Unidad IV: Fundamentos de la programación lógica

4.1. Repaso de la lógica de primer orden

La programación lógica es un tipo de paradigmas de programación dentro del paradigma de programación declarativa. El resto de los subparadigmas de programación dentro de la programación declarativa son: programación funcional, programación con restricciones, programas DSL (de dominio específico) e híbridos. La programación funcional se basa en el concepto de función (que no es más que una evolución de los predicados), de corte más matemático. La programación lógica gira en torno al concepto de predicado, o relación entre elementos.

4.2. Unificación y resolución

Para probar la existencia de algo:

Suponer lo opuesto y usar modus ponens y la regla de eliminación del cuanticador universal, para encontrar un contra ejemplo al supuesto.

4.3. Cláusulas de Horn. Resolución SLD

Seleccionar una literal, usando una estrategia **L**ineal, restringido a cláusulas **D**efinitivas

Un caso especial de resolución lineal

Resolución lineal: el último resolvente se toma como cláusula padre

La otra cláusula padre se toma de otro resolvente o del conjunto original

Una forma especial de resolución lineal es: *input resolution*. En esta estrategia, cada paso de resolución, exceptuando el primero, se toma del último resolvente (cláusulas metas) y del conjunto original (cláusulas de entrada)

Input resolution es *completa* para cláusulas de Horn, pero no para cláusulas en general

Una variante de *resolución de entrada* es *resolución SLD* para cláusulas de Horn. Resolución de entrada se extiende con una regla de selección que determina en cada paso que literal de la cláusula meta es seleccionada.

e.g., $\{R(g(x) \leftarrow T(x, y, f(y)), T(a, b, f(a)),$
$P(v, w) \leftarrow R(v)\}$ $\{\leftarrow P(u, b)\}$, Meta:

Resolución SLD es *sound* y *complete* para cláusulas de Horn

La estrategia de búsqueda afecta el resultado

e.g., depth-first con diferente orden de cláusulas:
$C_1 = P(x) \leftarrow P(f(x))$

$C_2 = P(f(f(a))) \leftarrow$

$\leftarrow P(a)$
Meta:

Aunque resolución SLD es *sound* y *complete* para cláusulas de Horn, en la práctica (por razones de eficiencia) se hacen variantes

- eliminar el ``occur check'' de unificación
- usar un orden específico

4.4. Programación lógica con cláusulas de Horn

En lógica proposicional, una fórmula lógica es una **cláusula de Horn** si es una cláusula (disyunción de literales) con, como máximo, un literal positivo. Se llaman así por el lógico Alfred Horn, el primero en señalar la importancia de estas cláusulas en 1951.

Esto es un ejemplo de una cláusula de Horn:

$$\neg p \vee \neg q \vee \cdots \vee \neg t \vee u$$

Una fórmula como esta también puede reescribirse de forma equivalente como una implicación:

$$(p \wedge q \wedge \cdots \wedge t) \rightarrow u$$

Una cláusula de Horn con exactamente un literal positivo es una cláusula "definite"; en álgebra universal las cláusulas "definites" resultan (aparecen) como cuasi-identidades. Una cláusula de Horn sin ningún literal positivo es a veces llamada cláusula objetivo (goal) o consulta (query), especialmente en programación lógica.

Una fórmula de Horn es una cadena textual (string) de cuantificadores existentiales o universales seguidos por una conjunción nde cláusulas de Horn.

4.5. Semántica de los programas lógicos

Observen que resoluci´on (incluso la SLD) no prescribe una forma particular deresponder estas preguntas :

• ¿Qué átomo en una cl´ausula objetivo debemos usar en la resolución?

• ¿Cuál cláusula definida se debe usar para la resolución?

4.6. Representación clausada del conocimiento

La representación del conocimiento y el razonamiento es un área de la inteligencia artificial cuyo objetivo fundamental es representar el conocimiento de una manera que facilite la inferencia (sacar conclusiones) a partir de dicho conocimiento. Analiza cómo pensar formalmente - cómo usar un sistema de símbolos para representar un dominio del discurso (aquello de lo que se puede hablar), junto con funciones que permitan inferir (realizar un razonamiento formal) sobre los objetos. Generalmente, se usa algún tipo de lógica para proveer una semántica formal de como las funciones de razonamiento se aplican a los símbolos del dominio del discurso, además de proveer operadores como cuantificadores, operadores modales, etc. Esto, junto a una teoría de interpretación, dan significado a las frases en la lógica.

Cuando diseñamos una representación del conocimiento (y un sistema de representación del conocimiento para interpretar frases en la lógica para poder derivar inferencias de ellas) tenemos que hacer elecciones a lo largo de un número de ámbitos de diseño. La decisión más importante que hay que tomar es la *expresividad* de la representación del conocimiento. Cuanto más expresiva es, decir algo es más fácil y más compacto. Sin embargo, cuanto más expresivo es un lenguaje, más difícil es derivar inferencias automáticamente de él. Un ejemplo de una representación del conocimiento poco expresiva es la lógica proposicional. Un ejemplo de una representación del conocimiento muy expresiva es la lógica autoepistémica. Las representaciones del conocimiento poco expresivas pueden ser tanto completas como consistentes (formalmente menos expresivas que la teoría de conjuntos). Las representaciones del conocimiento más expresivas pueden ser ni completas ni consistentes.

El principal problema es encontrar una representación del conocimiento y un sistema de razonamiento que la soporte que pueda hacer las inferencias que necesite tu aplicación dentro de los límites de recursos del problema a tratar. Los desarrollos recientes en la representación del conocimiento han sido liderados por la web semántica, y han incorporado el desarrollo de lenguajes y estándares de representación del conocimiento basados en XML, que incluyen Resource Description Framework (RDF), RDF Schema, DARPA Agent Markup Language (DAML), y Web Ontology Language (OWL).

4.7. Consulta de una base de cláusulas

4.8. Espacios de búsqueda

4.9. Programación lógica con números, listas y árboles.

4.10. Control de búsqueda en programas lógicos

4.11. Manipulación de términos. Predicados metaógicos.