflejo frena cuando ve luces de freno. Un agente basado en objetivos, en principio, podría razonar que si el automóvil en frente tiene sus luces de freno encendidas, disminuirá la velocidad. Dada la forma en que el mundo suele evolucionar, la única acción que logrará el objetivo de no golpear a otros autos es frenar.

Observe que la toma de decisiones de este tipo es fundamentalmente diferente de las reglas de condición-acción descritas anteriormente, en que involucra la consideración del futuro, tanto "¿Qué pasará si hago tal o cual cosa?" Y "¿Eso me hará feliz? ? ”En los diseños de los agentes reflejo, esta información no se representa explícitamente, porque las reglas incorporadas se asignan directamente de las percepciones a las acciones. El agente reflejo frena cuando ve luces de freno. Un agente basado en objetivos, en principio, podría razonar que si el automóvil en frente tiene sus luces de freno encendidas, disminuirá la velocidad. Dada la forma en que el mundo suele evolucionar, la única acción que logrará el objetivo de no golpear a otros autos es frenar.

2.4.5 Agentes basados en utilidad **- Nos Sirve para el agente productor!**

Los objetivos por sí solos no son suficientes para generar un comportamiento de alta calidad en la mayoría de los entornos. Por ejemplo, muchas secuencias de acción llevarán el taxi a su destino (logrando así el objetivo) pero algunas son más rápidas, más seguras, más confiables o más baratas que otras. Los objetivos solo proporcionan una distinción binaria cruda entre estados "felices" e "infelices". Una medida de rendimiento más general debería permitir una comparación de diferentes estados del mundo según exactamente qué tan felices serían los agentes. Debido a que "feliz" no suena muy científico, economistas e informáticos usan el término utilidad en su lugar.

Ya hemos visto que una medida de rendimiento asigna una puntuación a cualquier secuencia de estados ambientales, por lo que puede distinguir fácilmente entre las formas más y menos deseables de llegar al destino del taxi. La función de utilidad de un agente es esencialmente una internalización de la medida de desempeño. Si la función de utilidad interna y la medida de rendimiento externo están de acuerdo, entonces un agente que elige acciones para maximizar su utilidad será racional de acuerdo con la medida de rendimiento externo.

Hagamos hincapié nuevamente en que esta no es la única manera de ser racional: ya hemos visto un programa de agentes racionales para el mundo vacío (Figura 2.8) que no tiene idea de cuál es su función de utilidad, pero, como los agentes basados en objetivos, un agente basado en la utilidad tiene muchas ventajas en términos de flexibilidad y aprendizaje. Además, en dos tipos de casos, los objetivos son inadecuados, pero un agente basado en la utilidad todavía puede tomar decisiones racionales. Primero, cuando hay objetivos en conflicto, sólo algunos de los cuales se pueden lograr (por ejemplo, velocidad y seguridad), la función de utilidad especifica la compensación adecuada. En segundo lugar, cuando el agente puede alcanzar varios objetivos, ninguno de los cuales se puede lograr con certeza, la utilidad proporciona una forma en la que la probabilidad de éxito puede compararse con la importancia de los objetivos.

La observabilidad parcial y la estocasticidad son omnipresentes en el mundo real, por lo que, por lo tanto, la toma de decisiones es incierta. En términos técnicos, un agente basado en la

utilidad racional elige la acción que maximiza la utilidad esperada de los resultados de la acción, es decir, la utilidad que el agente espera obtener, en promedio, dadas las probabilidades y utilidades de cada resultado. (El Apéndice A define las expectativas con mayor precisión). En el Capítulo 16, mostramos que cualquier agente racional debe comportarse como si tuviera una función de utilidad cuyo valor esperado intenta maximizar. Un agente que posee una función de utilidad explícita puede tomar decisiones racionales con un algoritmo de propósito general que no depende de la función de utilidad específica que se maximiza. De esta manera, la definición "global" de racionalidad, que designa como racionales las funciones de agente que tienen el mayor rendimiento, se convierte en una restricción "local" en los diseños de agente racional que se puede expresar en un programa simple.

La estructura del agente basado en la utilidad aparece en la Figura 2.14. Los programas de agentes basados en servicios públicos aparecen en la Parte IV, donde diseñamos agentes de toma de decisiones que deben manejar la incertidumbre inherente en entornos estocásticos o parcialmente observables.

En este punto, el lector puede estar preguntándose: "¿Es así de simple? Simplemente construimos agentes que maximizan la utilidad esperada, ¿y estamos listos? "Es cierto que tales agentes serían inteligentes, pero no es simple. Un agente de servicios públicos tiene que modelar y realizar un seguimiento de su entorno, tareas que han implicado una gran cantidad de investigación sobre la percepción, la representación, el razonamiento y el aprendizaje. Los resultados de esta investigación llenan muchos de los capítulos de este libro. Elegir el curso de acción que maximiza la utilidad también es una tarea difícil, ya que requiere algoritmos ingeniosos que llenan varios capítulos más. Incluso con estos algoritmos, la racionalidad perfecta es generalmente inalcanzable en la práctica debido a la complejidad computacional, como señalamos en el Capítulo 1.