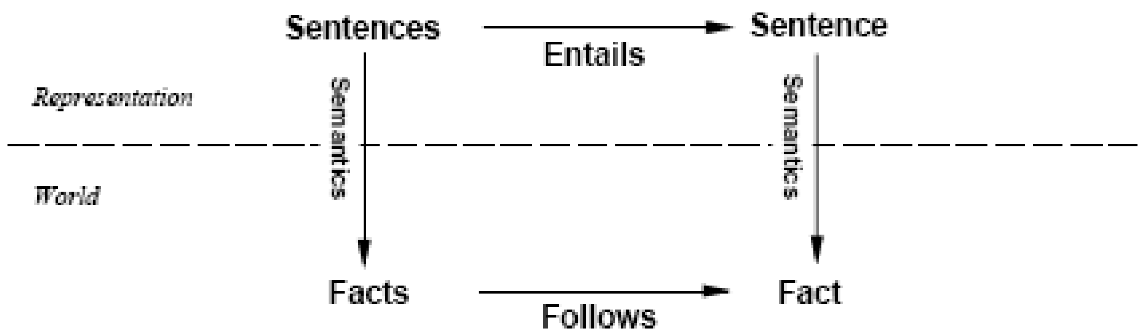
Inteligencia Artificial

Conocimiento y razonamiento
2. Lógica proposicional

Dr. Edgard Iván Benítez Guerrero

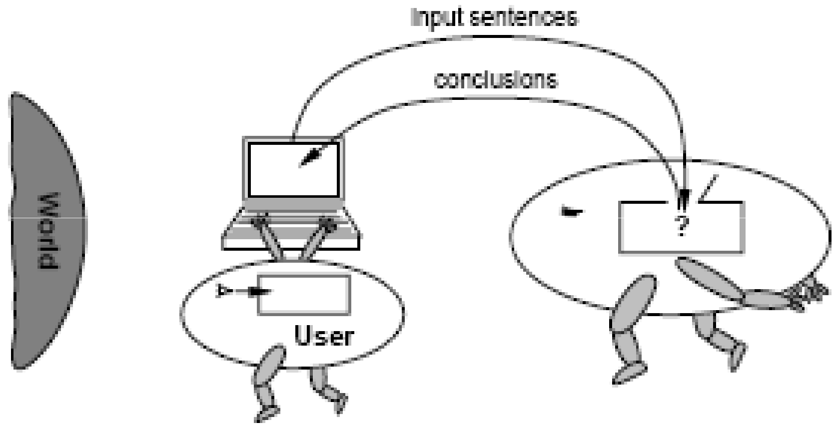
Lenguajes lógicos



- Los hechos forman parte del mundo, mientras que las sentencias son la representación que de ellos tiene el agente
- Se desea generar sentencias que sean necesariamente ciertas a partir de otras que también lo son – que estén lógicamente implicadas - (*entails*), de igual manera que algunos hechos siguen/se derivan de otros (*follows*)
- Es necesario implementar la relación de implicación mediante un procedimiento de inferencia
- Un procedimiento de inferencia que genere sólo sentencias ciertas a partir de sentencias anteriores ciertas se dice que es robusto (*sound*)

Sentencias y su validez

- Una sentencia es válida si y sólo si es cierta sea cual sea su significado y sea cual sea el estado del mundo
- La validez sirve para que el agente pueda razonar
- Normalmente el agente
  - Tendrá el conocimiento de la KB pero no nuestra interpretación
  - No tendrá acceso al mundo al que se refieren las sentencias



Lógica proposicional

- La lógica proposicional es la más simple pero ilustra las ideas básicas
- Supone que existen hechos (proposiciones) que pueden darse o no en el mundo, es decir, ser ciertos o falsos

Lógica proposicional: sintaxis

- Las sentencias se construyen siguiendo las reglas:
  - Las constantes y los símbolos proposicionales son sentencias
  - Una sentencia entre paréntesis es una sentencia
  - Si S es una sentencia, $\neg S$ es una sentencia (negación)
  - Si S1 y S2 son sentencias, $S1 \land S2$ es una sentencia (conjunción)
  - Si S1 y S2 son sentencias, $S1 \lor S2$ es una sentencia (disyunción)
  - Si S1 y S2 son sentencias, $S1 \Rightarrow S2$ es una sentencia (implicación)
  - Si S1 y S2 son sentencias, $S1 \Leftrightarrow S2$ es una sentencia (bi-condicional)
- Existe un orden de precedencia entre los operadores: $\neg$, $\land$, $\lor$, $\Rightarrow$ y $\Leftrightarrow$

Lógica proposicional: semántica

- Cada modelo especifica un valor de verdad para cada símbolo proposicional
- El significado de las conectivas se especifica mediante sus tablas de verdad

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
False	*False*	*True*	*False*	*False*	*True*	*True*
False	*True*	*True*	*False*	*True*	*True*	*False*
True	*False*	*False*	*False*	*True*	*False*	*False*
True	*True*	*False*	*True*	*True*	*True*	*True*

- Para definir el significado de sentencias mas complejas se procede incrementalmente. P. ej.: para definir $(P \vee Q) \wedge \neg S$, obtendremos primero el significado de $(P \vee Q)$ y el de $\neg S$

Inferencia por tablas de verdad

- Las tablas de verdad se pueden utilizar como método para comprobar la validez de una sentencia
- Por ejemplo, para comprobar la validez de

$$((P \lor H) \land \neg H) \Rightarrow P$$

P	H	$P \lor H$	$(P \lor H) \land \neg H$	$((P \lor H) \land \neg H) \Rightarrow P$
False	*False*	*False*	*False*	*True*
False	*True*	*True*	*False*	*True*
True	*False*	*True*	*True*	*True*
True	*True*	*True*	*False*	*True*

Aplicación de reglas de inferencia

- Patrones de inferencia que se utilizan a menudo y cuya robustez se ha probado. Algunos de los más comunes:

Modus Ponens o eliminación de la implicación: $\dfrac{\alpha \Rightarrow \beta,\ \alpha}{\beta}$

Eliminación del "y": $\dfrac{\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n}{\alpha_i}$

Introducción del "y": $\dfrac{\alpha_1,\ \alpha_2,\ \ldots,\ \alpha_n}{\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n}$

Introducción del "o": $\dfrac{\alpha_i}{\alpha_1 \vee \alpha_2 \vee \ldots \vee \alpha_n}$

Eliminación de la doble negación: $\dfrac{\neg\neg\alpha}{\alpha}$

Resolución unitaria: $\dfrac{\alpha \vee \beta,\ \neg\beta}{\alpha}$

Resolución: $\dfrac{\alpha \vee \beta,\ \neg\beta \vee \gamma}{\alpha \vee \gamma}$

- Ej.: para probar que P se deriva de (P ∨ H) y ¬H se debe aplicar la regla de resolución unitaria con P en lugar de α y H en lugar de β

Equivalencia lógica

$$
\begin{array}{rcll}
(\alpha \land \beta) & \equiv & (\beta \land \alpha) & \text{commutativity of } \land \\
(\alpha \lor \beta) & \equiv & (\beta \lor \alpha) & \text{commutativity of } \lor \\
((\alpha \land \beta) \land \gamma) & \equiv & (\alpha \land (\beta \land \gamma)) & \text{associativity of } \land \\
((\alpha \lor \beta) \lor \gamma) & \equiv & (\alpha \lor (\beta \lor \gamma)) & \text{associativity of } \lor \\
\neg(\neg\alpha) & \equiv & \alpha & \text{double-negation elimination} \\
(\alpha \Rightarrow \beta) & \equiv & (\neg\beta \Rightarrow \neg\alpha) & \text{contraposition} \\
(\alpha \Rightarrow \beta) & \equiv & (\neg\alpha \lor \beta) & \text{implication elimination} \\
(\alpha \Leftrightarrow \beta) & \equiv & ((\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha)) & \text{biconditional elimination} \\
\neg(\alpha \land \beta) & \equiv & (\neg\alpha \lor \neg\beta) & \text{de Morgan} \\
\neg(\alpha \lor \beta) & \equiv & (\neg\alpha \land \neg\beta) & \text{de Morgan} \\
(\alpha \land (\beta \lor \gamma)) & \equiv & ((\alpha \land \beta) \lor (\alpha \land \gamma)) & \text{distributivity of } \land \text{ over } \lor \\
(\alpha \lor (\beta \land \gamma)) & \equiv & ((\alpha \lor \beta) \land (\alpha \lor \gamma)) & \text{distributivity of } \lor \text{ over } \land
\end{array}
$$

Encadenamientos hacia adelante y hacia atrás

- Horn Form (restricted)
  - KB = conjunto de clausulas de Horn
  - Una clausula de Horn es un símbolo proposicional o una (conjunción de símbolos) $\Rightarrow$ símbolo; e.g., C, (B $\Rightarrow$ A), (C $\wedge$ D $\Rightarrow$ B)
- Modus Ponens (para forma de Horn):

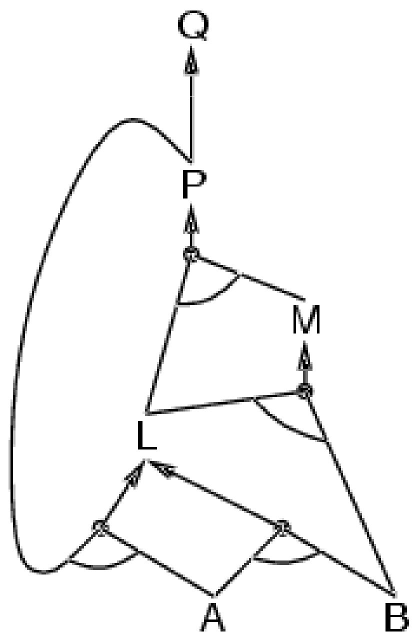
$$\frac{\alpha 1, \ldots, \alpha n, \qquad \alpha 1 \wedge \ldots \wedge \alpha n \Rightarrow \beta}{\beta}$$

- Puede usarse con encadenamientos hacia adelante y hacia atrás

Encadenamiento hacia adelante

- Idea: disparar cualquier regla cuyas premisas se satisfagan en la BC, añadir su conclusión a la BC, hasta que se encuentre la consulta

$$
\begin{aligned}
&P \Rightarrow Q\\
&L \wedge M \Rightarrow P\\
&B \wedge L \Rightarrow M\\
&A \wedge P \Rightarrow L\\
&A \wedge B \Rightarrow L\\
&A\\
&B
\end{aligned}
$$



Algoritmo de encadenamiento hacia adelante

```
function PL-FC-ENTAILS?(KB, q) returns true or false
  local variables: count, a table, indexed by clause, initially the number of premises
                   inferred, a table, indexed by symbol, each entry initially false
                   agenda, a list of symbols, initially the symbols known to be true

  while agenda is not empty do
      p ← POP(agenda)
      unless inferred[p] do
          inferred[p] ← true
          for each Horn clause c in whose premise p appears do
              decrement count[c]
              if count[c] = 0 then do
                  if HEAD[c] = q then return true
                  PUSH(HEAD[c], agenda)
  return false
```

Encadenamiento hacia atrás

- Idea: trabajar hacia atrás desde la consulta q; i.e. para probar q usando BC,
  - Verificar si q ya se conoce o
  - Probar hacia atrás todas las premisas de alguna regla que concluya q
- Evitar ciclos: verificar si una nueva sub-meta ya está en la pila de metas
- Evitar trabajo repetido: verificar si una nueva sub-meta
  - Ya ha sido verificada como verdadera o
  - Ya falló

Comparativa de encadenamientos

- Encadenamiento hacia adelante
  - Proceso guiado por los datos, automatico e inconciente; e.g., reconocimiento de objetos, decisiones rutinarias
  - Puede hacer mucho trabajo que es irrelevante para la meta
- Encadenamiento hacia atras
  - Proceso guiado por la meta
  - Apropiado para la resolución de problemas

Resumen

- Los agentes lógicos aplican inferencia a una base de conocimiento para derivar nueva información y tomar decisiones
- Conceptos básicos de la lógica:
  - Sintaxis: estructura formal de sentencias
  - Semántica: valor de verdad de las sentencias con respecto a modelos
  - Implicación: verdad necesaria de una sentencia dada otra
  - Inferencia: derivar sentencias a partir de otras
  - Robustez: derivaciones producen solamente sentencias implicadas
  - Completez: derivaciones pueden producir todas las sentencias implicadas
- Encadenamientos hacia adelante y hacia atrás
- La lógica proposicional tiene inconvenientes:
  - Está limitada en cuanto a poder expresivo
  - Conocimiento sencillo requiere un gran número de reglas