Ingeniería de Aplicaciones para la Web Semántica

Clase 09

La capa lógica

Mg. A. G. Stankevicius

Segundo Cuatrimestre

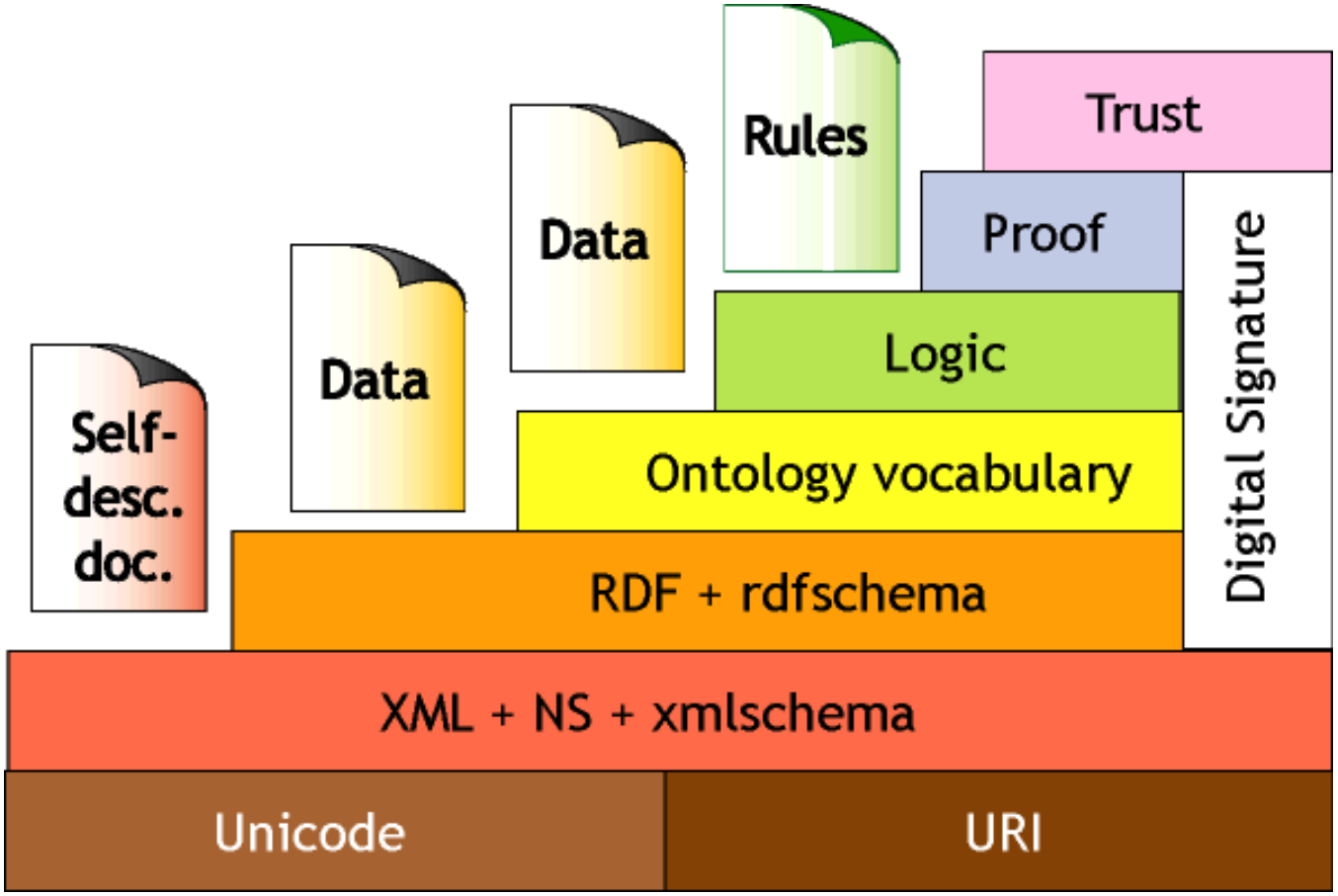
2005

Copyright

Contenidos

- El rol de la lógica en la representación de conocimiento.
- Monotonía vs. No Monotonía.
- Usos de la información codificada como reglas.
- Representación de reglas monótonas.
- Sintaxis y semántica asociada.
- Representación de reglas no monótonas.
- Sintaxis asociada.

Implementación basada en capas de la web semántica



Representación de conocimiento

- La totalidad de los conceptos abordados hasta ahora representan tareas propias de la representación de conocimiento.
- La representación de conocimiento es una disciplina que viene siendo estudiada en el seno de la inteligencia artificial aun desde antes del surgimiento de la WWW.
- La lógica ha sido y sigue siendo la base de la representaciónde conocimiento.

La importancia de la lógica

- La lógica brinda un lenguaje de alto nivel para expresar conocimiento.
- Cuenta con un alto poder expresivo.
- Su semántica es clara, ha sido ampliamente entendida y estudiada.
- Cuenta con una noción de consecuencia lógica cláramente especificada.
- Se han definido sistemas de pruebas que permiten derivar conclusiones a partir de un cierto conjunto de premisas.

La importancia de la lógica

- Existen sistemas de pruebas para los cuales las consecuencias semánticas coinciden con las consecuencias derivadas dentro del sistema de prueba.
  - Estos sistemas satisfacen las propiedades de sensatez y completitud.
- El cálculo de predicados satisface estas propiedades.
  - No así las lógicas de orden superior.
- La lógica puede explicar las inferencias.

Especializaciones del cálculo de predicados

- RDF/S y OWL (tanto Lite como DL) son especializaciones del cálculo de predicados.
  - Tienen una lógica de descripción asociada.
- Describen un subconjunto atractivo de la lógica, que alcanza un adecuado balance entre el poder expresivo y la complejidad computacional.
  - A mayor poder expresivo, menor eficiencia muestra el razonador automático asociado.

Otras especializaciones del cálculo de predicados

- Las lógicas Horn constituyen otra especialización del cálculo de predicados, donde sólo se permite expresiones de la forma:

$$A_1, \ldots, A_n \rightarrow B$$

- Estas reglas admiten dos lecturas:
  - Reglas deductivas: Si se conocen todas las premisas, entonces la conclusión debe valer.
  - Reglas reactivas: Si se verifican todas las premisas, llevar adelante la acción asociada.

Lógicas Horn vs. lógicas de descripción

- Se trata de dos subconjuntos diferentes del cálculo de predicados.
- Por caso, en OWL es imposible expresar conocimiento basado en reglas:

$$estudia(X,Y),\ vive\text{-}en(X,Z),\ ubicación(Y,Z)$$

$$\rightarrow estudiante\text{-}local(X)$$

- De igual forma, no es posible capturar mediante reglas que una persona es una mujer o bien es un hombre, no ambos.

Monotonía vs. No Monotonía

- La lógica convencional presenta un comportamiento monótono: toda conclusión alcanzada en base a un cierto conjunto de premisa no será invalidada al incorporar nuevas premisas.
  - Esta propiedad asegura la validez de los lemas, resultados previos que son usados posteriormentes sin revisitar la totalidad de las demostración asociada.

Monotonía vs. No Monotonía

- No obstante, este comportamiento monótono no parece ser adecuado en un contexto más amplio.
- Muchas situaciones del mundo real requieren poder sacionar conclusiones de manera no monótona.
- Necesitamos poder dejar de sancionar una cierta conclusión a la luz de nueva evidencia.

Un caso concreto

- Supongamos que queremos modelar el hecho de que conviene llevar paragua si sabemos que llueve y conviene no llevarlo si sabemos que no llueve.
- Solución tentativa:
  - Si “llueve”, entonces “llevar paragua”.
  - Si “no llueve”, entonces “no llevar paragua”.
- Pero, ¿qué pasa si no sabemos si está lloviendo?

Solución alternativa

- Una forma de evitar este problema es modificar un poco las premisas de las reglas:
  - Si “llueve”, entonces “llevar paragua”.
  - Si “no tenemos información acerca de si llueve o no”, entonces “no llevar paragua”.
- Esta propuesta soluciona el problema antes identificado, pero:
  - La premisa de la segunda regla no puede ser capturada mediante cálculo de predicados.
  - Necesitamos un nuevo sistema de reglas.

Reglas no monótonas

- La primer solución resulta aplicable en los escenarios en donde se disponga de información total o completa.
- La segunda solución es aplicable a escenario donde sólo se dispone de información parcial o incompleta.
- La segunda regla en el ejemplo anterior es una regla no monótona.

Intercambio de reglas

- Es razonable suponer que diferentes aplicaciones deseen intercambiar conocimiento codificado como reglas:
  - Por caso, una tienda virtual publicitando las políticas de devolución de artículos.
- La idea es hacer uso de alguno de los estandares de la red que permitan un procesamiento automático.
- Solución: codificar este conocimiento en XML (Rule Markup Languages).

Representación de vínculos familiares

- Supongamos que contamos con una base de hechos acerca de las relaciones:
  - $madre(X,\ Y)$, indicando que X es madre de Y.
  - $padre(X,\ Y)$, indicando que X es padre de Y.
  - $hombre(X)$, indicando que X es hombre.
  - $mujer(X)$, indicando que X es mujer.
- En este contexto es posible inferir un número de otros vínculos familiares.

Relaciones inferidas

- Progenitor:

$$madre(X,\ Y) \rightarrow progenitor(X,\ Y)$$

$$padre(X,\ Y) \rightarrow progenitor(X,\ Y)$$

- Hermanos:

$$hombre(X),\ progenitor(Y,\ X),\ progenitor(Y,\ Z),$$
$$distintos(X,\ Z) \rightarrow hermano(X,\ Z)$$

$$mujer(X),\ progenitor(Y,\ X),\ progenitor(Y,\ Z),$$
$$distintos(X,\ Z) \rightarrow hermana(X,\ Z)$$

Reglas monótonas: sintáxis

$$clienteFiel(X),\ edad(X) > 60 \rightarrow rebajar(X)$$

- Observemos los diversos componentes:
  - Variables, que representan a un determinado elemento del dominio.
  - Constantes, que denotan elementos fijos.
  - Predicados, que relacionan elementos.
  - Funciones, que asocian elementos un conjunto de elementos con otro elemento.

Reglas

$$A_1, A_2, \ldots, A_n \rightarrow B$$

- $B, A_1, A_2, \ldots, A_n$ son fórmulas atómicas.
- B es la cabeza de la reglas.
- $A_1, A_2, \ldots, A_n$ es el cuerpo de la regla.
- Las comas en el cuerpo de la regla se leen de forma conjuntiva.
- $B, A_1, A_2, \ldots, A_n$ pueden tener variables.

Programas lógicos

- Los hechos son simples fórmulas atómicas.
  - Las variables que aparecen en los hechos se asumen cuantificadas universalmente.
- Un programa lógico es un conjunto finito de hechos y reglas.
- A todo programa lógico P se le puede asociar su traducción en términos del cálculo de predicados $pl(P)$.

Metas

- Las metas representan las consultas que se pueden determinar mediante un cierto programa lógico.
- Guardan la siguiente estructura:

$$A_1, A_2, \ldots, A_n \rightarrow$$

- Cuando n=0, representa la meta vacía.
- Las variables que aparezcan en las consultas tienen una interpretación existencial.

Las metas como predicados de primer orden

- La meta:

  $A_1, A_2, \ldots, A_n \rightarrow$

- Puede ser reinterpretada como:

  $\forall X_1 \forall X_2 \ldots \forall X_k (\neg A_1 \vee \neg A_2 \vee \ldots \vee \neg A_n)$

- Donde $X_1, X_2, \ldots, X_k$ son todas las variables que aparecen en $A_1, A_2, \ldots, A_n$.

Las metas como predicados de primer orden

- Finalmente, la fórmula:

$$\forall X_1 \, \forall X_2 \, \ldots \, \forall X_k \, (\neg A_1 \vee \neg A_2 \vee \ldots \vee \neg A_n)$$

- Puede ser reescrita como:

$$\neg(\exists X_1 \, \exists X_2 \, \ldots \, \exists X_k \, (A_1 \wedge A_2 \wedge \ldots \wedge A_n))$$

- Observemos que la formula que debe ser probada es negada.
  - Se está haciendo uso de una forma especial de prueba por contradicción.

Semántica asociada

- Dado un programa lógico P y una meta:

  $A_1, A_2, \ldots, A_n \rightarrow$

- Suponiendo que sólo contiene las variables $X_1, X_2, \ldots, X_k$, será satisfecha si:
  - $pl(P) \models \exists X_1 \exists X_2 \ldots \exists X_k (A_1 \wedge A_2 \wedge \ldots \wedge A_n)$
- O bien, equivalentemente:
  - $pl(P) \cup \{\neg(\exists X_1 \exists X_2 \ldots \exists X_k (A_1 \wedge A_2 \wedge \ldots \wedge A_n))\} \vdash \bot$

Obtención de respuestas

- Las consultas hasta ahora reciben una respuesta del tipo **SI**/**NO**.
- Supongamos, el programa p(a) ante la consulta P(X) responde un simple SI.
  - Si bien correcta, no es del todo satisfactoria.
- Sería más apropiado poder computar y retornar la sustitución $\{X/a\}$.
- La resolución SLD permite obtener estas respuestas más apropiadas.

Reglas no monótonas

- En un sistema no monótono es posible que una regla no se pueda aplicar, aun cuando todas sus premisas si lo sean.
  - Se deben contemplar cadenas de razonamiento a favor y en contra.
- Las reglas rebatibles capturan este comportamiento, pueden ser derrotadas por otras reglas.
- Se deben permitir átomos negados en la cabeza de las reglas.

Reglas rebatibles

- Un programa lógico conteniendo las siguientes reglas es contradictorio:
  - $p(X) \rightarrow r(X)$
  - $q(X) \rightarrow \neg r(X)$
- El mismo programa lógico puede ser expresado apelando a reglas rebatibles:
  - $p(X) \Rightarrow r(X)$
  - $q(X) \Rightarrow \neg r(X)$

Reglas rebatibles

- El nuevo programa sigue capturando el conflicto entre las reglas.
- De contar con los hechos $p(a)$ y $q(a)$, no se puede concluir ni $r(a)$ ni $\neg r(a)$.
- Este tipo de conflicto puede ser resuelto apelando a prioridades entre las reglas.
- Supongamos, si la primer regla es mejor que la segunda, inmediantemente podemos concluir que $r(a)$.

Prioridades entre las reglas

- Las prioridades entre las reglas pueden tener distinto origen:
  - Mayor autoridad.
  - Estampillas de tiempo.
  - Conocimiento más específico.
  - Otros.
- Es posible abstraerse del origen de las prioridades asumiento la existencia de una relación entre reglas externa.

Reglas en conflicto

- Al permitir negación en la cabeza de las reglas, es posible que distintas reglas entren en conflicto entre sí.
- El caso más simple es el conflicto a nivel de cabezas de reglas.
  - Una regla sanciona $p(a)$ y la otra $\neg p(a)$.
- Por otra parte, también puede haber conflicto a nivel de conclusiones intermedias.

Reglas en conflicto

- El modelo propuesto atiende a esta multiplicidad de punto de conflicto manteniendo un conjunto de conflicto para cada literal del programa lógico.
- El conjunto de conflicto para un cierto literal L, notado $C(L)$, siempre contendrá al complemento de L, pero bien puede contener a otros literales.

Sintáxis de las reglas rebatibles

$$r: L_1, L_2, \ldots, L_n \Rightarrow L$$

- r es la etiqueta de la regla.
- $L_1, L_2, \ldots, L_n$ son las premisas de la regla.
- L es la conclusión de la regla.
- $L, L_1, L_2, \ldots, L_n$ son literales positivos o negativos.
- No puede contener funciones.

Programas lógicos rebatibles

- Un programa lógico rebatible es una terna $(F, R, >)$, donde:
  - F es un conjunto de hechos.
  - R es un conjunto de reglas rebatibles.
  - $>$ es una relación binaria acíclica sobre los elementos de R.
    - Concretamente, un conjunto de pares $r > r'$, donde r y r' son etiquetas de reglas en R.

Futuras extensiones

- Las propuestas de representación de reglas monótonos y no monótonas dentro de la web semántica son propuestas muy recientes.
- Las condiciones están dadas para proponer nuevas extensiones, que apelen a otras formalizaciones del razonamiento basado en reglas:
  - ¿DeLP?
  - ¿Algún subconjunto de DeLP?