

Práctica de Sistemas Basados en el Conocimiento

Recomendación de Viviendas en Alquiler

Grupo XX
Nombres de los integrantes

13 de diciembre de 2025

1. Introducción y planteamiento del problema

1.1. Estructura del documento

Este documento se organiza en tres partes principales, siguiendo las directrices de la rúbrica de evaluación:

- **Parte 1 (Introducción):** Planteamiento del problema, objetivos y metodología.
- **Parte 2 (Desarrollo):** Conceptualización, formalización (ontología) e implementación en CLIPS.
- **Parte 3 (Resultados):** Casos de prueba, análisis de resultados, conclusiones y trabajo en equipo.

1.2. Contexto y motivación

La práctica plantea el desarrollo de un sistema de recomendación de viviendas en alquiler que pueda ser utilizado por una entidad pública para ayudar a los ciudadanos a encontrar ofertas adecuadas a sus necesidades.

El problema implica combinar dos tipos de información:

- **Características de las viviendas:** precio, superficie, tipo de propiedad, amenities, ubicación, etc.
- **Perfil y preferencias del usuario:** presupuesto, composición familiar, necesidades de proximidad a servicios, restricciones, etc.

La complejidad del dominio y la necesidad de aplicar conocimiento experto para ponderar múltiples factores convierten este problema en un caso idóneo para un Sistema Basado en el Conocimiento (SBC).

1.3. Objetivos de la práctica

El objetivo principal de esta práctica es diseñar e implementar un **Sistema Basado en el Conocimiento (SBC)** que emule el razonamiento de un experto inmobiliario para recomendar viviendas en alquiler. Los objetivos específicos son:

- Analizar el dominio del problema e identificar las fuentes de conocimiento necesarias.
- Construir una ontología formal que represente los conceptos clave: **viviendas, solicitantes, servicios urbanos y zonas de la ciudad**.
- Implementar en CLIPS un sistema de reglas que realice:
 - Evaluación de compatibilidad entre perfil del solicitante y características de la vivienda.
 - Cálculo de proximidad a servicios relevantes (transporte, educación, ocio, salud, etc.).
 - Clasificación de las recomendaciones en tres categorías: *parcialmente adecuada, adecuada y muy recomendable*.
- Validar el sistema mediante casos de prueba representativos y documentar el proceso de desarrollo siguiendo la metodología de ingeniería del conocimiento.

1.4. Metodología de desarrollo

Para abordar este problema, se ha seguido la metodología de **Ingeniería del Conocimiento** explicada en clase, estructurada en las siguientes fases:

1. **Identificación:** Análisis del enunciado, definición de objetivos y alcance.
2. **Conceptualización:** Extracción y organización del conocimiento experto, identificación de conceptos y relaciones.
3. **Formalización:** Construcción de la ontología en Protégé y diseño de la estrategia de razonamiento.
4. **Implementación:** Codificación en CLIPS del sistema de reglas y hechos.
5. **Prueba:** Validación mediante casos de prueba y análisis de resultados.

2. Desarrollo de la solución

2.1. Conceptualización

2.1.1. Extracción de conocimiento

Si se usó modelo de lenguaje: contexto dado, preguntas realizadas, conocimiento extraído. Si no, explicar cómo se obtuvo el conocimiento (sentido común, investigación).

2.1.2. Conceptos y relaciones principales

A partir del análisis del dominio y de la experiencia en portales inmobiliarios, identificamos los siguientes conceptos clave, que posteriormente se formalizarán en la ontología:

- **Cliente**: persona o grupo que busca vivienda. Se distingue entre:
 - **Individual**: Adulto, Joven, Estudiante, Anciano.
 - **Grupo**: Pareja (con o sin hijos previstos, joven o anciana), Familia (con o sin ancianos), Grupo de estudiantes.
- **Vivienda**: oferta de alquiler. Incluye subtipos como:
 - **Piso, Dúplex, Ático, Estudio**.
 - **Casa individual, Casa pareada, Casa adosada**.
 - **Habitación** para alquiler individual.
- **Servicio**: elemento urbano que influye en la calidad de vida:
 - **Transporte**: Parada de autobús, metro, tren, aparcamiento.
 - **Educación**: Colegio, instituto, universidad.
 - **Salud**: Clínica, hospital, farmacia.
 - **Comercio**: Supermercado, hipermercado, centro comercial, tienda.
 - **Ocio**: Parque, piscina, gimnasio, cine, museo, zona de vida nocturna, restaurante, playa, polideportivo.
- **Zona**: área de la ciudad con características homogéneas:
 - **Residencial, Centro, Núcleo urbano, Negocios, Industrial**.

Relaciones principales identificadas:

- Una **Vivienda** *está ubicada en* una **Zona** (relación `is_located_in_zone`).
- Una **Vivienda** *está cerca de* uno o más **Servicios** (relación `is_near_service`).
- Un **Cliente** *tiene preferencias y restricciones* sobre:
 - Precio máximo, flexibilidad presupuestaria, precio mínimo sospechoso (*too bargain*).
 - Número mínimo de dormitorios y baños.
 - Necesidad de transporte público, posesión de coche.
 - Ubicación de trabajo/estudio (coordenadas).
 - Mascotas, ruido tolerado, características deseadas (terraza, ascensor, amueblado, etc.).
- Una **Vivienda** *puede satisfacer o no* dichas preferencias, lo que determina su **grado de recomendación** (parcialmente adecuada, adecuada, muy recomendable).

2.1.3. Descomposición en subproblemas

Explicar cómo se divide el problema global en tareas más pequeñas:

- Subproblema 1: Captura de preferencias del usuario.
- Subproblema 2: Evaluación de viviendas respecto a preferencias.
- Subproblema 3: Cálculo de distancias a servicios.
- Subproblema 4: Clasificación y recomendación final.

2.2. Formalización

2.2.1. Ontología del dominio

La ontología se ha construido en Protégé siguiendo la metodología de desarrollo de ontologías 101, y está compuesta por 4 conceptos raíz principales, sus jerarquías de subclases, propiedades de objeto y datos, y restricciones de dominio/rango.

Jerarquía de clases

- **Client_Group**: representa a los solicitantes. Se divide en:
 - Individual: Adult, Young, Student, Elderly.
 - Group: Couple (Elderly_Couple, Planning_Kids, Young_No_Kids), Family (No_Elderly, With_Elderly), Student_Group.

Se ha modelado así para capturar perfiles con necesidades diferenciadas (ej: jóvenes buscan ocio, familias buscan escuelas).

- **Property**: representa las viviendas en alquiler. Incluye:
 - Tipos urbanos: Apartment, Duplex, Penthouse, Studio, Triplex, Room.
 - Tipos unifamiliares: Detached_House, Semidetached_House, Row_House, Single-family_House.

Se han definido como *disjoint* para evitar inconsistencias.

- **Service**: modela los servicios urbanos. Subclases:
 - Transport (Bus_stop, Metro_station, Train_station, Parking, Cable_car_station).
 - Education (School, High_School, University).
 - Healthcare (Clinic, Hospital, Pharmacy).
 - Shopping (Supermarket, Hipermarket, Mall, Store).
 - Recreation (Park, Pool, Gym, Beach, Cinema, Museum, Nightlife, etc.).

Esta categorización permite evaluar la proximidad a servicios de forma estructurada.

- **Zone**: describe zonas de la ciudad:
 - Residential, Business, Downtown, Industrial, Urban_core.

Cada zona tiene atributos como nivel de ruido y seguridad.

Propiedades de objeto

- `is_located_in_zone`: relaciona una `Property` con una `Zone`. Es funcional (cada vivienda está en una sola zona).
- `is_near_service`: relaciona una `Property` con un `Service`. Permite modelar proximidad a múltiples servicios.

Propiedades de datos Se han agrupado en tres categorías para facilitar la modularidad:

- `property_attribute`: atributos de la vivienda (precio, metros², habitaciones, terraza, ascensor, amueblado, etc.).
- `client_attribute`: atributos del solicitante (edad, presupuesto, preferencias, ubicación trabajo/estudio, etc.).
- `zone_attribute`: atributos de la zona (nivel de ruido, seguridad).

Se han usado tipos de datos específicos (`boolean`, `int`, `float`) y enumerados (ej: `Sun_Time`: `Morning`, `Afternoon`, `All_Day`, `Never`).

Decisiones de diseño justificadas

- **Coordenadas enteras**: se usan `geo_lat` y `geo_long` como enteros para simplificar el cálculo de distancias en CLIPS.
- `floor_level` solo aplica a `Apartment`, `Room` y `Studio`, no a viviendas unifamiliares.
- `Too_Bargain`: se incluye como atributo del cliente para detectar precios sospechosamente bajos.
- *Disjoint* entre clases: evita que una misma instancia sea, por ejemplo, un `Apartment` y una `Detached_House`.

2.2.2. Metodología de resolución de problemas

Explicar qué metodología se ha elegido (p. ej., clasificación, razonamiento basado en casos) y cómo los subproblemas identificados encajan en ella.

2.2.3. Diseño del Sistema de Reglas

El sistema implementa tres tipos principales de reglas que permiten transformar los datos concretos del dominio en conocimiento abstracto para la toma de decisiones. Estas reglas siguen la metodología de resolución de problemas descrita anteriormente.

2.2.3.1 Reglas de Abstracción Estas reglas realizan la transición del **problema concreto** al **problema abstracto**, transformando datos brutos en atributos categóricos que permiten una evaluación sistemática. Se dividen en tres subtipos:

Reglas de Abstracción Implícita: La información ya se encuentra en formato abstracto cuando el usuario la proporciona. No se requiere procesamiento adicional.

- **Ejemplo:** Cuando el usuario responde “*¿Tiene coche propio?*” con `yes`, el sistema asigna directamente `yes` al atributo `owns_car`.
- **Otros ejemplos:** `has_pets`, `Budget_Is_Strict`, `Works_In_City`.

Reglas de Abstracción Inducida: Se infieren preferencias a partir de información demográfica, basándose en conocimiento experto sobre patrones comunes.

- **Ejemplo:** Si el cliente es clasificado como `Elderly`, `Elderly_Couple` o `With_Elderly`, el sistema induce preferencias como `"elevator"`, `"quiet"`, `"single-floor"`, `"healthcare"`.
- Estas preferencias se añaden automáticamente a `desired_features`.

Reglas de Abstracción Explícita: Transforman datos numéricos o geográficos en categorías simbólicas mediante fórmulas y umbrales predefinidos. Estas reglas realizan cálculos concretos que permiten evaluar características espaciales y cuantitativas de las propiedades.

- **Cálculo de densidad de habitaciones (room-density):** Para determinar si una propiedad tiene espacio suficiente para el grupo de clientes, el sistema calcula:

$$\text{capacidad} = (\text{Num_Double_Rooms} \times 2) + \text{Num_Single_Rooms}$$

$$\text{ratio} = \frac{\text{capacidad}}{\text{num_people}}$$

La clasificación resultante es:

- **saturated** ($\text{ratio} < 1.0$): insuficiente espacio
 - **sufficient** ($\text{ratio} = 1.0$): espacio justo
 - **spacious** ($\text{ratio} > 1.0$): espacio abundante
- **Cálculo de distancias euclidianas:** Todas las mediciones de proximidad se basan en la función `euclid-dist`, que calcula la distancia en línea recta entre dos puntos geográficos:

$$\text{distancia}(x_1, y_1, x_2, y_2) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Donde (x_1, y_1) y (x_2, y_2) son las coordenadas enteras de los puntos a comparar.

- **Asignación de servicios cercanos:** Aplicando la función `euclid-dist`, el sistema clasifica la proximidad de cada servicio a una propiedad mediante dos reglas:
 - `inicio-calcular-cercania`: servicio ≤ 200 m \rightarrow `is_near_service`
 - `inicio-calcular-media-cercania`: servicio entre 200-700 m \rightarrow `is_semi_near_service`

Estas reglas se ejecutan al inicio del sistema con prioridad alta (`salience` 100 y 99) para garantizar que todas las conexiones se establezcan antes del proceso de recomendación.

- **Cálculo de distancias a trabajo/estudio para grupos:** Cuando múltiples personas trabajan o estudian en diferentes ubicaciones, el sistema implementa una estrategia conservadora considerando la **peor situación**. Para cada propiedad:

1. Se calcula la distancia entre la propiedad y cada ubicación de trabajo/estudio usando `euclid-dist`.
2. Se determina la **distancia máxima** mediante la función `max-distance`, que itera sobre todas las coordenadas proporcionadas.
3. Esta distancia máxima se clasifica según umbrales predefinidos:
 - <1000 m → `cerca`
 - 1000-3500 m → `media_distancia`
 - >3500 m → `lejos`

Justificación del diseño: Esta aproximación garantiza que la recomendación sea válida para todos los miembros del grupo, incluso para quien tenga el trayecto más largo. Si la propiedad satisface la distancia máxima, automáticamente satisface las distancias menores de otros miembros.

- **Función de clasificación de distancias:** La función auxiliar `classify-distance` encapsula la lógica de umbrales, permitiendo una modificación centralizada de los criterios de proximidad si fuera necesario en futuras versiones del sistema.

2.2.3.2 Reglas de Evaluación de Criterios Una vez obtenidos los atributos abstractos, el sistema evalúa cada propiedad contra los criterios del cliente. Estas reglas comparan características específicas y generan evaluaciones cualitativas.

Ejemplos de criterios evaluados:

- **Compatibilidad presupuestaria:** Compara `monthly_price` de la propiedad con `max_budget` del cliente.
- **Compatibilidad con mascotas:** Verifica `Pets_Allowed` de la propiedad contra `has_pets` del cliente.
- **Densidad adecuada:** Evalúa `room-density` para determinar si hay suficiente espacio.
- **Distancia a servicios clave:** Comprueba proximidad a trabajo/estudio según categorías `cerca`, `media_distancia`, `lejos`.
- **Satisfacción de características deseadas:** Para cada elemento en `desired_features`, verifica si la propiedad lo cumple mediante la función `feature-satisfied`.

Salida de estas reglas: Cada propiedad recibe dos listas:

- **met-criteria:** Criterios que satisface.
- **failed-criteria:** Criterios que no satisface.
- **extra-criteria:** Características adicionales positivas no solicitadas explícitamente.

2.2.3.3 Reglas de Clasificación y Recomendación Finalmente, el sistema clasifica cada propiedad en una de cuatro categorías de recomendación basándose en los resultados de la evaluación.

Lógica de clasificación implementada:

1. **No recomendado:** Si la propiedad falla en más de 3 criterios esenciales.
2. **Parcialmente adecuada:** Si falla en 1-3 criterios, pero satisface los esenciales.
3. **Adecuada:** Si satisface todos los criterios sin características extras destacables.
4. **Muy recomendable:** Si satisface todos los criterios y además tiene características extras valoradas (ej: estado excelente, parque cercano, terraza, piscina).

Estrategia de implementación: La regla `classify-property` analiza las listas `failed-criteria` y `extra-criteria` para asignar la categoría final. Esta clasificación permite presentar al usuario un ranking cualitativo de las propiedades disponibles.

Ejemplo de salida:

```
Propiedad apt-101 -> muy-adecuado
Pros: [entra-en-presupuesto, admite-mascotas, ...]
Contras: ninguno
Extras: [estado-excelente, parque-cerca]
```

2.3. Implementación en CLIPS

2.3.1. Estructura del proyecto

Organización de módulos, templates, hechos iniciales.

2.3.2. Ejemplo de reglas clave

Fragmentos de código comentados que ilustren el razonamiento.

```
(defrule evaluar-precio
  ?s <- (solicitante (precio-max ?pmax) (flexible ?flex))
  ?v <- (vivienda (precio ?p) (id ?id))
  (test (<= ?p ?pmax))
  =>
  (assert (candidata (id ?id) (motivo "Dentro de presupuesto"))))
```

Listing 1: Ejemplo de regla en CLIPS

2.3.3. Desarrollo incremental

Descripción de los prototipos desarrollados, evolución desde el primer prototipo funcional hasta la versión final.

3. Resultados y validación

3.1. Juegos de prueba

Explicación de cómo se seleccionaron los casos: perfiles variados (estudiante, familia, pareja), casos límite, cobertura de funcionalidades.

3.2. Ejecución de pruebas

Tabla resumen con al menos 6 casos diferentes, mostrando:

- Perfil de entrada.
- Viviendas recomendadas (adecuadas, parcialmente adecuadas, muy recomendables).
- Criterios incumplidos o características destacadas.

Cuadro 1: Ejemplo de casos de prueba

Perfil	Vivienda	Grado	Observaciones
Estudiante, precio bajo	Atico céntrico	Parcial	Sin terraza
Familia con hijos	Dúplex con jardín	Muy recomendable	Cerca de colegio y parque

3.3. Análisis de resultados

Explicación de por qué el sistema toma ciertas decisiones, ejemplos concretos basados en el conocimiento codificado.

3.4. Conclusiones

3.4.1. Logros y dificultades

Qué se ha conseguido, qué partes fueron más complejas, limitaciones del sistema.

3.4.2. Trabajo en equipo

Breve descripción de cómo se organizó el grupo, reuniones, reparto de tareas, uso de la planificación propuesta.

3.4.3. Posibles mejoras

Extensiones futuras: integración con datos reales de mapas, más factores de ponderación, interfaz gráfica.

A. Anexo A: Ontología completa

Gráfico exportado desde Protégé con la jerarquía de clases y propiedades.

B. Anexo B: Diálogo con modelo de lenguaje

Si aplica: contexto usado, preguntas más relevantes y respuestas resumidas.

C. Anexo C: Código fuente completo

Listado completo del código CLIPS (o enlace al repositorio).