Diseño y Construcción de Ontologías

Inferencia Lógica

Dra. Maricela Bravo
mcbc@correo.azc.uam.mx

UNIVERSIDAD AUTONOMA METROPOLITANA
Casa abierta al tiempo Azcapotzalco

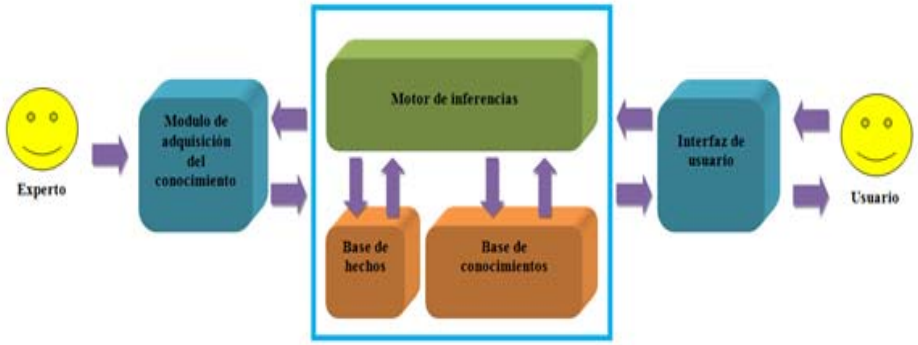
INFERENCIA LÓGICA

Inferencia Lógica

- Los sistemas basados en reglas fueron desarrollados por Newell y Simmon (1973).
- Este tipo de representación de conocimiento se relaciona más con las heurísticas y formas de proceder de los expertos en contraposición de las representaciones declarativas de los esquemas anteriores (Alonso y colaboradores, 2004).
- La arquitectura de este tipo de representación de conocimiento generalmente consta de una base de hechos, una base de reglas y un motor o maquina de inferencia.

3

Inferencia Lógica



4

Procesos de Inferencia

- Estado Inicial: Representa la situación original del problema
- Estados finales: Representan las situaciones y objetivos que se pretenden alcanzar.
- Estados Intermedios o memoria de trabajo: Contiene la descripción de la solución en curso de resolución. La ejecución de las reglas modifica este estado y añade o elimina hechos.

5

Procesos de Inferencia

- La base de reglas contiene la parte fundamental de la base de conocimiento y las reglas describen los elementos de deducción básico que utilizara el sistema. La forma más sencilla de representación es del tipo

"*SI condiciones Entonces acciones*".

6

Reglas de SWRL

- Contiene:
  - Cuerpo: parte antecedente
  - Cabeza: parte consecuente
- Tanto el cuerpo y la cabeza consisten solamente de conjunciones positivas de los átomos:

  atom ^ atom .... -> atom ^ atom

- Si todos los átomos en el antecedente son verdaderos, entonces el consecuente también debe ser verdadero.
- SWRL no es compatible con los átomos negados o disyunción.

7

Reglas de SWRL

- Un átomo es una expresión de la forma:

  p(arg1, arg2, ... argn)

Donde:

p es un símbolo de predicado y *arg1*, *arg2*, ..., *argn* son los términos o argumentos de la expresión.

- Los símbolos de predicados pueden incluir clases, propiedades o tipos de datos.
- Los argumentos pueden ser individuos o valores de datos, o variables refiriéndose a éstos.

8

Tipos de Átomos en el SWRL

Átomos de Clase:

- Consiste de una clase nombrada en OWL o una expresión de clase y un argumento único que representa a un individuo OWL.

Person(?p)
Man(Fred)

- *Person* y *Man* son clases nombradas.
- ?p es una variable que representa un individuo.
- Fred es el nombre de un individuo.

9

Tipos de Átomos en el SWRL

- Un ejemplo sencillo del uso de reglas utilizando átomos de clase es declarar que un individuo del tipo *Man* es también individuo de la clase *Person*.

Man(?p) -> Person(?p)

- Una forma sencilla de establecer esto es hacerlo directamente en OWL.

10

Átomos de Propiedades Individuales

- Consiste de un objeto de propiedad OWL y dos argumentos que representan individuos OWL.

hasBrother(?x, ?y)
hasSibling(Fred, ?y)

- *hasBrother* y *hasSibling* son propiedades de objetos OWL.
- ?x y ?y son variables que representan individuos OWL.
- Fred es el nombre de un individuo.

11

Átomos de Propiedades Individuales

- Ejemplo de una regla SWRL:
  - Una persona con la propiedad "*tiene un hermano*" requeriría capturar los conceptos de persona, femenino, hermano de y hermana de en OWL.
  - El concepto de persona y masculino pueden capturarse utilizando una clase llamada *Person* con una subclase *Man*, las relaciones de hermana y hermano pueden expresarse utilizando propiedades de objetos de OWL *hasSibling* y *hasBrother* con un dominio y rango de *Person*.

Person(?p) ^ hasSibling(?p,?s) ^ Man(?s) ->hasBrother(?p,?s)

12

Átomos de Propiedades de Valores de Datos

- Consiste de una propiedad de datos OWL y dos argumentos:
  - El primero representando un individuo OWL
  - El segundo un valor de dato.
- Ejemplos:

hasAge(?x, ?age)
hasHeight(Fred, ?h)
hasAge(?x, 232)
hasName(?x, "Fred")

13

Átomos de Propiedades de Valores de Datos

Donde:

- *hasHeight*, *hasAge*, y *hasName* son propiedades de datos en OWL.
- ?x es una variable que representa un individuo OWL.
- Fred es el nombre de un individuo OWL
- ?h y ?age son variables que representan valores de datos.

14

Átomos de Propiedades de Valores de Datos

- Ejemplo de una propiedad de dato boleano:
  - Todas las personas que poseen un auto deben ser clasificadas como conductores.

Person(?p) ^ hasCar(?p, true) -> Driver(?p)

- Esta regla clasifica a todos los individuos que son personas que tienen un automóvil como miembros de la clase conductor.

15

Átomos de Propiedades de Valores de Datos

- Los individuos nombrados en una ontología también pueden ser referidos directamente.
- Ejemplo:
  - "*individuo llamado Fred que es clasificado como conductor*"

Person(Fred) ^ hasCar(Fred, true) -> Driver(Fred)

- Es importante hacer notar que esta regla trabaja con un individuo conocido llamado Fred en la ontología.
- No es posible crear nuevos individuos utilizando reglas de esta forma.

16

Átomos de Individuos Diferentes

- Consiste del símbolo *differentFrom* y dos argumentos que representan individuos OWL.

Ejemplo:

differentFrom(?x, ?y)

differentFrom(Fred, Joe)

- Las variables ?x y ?y representan individuos OWL
- Fred y Joe son nombres de individuos OWL.

17

Átomos de Individuos Iguales

- Consiste del símbolo *sameAs* y dos argumentos que representan individuos OWL.
- Ejemplo:

  sameAs(?x, ?y)

  sameAs(Fred, Freddy)

- Las variables ?x y ?y representan individuos OWL.
- Fred y Joe son nombres de individuos OWL.

18

Átomos Incorporados (Built-in)

- Es un predicado que toma uno o más argumentos y resulta verdadero si los argumentos satisfacen el predicado.
- Ejemplo:

  Un built-in de igualdad puede definirse para que acepte dos argumentos y devuelva verdadero si los argumentos son iguales.
- SWRL Built-in ofrece una gran variedad de funciones matemáticas y operaciones con cadenas.

19

Átomos Incorporados (Built-in)

- Ejemplo: "*Indica que una persona con edad mayor a 17 es un adulto*".

Person(?p) ^ hasAge(?p, ?age) ^ swrlb:greaterThan(?age, 17) -> Adult(?p)

- Por convención, los built-ins son precedidos por el espacio de nombres swrlb.
- Cuando es ejecutada, esta regla clasificará a los individuos de la clase *Person* que tengan un valor en la propiedad *hasAge* que sea mayor a 17 como miembros de la clase *Adult*.

20

Átomos Incorporados (Built-in)

- Ejemplo: "*Reglas SWRL built-ins para determinar si el número telefónico de una persona comienza con el código de acceso internacional '+'*".

Person(?p) ^ hasNumber(?p, ?number) ^
swrlb:startsWith(?number, "+")
-> hasInternationalNumber(?p, true)

- Los built-ins pueden tomar cualquier número de combinaciones de valores de tipos de datos en OWL.
- Los tipos de datos pueden ser valores de objetos, clases o valores de propiedades.

21

Asignación de Valores a los Argumentos

- La especificación de built-ins de SWRL permite asignar valores directamente a los argumentos.
- Ejemplo:

swrlb:add(?x, 2, 3)

  - El método *add* suma números enteros. Si x no tiene un valor asignado tomará el valor de 5. Si x ya tiene un valor asignado, entonces simplemente determinará si su valor es de 5.

- Un método built-in que exitosamente asigna un valor a un argumento debe devolver true de lo contrario regresa falso.

22

Asignación de Valores a los Argumentos

- Ejemplo: "*Regla que utiliza built-ins para asignar el cálculo del área de un rectángulo*".

```
Rectangle(?r) ^ hasWidthInMeters(?r, ?w) ^ hasHeightInMeters(?r, ?h) ^
swrlb:multiply(?areaInSquareMeters, ?w, ?h) ->
hasAreaInSquareMeters(?r, ?areaInSquareMeters)
```

- Debido a que la variable areaInSquareMeters no tiene valor asignado cuando el método multiply es llamado, el built-in le asignará un valor de tal forma que el built-in se evalúe a true.

23

Asignación de Valores a los Argumentos

- Si la variable ya tiene valor asignado cuando el built-in es ejecutado, su valor será utilizado directamente en la evaluación del predicado.
  - Ejemplo: "*Regla similar para clasificar un rectángulo con un área mayor a 100 metros cuadrados como un BigRectangle*".

```
Rectangle(?r) ^ hasWidthInMetres(?r, ?w) ^ hasHeightInMetres(?r,
?h) ^ swrlb:multiply(?areaInSquareMeters, ?w, ?h) ^
swrlb:greaterThan(100, ?areaInSquareMeters) ->
hasAreaInSquareMeters(?r, ?areaInSquareMeters) ^
BigRectangle(?r)
```

24

Asignación de Valores a los Argumentos

- Las variables no asignadas pueden ocurrir en cualquier posición de argumentos.
  - Ejemplo: "*Reglas que convierten el salario de una persona de pesos a dólares*".

Persona(?p), tieneSalarioEnPesos(?p, ?pesos), divide(?dollars, ?pesos , 18.50) -> tieneSalarioEnDolares(?p, ?dollars)

- En estos dos casos, la variable dólares se encuentra en la segunda y tercera posición, respectivamente.

25

Asignación de Valores a los Argumentos

- Los argumentos sin valor asignado pueden tomar más de un valor válido que satisface el built-in.
  - Ejemplo: "*La invocación del built-in swrlb:abs(7, ?x) puede ser satisfecha con 7 y con -7*".
- Por supuesto, no todos los built-ins son indistintos al orden de los argumentos.
  - Ejemplo: "*El built-in greaterThan debe especificar cuál es el argumento principal*".

26

Átomos de Rango de Datos

- Consiste del nombre de un tipo de datos o un conjunto de literales y un argumento único que representa el valor del dato.
- Ejemplo:

xsd:int(?x)

[3, 4, 5](?x)

27

Expresiones de Clase en SWRL

- SWRL soporta el uso de expresiones de clase (descripciones de clase) en las reglas.
  - Ejemplo: "*Regla que clasifica a un individuo como Parent si es miembro de una clase con una propiedad hasChild con una cardinalidad mínima de uno*":

(hasChild >= 1)(?x) -> Parent(?x)

- Esta regla no establece que todos los individuos con un hijo son padres.
- Establece que todos los individuos con la restricción de que su propiedad hasChild tenga una cardinalidad mínima de 1.

28

Expresiones de Clase en SWRL

- Las reglas también pueden usar descripciones de clase para establecer conclusiones acerca de los individuos:

Parent(?x) ->(hasChild >= 1)(?x)

- La regla establece que todos los individuos de la clase Parent tengan una propiedad hasChild con una cardinalidad de uno o mayor.

29

Expresiones de Clase en SWRL

- Ejemplo: "*Regla que establece que si una publicación tiene una restricción de cardinalidad de exactamente uno en su propiedad hasAuthor, es posible concluir que es una publicación de un único autor*".

Publication(?p) ^ (hasAuthor = 1)(?p) ->
SingleAuthorPublication(?p)

- Todas las descripciones de clase utilizadas en SWRL poseen semántica idéntica, esto es que son enunciados lógicos.

30

Expresiones de Clase en SWRL

- La regla:

(hasChild >= 1)(?x) -> Parent(?x)

- Encuentra a todos los individuos para los cuales se puede demostrar (directa o indirectamente) que son miembros de una clase que tiene una restricción de cardinalidad especificada sobre la propiedad hasChild.

31

Expresiones de Clase en SWRL

- El punto más importante a tener en cuenta sobre SWRL es que comparte la suposición del mundo abierto de OWL.
  - Ejemplo: "*Es posible esperar que una regla que infiere si dos individuos de OWL de tipo Autor cooperan en la misma publicación son colaboradores*":

Publication(?p) ^ hasAuthor(?p, ?y) ^ hasAuthor(?p, ?z) -> collaboratesWith(?y, ?z)

32

Expresiones de Clase en SWRL

- La semántica de OWL del mundo abierto no permite suponer que dos personas son automáticamente distintas si tienen diferentes nombres.
- Debido a la coincidencia de patrón las variables y, y z también pueden coincidir con el mismo individuo en esta regla.
- Como se mencionó anteriormente, SWRL soporta los átomos sameAs y differentFrom para determinar si los individuos se refieren al mismo individuo o son distintos.

33

Expresiones de Clase en SWRL

- Ejemplo: "*Regla que indica que dos individuos cooperan entre ellos si son co-autores de un artículo, utilizando la clausula differentFrom en asociación con el axioma owl:allDifferents que establece que todos los individuos de la clase Autor son diferentes entre ellos*".

Publication(?x) ^ hasAuthor(?x, ?y) ^ hasAuthor(?x, ?z) ^ differentFrom(?y, ?z) -> cooperatedWith(?y, ?z)

34

Expresiones de Clase en SWRL

- Del mismo modo, SWRL sólo deduce que dos individuos son los mismos si hay un axioma explícito sameAs de OWL definido para los individuos en una ontología, o si su identidad está implícita mediante otros axiomas en una ontología.

35

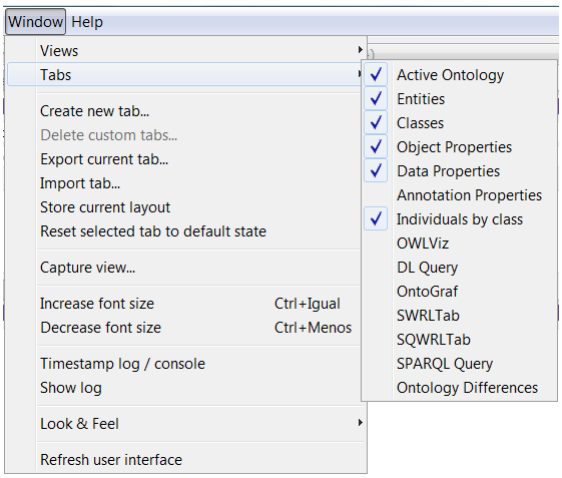
SWRL
SQWRL

Descargar el plugin SWRL

swrltab-plugin-1.0.0-beta-5

https://github.com/protegeproject/swrltab-plugin/releases

SWRLTab



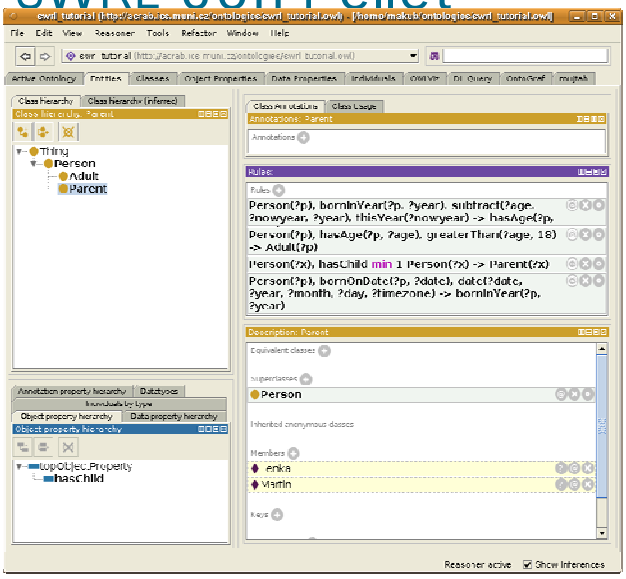
SWRL Editor

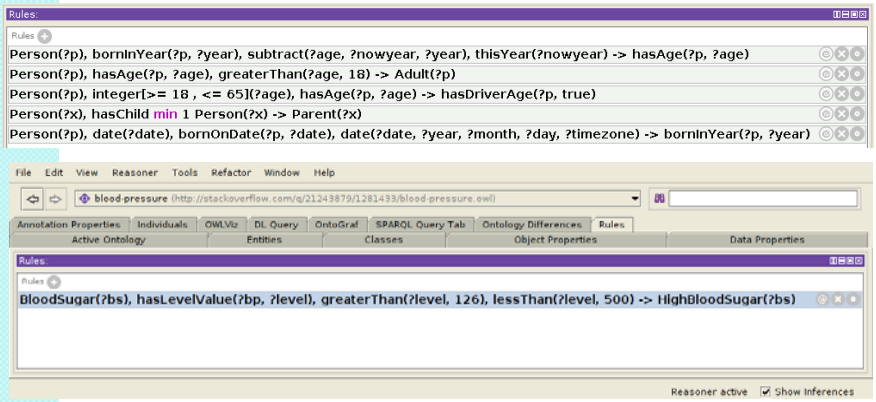
- Es una extensión del componente gráfico que proporciona la SWRLAPI que permite la edición de reglas SWRL.

SQWRL

- Persona(?p) -> sqwrl:select(?p)

SWRL con Pellet





Ejercicio

- Realizar la traducción de las preguntas de competencia a SWRL.

¿Preguntas?