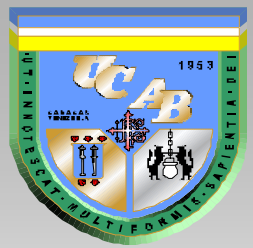


# *Lógica en Ciencias de la Computación. Caso de estudio: PROLOG*

*Prof. Wílmer Pereira*

UCAB / USB

# Papel de la Lógica en Informática

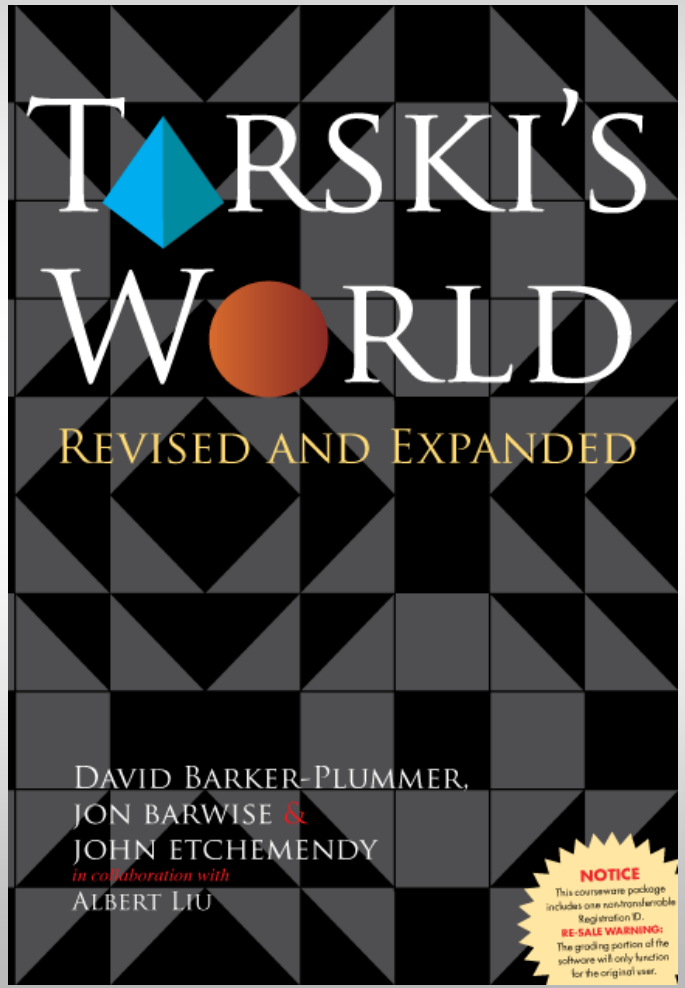
Formación:

- Menos discurso, más razonamiento …
- Área genérica que aplica a múltiples dominios de conocimiento
- Necesaria en bases de datos (restricciones lógicas de integridad) y verificación de programas: correctitud y terminación (GCL de Dijsktra)

Creación:

- En Inteligencia Artificial para representar el conocimiento con más libertad y hacer más flexible la inferencia
- No monotonía: lógicas autoepistémicas, *default logic*, *circumscription*, ...
- Aprendizaje automático desde el punto de vista lógico (árboles de decisión, búsqueda de la mejor hipótesis, planificación lógica en robótica, ...)

# Formación: Softwares educativos




- Un ambiente gráfico permite posicionar poliedros
- Usando predicados específicos, el estudiante escribe fórmulas indicando la posición relativa de los poliedros
- El estudiante puede constatar si la fórmula es verdadera o falsa en el mundo que visualiza
- La interacción es estilo juego ...

# *Formación: Verificación de Programas*

- Dado un programa y su especificación lógica ¿Se obtiene de ambos los resultados?

- Las condiciones de terminación son fundamentales para predecir el comportamiento esperado

- Depuración o ajuste del programa a su especificación es util durante el proceso de desarrollo de software

Resultados concretos:

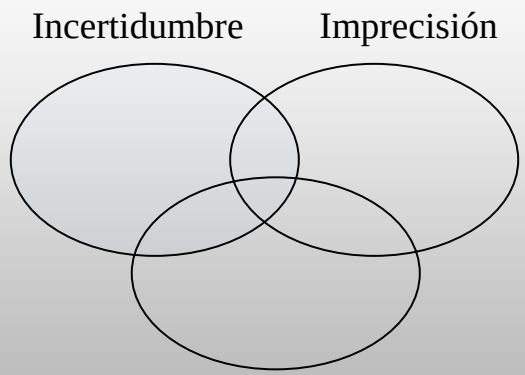
*GCL y muchos otros lenguajes que permiten escribir simultaneamente el programa y su especificación lógica*

# Alcances ...

- El teorema de incompletitud de Gödel aplica a cualquier sistema deductivo que incluya la aritmética ... ☹ ... sin embargo las lógicas no clásicas pretenden no sólo ser deductivas para no estar sujetas a estos límites ...

- La monotonía de la lógica clásica limita su uso para razonamiento de sentido común pues el conocimiento debe ser permanentemente revisable.

$$\beta \mid\!- \delta \quad => \quad \beta \wedge \phi \mid\!- \delta$$



# *Limitaciones ...*

- El cálculo de predicados, en general, no es completo y correcto. Además la NP-Completitud aqueja a todos los formalismos clásicos y no clásicos pues no escapan a las limitaciones de espacio o tiempo de procesamiento
- Afortunadamente hay conjuntos restringidos del cálculo de predicado (clausulas de Horn) que son completos y correctos para ciertos mecanismos de inferencia ... pero ¿serán NP-completos?
- Clausulas:

  Toda fórmula se puede representar en dos formas canónicas gracias a las reglas de distribución, disyunción y De Morgan

  Forma Normal Disyuntiva: FND

  $$(\alpha \wedge \neg\beta \wedge \delta) \vee (\theta \wedge \neg\omega \wedge \beta) \vee (\phi \wedge \neg\rho \wedge \neg\alpha)$$

  Forma Norma Conjuntiva: FNC

  $$(\neg\alpha \vee \neg\delta \vee \rho) \wedge (\theta \vee \rho \vee \phi) \wedge (\beta \vee \neg\omega \vee \alpha)$$

# Clausula de Horn

- Tienen un sólo literal positivo. Por ejemplo:

$$(\neg\alpha \vee \neg\delta \vee \rho)$$

- Así una base de conocimientos sería la conjunción de clausulas

$$(\neg\alpha \vee \neg\delta \vee \rho) \wedge (\theta \vee \neg\rho \vee \neg\phi) \wedge (\neg\beta \vee \omega \vee \neg\alpha)$$

- El principio de resolución de Robinson usa una sola regla de inferencia y es completo y correcto para las clausulas de Horn:

$$\frac{\alpha \qquad (\neg\alpha \vee \beta)}{\beta}$$

- Cada prueba se realiza por contradicción. Este sistema, a diferencia de los sistemas de deducción natural, como Gentzen o Fitch, tiene una sóla regla lo que simplifica su automatización. Es a partir de aquí que nace PROLOG ...

# Programación Lógica

- PROLOG es un lenguaje de programación declarativo desarrollado en la Universidad de Aix-Marseille principalmente por Phillipe Roussel y Alain Colmerauer

- Las clausulas de Horn constituyen la base de conocimiento, que gracias al teorema de la disyunción y De|Morgan se pueden traducir en implicaciones

$$(\neg\alpha \vee \neg\delta \vee \rho) \Leftrightarrow (\neg(\alpha \wedge \delta) \vee \rho) \Leftrightarrow ((\alpha \wedge \delta) \supset \rho)$$

  Finalmente una implicación se traduce en una regla que se expresa en PROLOG como

$((\alpha \wedge \delta) \supset \rho)$ es equivalente a $\rho$ :- $\alpha$ , $\delta$.

- Las reglas se escriben en cualquier orden y es responsabilidad del motor de inferencia (basado en el principio de Resolución) de inferir conocimiento. Por lo tanto separa claramente la especificación lógica del programa del control.

# PROLOG

- Además de clausulas de Horn deben estar en Forma Normal Prenexa y *Skolemizadas* (Cuantificadores existenciales substituidos por funciones).
- Todo programa en PROLOG necesita la unificación y el *backtracking*.
- Con respecto al programador el control lo lleva el motor de inferencia basado en el principio de resolución. Además el papel obscuro de la asignación desaparece.
- Es la inspiración de una corriente en IA denominada Sistemas Expertos

```
padre(juan,jose).
padre(luisa,jose).
madre(juan,maria).
madre(luisa,maria).
hermano(X,Y):-padre(X,Z),padre(X,Z),X\==Y.
hermano(X,Y):-madre(X,Z),madre(X,Z),X\==Y.
abueloPaterno(X,Y):-padre(X,Z),padre(Z,Y).

?- hermano(juan,luisa).
    yes
?-hermano(luisa,jose).
    no
```

# Problemas ...

- No se estudia en los cursos de lógica porque el estudiante aún no programa y ese paradigma de programación no es intutitivo (sobre todo por el *backtracking*)
- No se enseña además por tener poco uso en la industria debido a la preponderancia del paradigma de programación imperativo y orientado a objetos (JAVA, C++, ...).
- Sólo se vé en electivas y para pocas universidades es una materia obligatoria
- Lentitud en tiempo de ejecución y falta de entornos de desarrollo avanzados

# *Soluciones ...*

- Incluirla en los cursos introductorios de lógicos controlando las aplicaciones para evitar el *backtracking* y el dificil problema de entender la recursión.
- Desarrollar, en electivas, sistemas expertos para comprender el paradigma lógico-declarativo.
- También trabajar la programación lógica en cursos de traducción automática a partir de gramáticas (Chomski) y compilación (análisis lexicográfico y análisis sintáctico)
- Presentar la programación lógica como una posibilidad ante la fuerte fuerte presencia de los lenguajes imperativos (en particular con JAVA)
- Desarrollar habilidades para resolver problemas lógicos aplicados en problemas de robótica y en algoritmos bioinspirados.

Para consultar información sobre cursos de lógica, inteligencia artificial y publicaciones:

http://www.ldc.usb.ve/~wpereira

o

http://www.ucab.edu.ve/ingenieria/informatica/giiar

Gracias por su atención

¿PREGUNTAS ?