Representación del conocimiento

Javier Béjar

Inteligencia Artificial - 2021/2022 1Q

CS - GEI- FIB

Introducción

- Las representación mediante formalismos lógicos puede verse de forma procedimental
- Describimos 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 habitualmente está formado por un predicado atomico o una acción
- Analogía con búsqueda en espacio de estados
  - Hechos = Estado del problema
  - Reglas = Operadores de búsqueda

Un problema quedará 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:** Ejecuta el formalismo y obtiene la cadena de razonamiento que soluciona el problema

El motor de inferencias

- Funciones
  - Deducir nuevos hechos, 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
- Componentes
  - Intérprete de reglas + estrategia de control
- Fases
  - Detección (filtro): **REGLAS PERTINENTES**
    - Obtención del conjunto de instanciaciones de reglas. Formación del **conjunto de conflictos**
  - Selección: **¿QUÉ REGLA?**
    - Resolución de conflictos: selección de la instanciación a aplicar
  - Aplicación:
    - Aplicación de la regla

- Construcción del conjunto de instanciaciones de reglas (**Conjunto de conflicto**)
- El intérprete de reglas realiza los cálculos e instanciaciones necesarias que son posibles en cada estado de resolución del problema (matching)
- Una regla puede instanciarse más de una vez, caso de existir variables (p.ej. CP1)

- Selección de la *mejor* instanciación
- Las reglas instanciadas son seleccionadas para aplicarse dependiendo de la estrategia de control (*Estrategia de resolución de conflictos*)
  - estrategia fija
  - estrategia dinámica prefijada
  - estrategia guiada por meta-reglas
- Criterios aplicables:
  - 1ª regla por orden en la Base de Conocimientos
  - la regla más/ menos utilizada
  - la regla más específica/la 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), ...

- Ejecución de la regla $\Rightarrow$
  - Modificación de la base de hechos
  - Nuevos cálculos, nuevas acciones, preguntas al usuario
  - Nuevos subobjetivos
- Propagación de las instanciaciones (en CP1)
- 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?

Razonamiento

- Deductivos, progresivos, forward chaining (FC), encadenamiento dirigido por hechos
  - evidencias, síntomas, datos $\Rightarrow$ conclusiones, hipótesis
- Inductivos, regresivos, backward chaining (BC), encadenamiento dirigido por objetivos
  - conclusiones, hipótesis $\Rightarrow$ datos, evidencias, síntomas
- Mixtos, encadenamiento híbrido

- Basado en modus ponens: $A, A \rightarrow B \vdash B$
- La base de hechos (BH) se inicializa con los hechos que describen el problema

---

Procedure: Razonamiento Hacia Adelante

Input: Base de hechos, Base de reglas, Objetivos

Alternativas ← cierto

while $\exists o(o \in Objetivos \wedge o \notin Base\_de\_hechos) \wedge Alternativas$ **do**

    Conjunto_Conflicto ← Interprete.Antecedentes_satisfactibles(Base_de_hechos, Base_de_reglas)

    if *Conjunto_Conflicto* $\neq \emptyset$ **then**

        Regla ← Estrategia_Control.Resolucion_Conflictos(Conjunto_Conflicto)

        Interprete.Aplicar(Base_de_hechos, Regla)

    else

        Alternativas ← falso

- Problemas:
  - La búsqueda no está localizada en el objetivo
  - Explosión combinatoria, deducimos hechos no relacionados con la solución
- Ventajas:
  - Deducción intuitiva
  - Facilita la formalización del conocimiento al hacer un uso natural del mismo
  - Puede ser usado de manera exploratoria

- Método Inductivo. A partir de la hipótesis inicial se reconstruye la cadena de razonamiento en orden inverso hasta los hechos
- Cada paso implica nuevos subobjetivos: hipótesis a validar

Procedure: Razonamiento Hacia Atrás

Input: Base de hechos, Base de reglas, Objetivos

```
Alternativas ← cierto
while Objetivos ≠ ∅ ∧ Alternativas do
    Objetivo ← Estrategia_Control.Escoger_Objetivo(Objetivos)
    Objetivos.Quitar(Objetivo)
    Conjunto_Conflicto ← Interprete.Consecuentes_satisfactibles(Objetivo, Base_de_reglas)
    if Conjunto_Conflicto ≠ ∅ then
        Regla ← Estrategia_Control.Resolucion_Conflictos(Conjunto_Conflicto)
        Objetivos.Añadir(Regla.Extraer_antecedente_como_objetivos())
    else
        Alternativas ← falso
```

- La resolución se plantea como una descomposición en subproblemas
- El proceso de resolución consiste en la exploración de un árbol Y/O
- Ventajas
  - Sólo se considera lo necesario para la resolución del problema
- Desventajas
  - Hemos de conocer la solución del problema a priori

- Partes de la cadena de razonamiento que conduce de los hechos a los objetivos se construyen deductivamente y otras inductivamente
- Exploración bidireccional
- El cambio de estrategia suele llevarse a cabo a través de meta-reglas
- Se evita la explosión combinatoria del razonamiento deductivo
- Mejora la eficiencia del razonamiento inductivo cuando no existen objetivos claros

El uso de reglas como mecanismo de programación está muy extendido

- Como mecanismo de transformación, compilación, traducción, ...
  - Compiladores de lenguajes (LEX, YACC)
  - En la Web: XLTS (Extensible Stylesheet Language Transformations)
  - Automatización de tareas: Make, ANT, ...
- Como representación de **reglas de negocio** en aplicaciones
  - Motores de inferencia como parte del desarrollo de aplicaciones: Reglas interpretadas en lugar de código
  - Muchas herramientas de desarrollo comercial las incluyen: SAP, IBM, Oracle, Microsoft, ...
  - Forma parte de la filosofía de paradigmas de desarrollo de aplicaciones (Service Oriented Architectures)