

# Inteligencia Artificial

## Inferencia en lógica

Primavera 2007

profesor: Luigi Ceccaroni



# Inferencia en lógica

- Se quieren conseguir algoritmos que pueden responder a preguntas expresadas en forma lógica.
- Tres grandes familias de algoritmos de inferencia:
  - **encadenamiento hacia delante** y sus aplicaciones en las bases de datos deductivas y en los **sistemas de producción**
  - **encadenamiento hacia atrás** y los sistemas de programación lógica
  - sistemas de demostración de teoremas basados en la resolución

# Sistemas de producción

- La representación mediante formalismos lógicos es declarativa pero puede representar procedimientos.
- Se describen cuales son los pasos para resolver un problema como una cadena de deducciones.
- La representación se basa en dos elementos:
  - **hechos**: proposiciones o predicados;
  - **reglas**: formulas condicionales donde el consecuente está formado por un único átomo.
- Analogía con búsqueda por reducción a sub-problemas:
  - **hechos** = problemas primitivos;
  - **reglas** = operadores de reducción.

# Sistemas de producción

Un problema queda definido por:

- **Base de hechos:** predicados que describen el problema concreto
- **Base de conocimiento** (o de reglas): reglas que describen los mecanismos de razonamiento que permiten resolver problemas
- **Motor de inferencia:** interprete que ejecuta las reglas y obtiene la cadena de razonamiento que soluciona el problema

La solución de un problema depende del proceso seguido por el interprete.

# Hechos: terminología

- Los hechos constituyen los datos del problema en forma de:
  - Base de hechos (BH)
  - Memoria de trabajo
  - Memoria a corto plazo
  - Aserciones
- Ejemplos:
  - $x$  es un gato
  - $x$  es un animal doméstico



# Motor de inferencia: terminología

- El motor de inferencia o mecanismo de control está compuesto de dos elementos:
  - **Interprete de reglas** o mecanismo de inferencia
    - Mecanismo de razonamiento que determina que reglas de la BC se pueden aplicar para resolver el problema
  - **Estrategia de control** o estrategia de resolución de conflictos
- **Función**
  - Ejecutar acciones para resolver el problema (objetivo) a partir de un conjunto inicial de hechos y eventualmente a través de una interacción con el usuario
  - La ejecución puede llevar a la deducción de nuevos hechos.

# Motor de inferencia

- Fases del ciclo básico:
  1. Detección (filtro): **Reglas pertinentes**
    - Obtención, de la BC, del conjunto de reglas aplicables a una situación determinada (estado) de la BH; formación del conjunto de conflictos (**interprete de reglas**)
  2. Selección: **¿Qué regla?**
    - Resolución de conflictos: selección de la regla a aplicar (**estrategia de control**)
  3. Aplicación
    - Aplicación de la regla sobre una instanciación de las variables: modificación de la memoria de trabajo
  4. Vuelta al punto 1, o parada si el problema está resuelto
    - Si no se ha encontrado una solución y no hay reglas aplicables: **fracaso**.

# 1. Detección

- Construcción del conjunto de reglas aplicables
- El intérprete de reglas realiza los cálculos necesarios para obtener las instanciaciones que son posibles en cada estado de resolución del problema (**comparación** o *matching*).
- **Una regla puede instanciarse más de una vez**, caso de existir variables que lo permitan.

# 2. Selección

- Las reglas son o no aplicadas dependiendo de la estrategia de control:
  - estrategia fija
  - estrategia dinámica prefijada
  - estrategia guiada por meta-reglas
- Selección de la “mejor” instanciación
- Ejemplos de estrategia de control:
  - 1ª regla por orden en la base de conocimiento
  - la regla más/menos utilizada
  - la regla más específica (con más literales) / más general
  - la regla que tenga el grado de certeza más alto
  - la instanciación que satisfaga los hechos:
    - más prioritarios
    - más antiguos (instanciación más antigua)
    - más nuevos (instanciación más reciente)
  - aplicación de todas las reglas (sólo si se quieren todas las soluciones posibles)
  - la regla usada más recientemente
  - meta-reglas, que indican dinámicamente como seleccionar las reglas a aplicar
- Posible combinación de criterios

# 3. Aplicación

- Ejecución de la regla  $\Rightarrow$ 
  - Modificación de la base de hechos (en el razonamiento hacia delante)
  - Nuevos cálculos, nuevas acciones, preguntas al usuario
  - Nuevos sub-objetivos (en el razonamiento hacia atrás)
- Propagación de las instanciaciones
- Propagación del grado de certeza

El proceso de deducción acaba cuando:

- se encuentra la conclusión (el objetivo) buscado  $\Rightarrow$  éxito
- no queda ninguna regla aplicable  $\Rightarrow$  éxito? / fracaso?

# Tipos de razonamiento

- Deductivo, progresivo, encadenamiento hacia delante, dirigido por hechos
  - evidencias, síntomas, datos  $\Rightarrow$  conclusiones
- Inductivo, regresivo, encadenamiento hacia atrás, dirigido por objetivos
  - conclusiones  $\Rightarrow$  datos, evidencias, síntomas
- Mixto, encadenamiento híbrido

# Encadenamiento hacia delante

- Basado en *modus ponens*:  $A, A \Rightarrow B \quad |- \quad B$
- La base de hechos (BH) se inicializa con los hechos conocidos inicialmente.
- Se obtienen las consecuencias derivables de la BH:
  - se comparan los hechos de la BH con la parte izquierda de las reglas; se seleccionan las reglas aplicables: las que tienen antecedentes conocidos (que están en la BH);
  - las nuevas conclusiones de las reglas aplicadas se añaden a la BH (hay que decidir cómo);
  - se itera hasta encontrar una condición de finalización.

# Encadenamiento hacia delante

- Problemas:
  - No focaliza en el objetivo
  - Explosión combinatoria
- Ventajas:
  - Deducción intuitiva
  - Facilita la formalización del conocimiento al hacer un uso natural del mismo
- Ejemplo de lenguaje: OPS-5
  - L. Brownston, R. Farrell, E. Kant, y N. Martin.  
*Programming Expert Systems in OPS-5*. Addison-Wesley, 1985.

# Encadenamiento hacia atrás

- Basado en el *método inductivo*: va guiado por un objetivo que es la conclusión (o hipótesis) que se trata de validar reconstruyendo la cadena de razonamiento en orden inverso.
- Cada paso implica nuevos sub-objetivos: hipótesis que han de validarse.
- Funcionamiento:
  - se inicializa la BH con un conjunto inicial de hechos;
  - se inicializa el conjunto de hipótesis (CH) con los objetivos a verificar;
  - mientras existan hipótesis a validar en CH se escoge una de ellas y se valida:
    - se comparan los hechos de la BH y la parte derecha de las reglas con las hipótesis;
    - si una hipótesis está en BH eliminarla de CH;
    - si no: buscar reglas que tengan como conclusión la hipótesis. Seleccionar una, añadir las premisas no satisfechas a CH como sub-objetivos.

# Encadenamiento hacia atrás

- El encadenamiento hacia atrás es un tipo de **razonamiento dirigido por el objetivo**.
- Sólo se considera lo necesario para la resolución del problema.
- El proceso de resolución consiste en la exploración de un árbol.
- Ejemplo de lenguaje: Prolog
  - W. F. Clocksin y C. S. Mellish . *Programming in Prolog: Using the ISO Standard*. Springer, 2003 (primera edición de 1981).

# Prolog: lenguaje de programación lógica

- Programa Prolog:
  - conjunto de **aserciones lógicas**
  - cada aserción es una **cláusula de Horn**
  - proceso de comparación: **unificación**
  - el **orden** de las aserciones (reglas) es **significativo**
- Ejemplo
  - gato (bufa).
  - animaldomestico (X) :- gato (X).
  - animaldomestico (X) :- pequeño (X), conplumas (X).
- $p \rightarrow q \Leftrightarrow q :- p$
- Consultas: ?- predicado.
- La negación se representa con la falta de la aserción correspondiente: asunción de **mundo cerrado**.

# Cláusula de Horn

## •De Wikipedia:

Las cláusulas de Horn (instrucciones ejecutables de [PROLOG](#)) tienen el siguiente aspecto:

hija (\*A, \*B) :- mujer (\*A), padre (\*B, \*A). que podría leerse así: "A es hija de B si A es mujer y B es padre de A".

Obsérvese que, en [PROLOG](#), el símbolo :- separa la conclusión de las condiciones. En [PROLOG](#), las variables se escriben comenzando con un asterisco. Todas las condiciones deben cumplirse simultáneamente para que la conclusión sea válida. Por tanto, la coma que separa las distintas condiciones es equivalente a la conjunción copulativa (en algunas versiones de [PROLOG](#) se sustituye la coma por el símbolo &). La disyunción, en cambio, no se representa mediante símbolos especiales, sino definiendo reglas nuevas, como la siguiente:

hija (\*A, \*B) :- mujer (\*A), madre (\*B, \*A). que podría leerse así: "A es hija de B si A es mujer y B es madre de A".

Cláusulas con como mucho un literal positivo.

# Encadenamiento híbrido

- Partes de la cadena de razonamiento que conduce de los hechos a los objetivos se construyen deductivamente y otras partes inductivamente: exploración bi-direccional
- El cambio de estrategia suele llevarse a cabo a través de meta-reglas. Ejemplos:
  - función del número de estados iniciales y finales
  - función de la dirección de mayor ramificación
  - función de la necesidad de justificar el proceso de razonamiento
- Se evita la explosión combinatoria del razonamiento deductivo.

# Comparación (matching)

- Es más complicada en el razonamiento hacia delante.
  - Si las condiciones de una regla se cumplen una vez aplicada se puede entrar en un ciclo.
- Existen mecanismos eficientes de comparación y selección que evitan repasar todas las reglas de la BC:
  - OPS-5 usa el algoritmo RETE
  - Prolog indexa las cláusulas según los predicados

# Factores para decidir el sentido del encadenamiento

- Número de estados iniciales y finales
  - Preferible del conjunto más pequeño hacia el más grande
- Factor de ramificación
  - Preferible en el sentido del factor más pequeño
- Necesidad de justificar el proceso de razonamiento
  - Preferible el sentido de razonamiento habitual del usuario

# Bibliografía complementaria

- J. F. Sowa. *Knowledge Representation*. Brooks/Cole, 2000.
- W. F. Clocksin y C. S. Mellish.  
*Programming in Prolog: Using the ISO Standard*. Springer, 2003 (primera edición de 1981).