# Verificación formal e Inteligencia Artificial

Lourdes González Huesca
luglzhuesca@ciencias.unam.mx

Trabajo conjunto con Karla Ramírez Pulido

Facultad de Ciencias, UNAM

Seminario Inteligencia Artificial y Lógica
Instituto de Investigaciones Filosóficas, UNAM

1 de julio de 2025

# Lógica y Computación[1] [2]

- ✓ Lógica matemática: lenguaje y fundamento de las matemáticas.
- ✓ Ciencias de la Computación como el estudio del manejo de información, datos y programas, donde es necesario un pensamiento crítico y formas de razonamiento correctas.


1 Halpern et al., *On the Unusual Effectiveness of Logic in Computer Science*, 2001.

2 Vardi, *A brief history of logic* https://www.cs.rice.edu/~vardi/comp409/history.pdf

# Lógica y Computación[1] [2]

✓ Lógica matemática: lenguaje y fundamento de las matemáticas.

✓ Ciencias de la Computación como el estudio del manejo de información, datos y programas, donde es necesario un pensamiento crítico y formas de razonamiento correctas.

★ Lógica en la Computación
- fundamentos de hardware
- fundamentos de lenguajes de programación
- análisis de algoritmos
- complejidad computacional
- bases de datos
- inteligencia artificial, razonamiento automatizado


1 Halpern et al., *On the Unusual Effectiveness of Logic in Computer Science*, 2001.

2 Vardi, *A brief history of logic* https://www.cs.rice.edu/~vardi/comp409/history.pdf

# Lógica y Computación Teórica

teoría de la computabilidad, complejidad, teoría de modelos, fundamentos de programación, razonamiento automatizado, bases de datos, verificación asistida, teoría de la demostración, etc.

# Lógica y Computación Teórica

teoría de la computabilidad, complejidad, teoría de modelos, fundamentos de programación, razonamiento automatizado, bases de datos, verificación asistida, teoría de la demostración, etc.


Diferentes lógicas como fundamentos para entender/formalizar diferentes aspectos computacionales

- Semántica Axiomática (Lógica de Floyd-Hoare) para estudiar condiciones de programas
- Lógicas para modelar sistemas (Lógicas Modales y/o Temporales)
- Lógicas de orden superior para verificación deductiva (Demostradores de teoremas interactivos o automáticos)
- Sistemas de Tipos para Lenguajes de Programación

# Lógica en acción

Lógicas como lenguajes formales para el razonamiénto válido:

> interpretación de nociones de cómputo usando lógica, es decir, la lógica (matemática) como lenguaje para la construcción, manipulación, uso, etc., de conocimiento o información.

[3] Noam Chomsky menciona el nombre de “Plagiarism software”
https://www.nytimes.com/2023/03/08/opinion/noam-chomsky-chatgpt-ai.html

# Lógica en acción

Lógicas como lenguajes formales para el razonamiénto válido:

> interpretación de nociones de cómputo usando lógica, es decir, la lógica (matemática) como lenguaje para la construcción, manipulación, uso, etc., de conocimiento o información.

Existen muchas aplicaciones de diferentes lógicas, como las tecnologías relacionadas con la inteligencia artificial (IA).

[3] Noam Chomsky menciona el nombre de “Plagiarism software”
https://www.nytimes.com/2023/03/08/opinion/noam-chomsky-chatgpt-ai.html

# Lógica en acción

Lógicas como lenguajes formales para el razonamiénto válido:

> interpretación de nociones de cómputo usando lógica, es decir, la lógica (matemática) como lenguaje para la construcción, manipulación, uso, etc., de conocimiento o información.

Existen muchas aplicaciones de diferentes lógicas, como las tecnologías relacionadas con la inteligencia artificial (IA).

- ★ La IA tiene muchas definiciones pero ellas coinciden en que es un conglomerado de
  herramientas para simular el razonamiento humano en una máquina
- ★ También hay cuestionamientos respecto al nombre de IA, ya que estas herramientas sólo simulan comportamientos y realmente no son inteligentes en el sentido de comprender el lenguaje natural o realizar procesos de manera consciente [3].

[3] Noam Chomsky menciona el nombre de “Plagiarism software”
https://www.nytimes.com/2023/03/08/opinion/noam-chomsky-chatgpt-ai.html

# Garantías de las IAs

Como cualquier sistema computacional, las IAs

- ★ tienen múltiples procesos, entre ellos la traducción e interacción con otros sistemas: bases de datos, entrenamiento de redes, minería de datos, aprendizaje de máquina, etc.;
- ★ utilizan los modelos de lenguajes grandes, *Large Language Models (LLM)*, para obtener respuestas sobre cualquier tema;
- ★ se espera que no tengan errores, es decir
  - son implementaciones seguras
  - dan respuestas *correctas*
  - son sistemas optimizados (tiempo y espacio)

# Garantías de las IAs

Como cualquier sistema computacional, las IAs

- ★ tienen múltiples procesos, entre ellos la traducción e interacción con otros sistemas: bases de datos, entrenamiento de redes, minería de datos, aprendizaje de máquina, etc.;
- ★ utilizan los modelos de lenguajes grandes, *Large Language Models (LLM)*, para obtener respuestas sobre cualquier tema;
- ★ se espera que no tengan errores, es decir
  - son implementaciones seguras
  - dan respuestas *correctas*
  - son sistemas optimizados (tiempo y espacio)

¿cómo garantizar seguridad y confianza
de los sistemas relacionados con las IAs?

# Garantías de implementación respecto a la especificación

En el proceso de desarrollo de software es importante considerar los aspectos de

- seguridad de software (*software reliability*)
  la probabilidad de que un programa funcione sin fallas (cierto tiempo)
- confianza en el software (*software dependability*)
  el software debe estar disponible cuando se requiere, operar correctamente

Estos aspectos están determinados por el diseño y la implementación de software para minimizar los defectos.

# Garantías de implementación respecto a la especificación

En el proceso de desarrollo de software es importante considerar los aspectos de

- seguridad de software (*software reliability*)
  la probabilidad de que un programa funcione sin fallas (cierto tiempo)
- confianza en el software (*software dependability*)
  el software debe estar disponible cuando se requiere, operar correctamente

Estos aspectos están determinados por el diseño y la implementación de software para minimizar los defectos.

Los **métodos formales** son diferentes técnicas y herramientas basadas en teorías matemáticas y computacionales que incrementan la confianza en la corrección de sistemas.

# Métodos formales
## en el desarrollo de software

Usar métodos formales para las etapas de diseño, desarrollo y verificación:

- ✓ dan soporte para describir y desarrollar programas o sistemas
- ✓ son útiles para razonar respecto a ellos
- ✓ elevan los niveles de seguridad y confianza en los requerimientos

# Métodos formales
## en el desarrollo de software

Usar métodos formales para las etapas de diseño, desarrollo y verificación:

- ✓ dan soporte para describir y desarrollar programas o sistemas
- ✓ son útiles para razonar respecto a ellos
- ✓ elevan los niveles de seguridad y confianza en los requerimientos

Estas técnicas ofrecen ambientes y contextos rigurosos para

**especificar** uso de notaciones formales o lenguajes específicos;

**desarrollar** usar la especificación para el diseño como la semática;

**verificar** demostrar propiedades ya sea “a mano” o con ayuda de la computadora para automatizar.

# Verificación Formal

"Verificar que las propiedades de un programa se cumplen en **todas** sus posibles ejecuciones." –*Xavier Leroy*

La verificación formal es **decidir** la correctud de algoritmos/software/implementaciones respecto a una especificación dada.

# Verificación Formal

"Verificar que las propiedades de un programa se cumplen en **todas** sus posibles ejecuciones." –*Xavier Leroy*

La verificación formal es **decidir** la correctud de algoritmos/software/implementaciones respecto a una especificación dada.

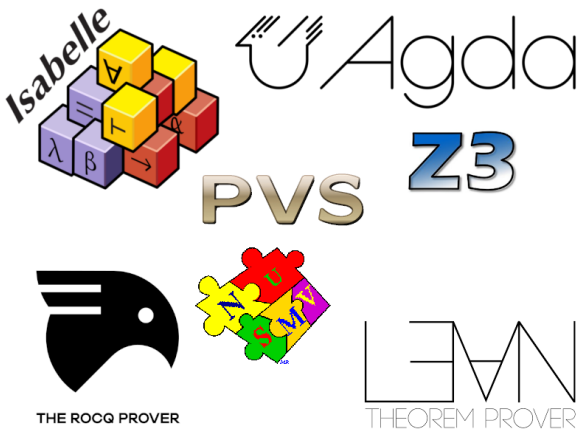
Existen diferentes formas de hacerlo:

- **Probar** un programa durante y después de su desarrollo (*Unit Testing*.)
- **Verificar** una implementación existente (*Hoare Logic, Model Checking, Static Analysis*, etc.)
- Escribir programas que son **correctos por construcción** (*Proof Assistants, Dependently Typed Programming Languages*.)

# AP e IA

Asistentes de prueba (AP) después de la década de 1980:

- ✓ están en auge para sustentar el conocimiento científico;
- ✓ analizar programas que se usan en la vida diaria (pilotos automáticos);
- ✓ además de favorecer el proceso de descubrimiento matemático.




Inteligencias artificiales (IA) después de la década de 1970:

- ✓ están en una explosión de uso para facilitar tareas;
- ✓ ahora son sistemas con los que interactuamos (in)directamente;
- ✓ también favorecen la manipulación y descubrimiento de datos.

# AP e IA

¿ Cómo favorecer ambas áreas computacionales para el mejor manejo de información ?

# AP e IA

¿ Cómo favorecer ambas áreas computacionales para el mejor manejo de información ?

Lenguajes de especificación y verificación de sistemas computacionales.

Correctud de modelos y sistemas.

Ambos deben facilitar la colaboración entre sistemas y usar lenguajes de programación ágiles y seguros.

# AP e IA

¿ Cómo favorecer ambas áreas computacionales para el mejor manejo de información ?

Lenguajes de especificación y verificación de sistemas computacionales.

Correctud de modelos y sistemas.

Ambos deben facilitar la colaboración entre sistemas y usar lenguajes de programación ágiles y seguros.

**verificación de sistemas**

# Verificación de programas

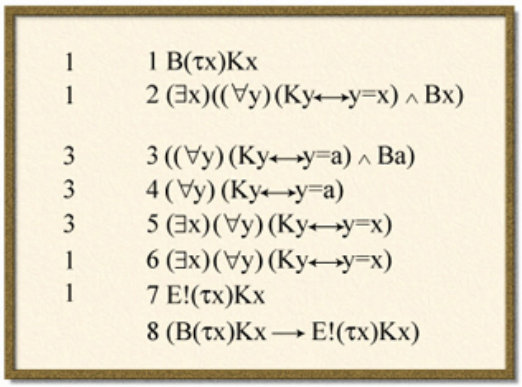
A través de la estrecha relación entre lógica y computación se construyen los verificadores:

las demostraciones equivalen los programas

# Verificación de programas

A través de la estrecha relación entre lógica y computación se construyen los verificadores:

las demostraciones equivalen los programas



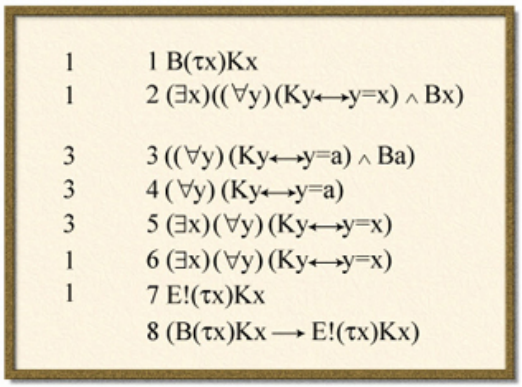
| | |
|---|---|
| 1 | 1 $B(\tau x)Kx$ |
| 1 | 2 $(\exists x)((\forall y)(Ky \leftrightarrow y=x) \wedge Bx)$ |
| 3 | 3 $((\forall y)(Ky \leftrightarrow y=a) \wedge Ba)$ |
| 3 | 4 $(\forall y)(Ky \leftrightarrow y=a)$ |
| 3 | 5 $(\exists x)(\forall y)(Ky \leftrightarrow y=x)$ |
| 1 | 6 $(\exists x)(\forall y)(Ky \leftrightarrow y=x)$ |
| 1 | 7 $E!(\tau x)Kx$ |
| | 8 $(B(\tau x)Kx \rightarrow E!(\tau x)Kx)$ |

```
qsort       :: [Int] → [Int]
qsort []     = []
qsort (x:xs) =
   qsort smaller ++ [x] ++ qsort larger
   where
      smaller = [a | a ← xs, a ≤ x]
      larger  = [b | b ← xs, b > x]
```

# Verificación de programas

A través de la estrecha relación entre lógica y computación se construyen los verificadores:

las demostraciones equivalen los programas



| | |
|---|---|
| 1 | 1 B(τx)Kx |
| 1 | 2 (∃x)((∀y)(Ky↔y=x) ∧ Bx) |
| 3 | 3 ((∀y)(Ky↔y=a) ∧ Ba) |
| 3 | 4 (∀y)(Ky↔y=a) |
| 3 | 5 (∃x)(∀y)(Ky↔y=x) |
| 1 | 6 (∃x)(∀y)(Ky↔y=x) |
| 1 | 7 E!(τx)Kx |
| | 8 (B(τx)Kx → E!(τx)Kx) |

```
qsort       :: [Int] → [Int]
qsort []     = []
qsort (x:xs) =
   qsort smaller ++ [x] ++ qsort larger
   where
      smaller = [a | a ← xs, a ≤ x]
      larger  = [b | b ← xs, b > x]
```

Los **verificadores** son *teorías* bien fundamentadas que han sido estudiadas a detalle:

son pequeños programas correctos que
ayudan a revisar programas más grandes y complejos.

# Sistemas manejadores de pruebas

## Asistente de pruebas

*Ambiente* para desarrollar pruebas ✚ Verificador de pruebas

# Sistemas manejadores de pruebas

## Asistente de pruebas

*Ambiente* para desarrollar pruebas ✚ Verificador de pruebas

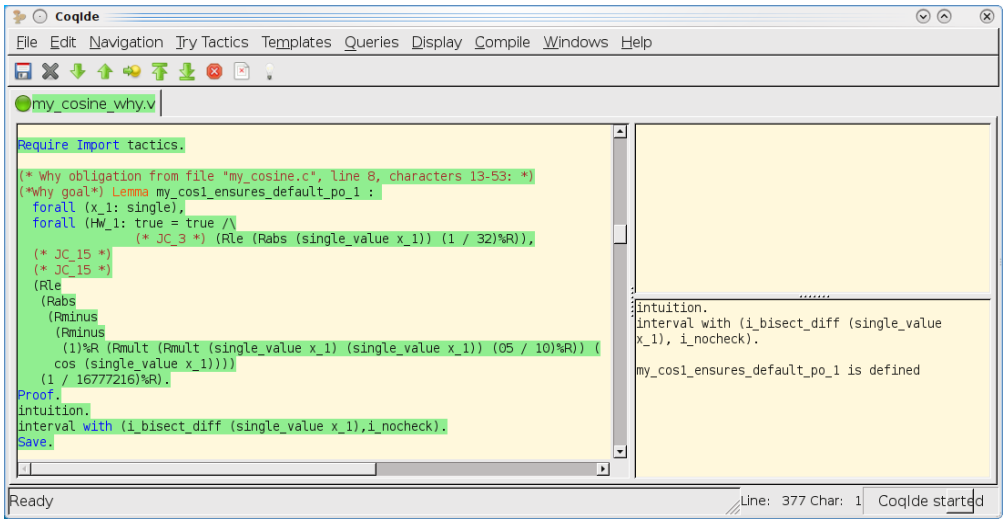
- ➔ un programa interactivo
- ➔ sirve de guía al usuario para obtener demostraciones
- ➔ además verifica las demostraciones y
- ➔ si son correctas, potencialmente son transformadas en programas.

# Sistemas manejadores de pruebas

## Asistente de pruebas

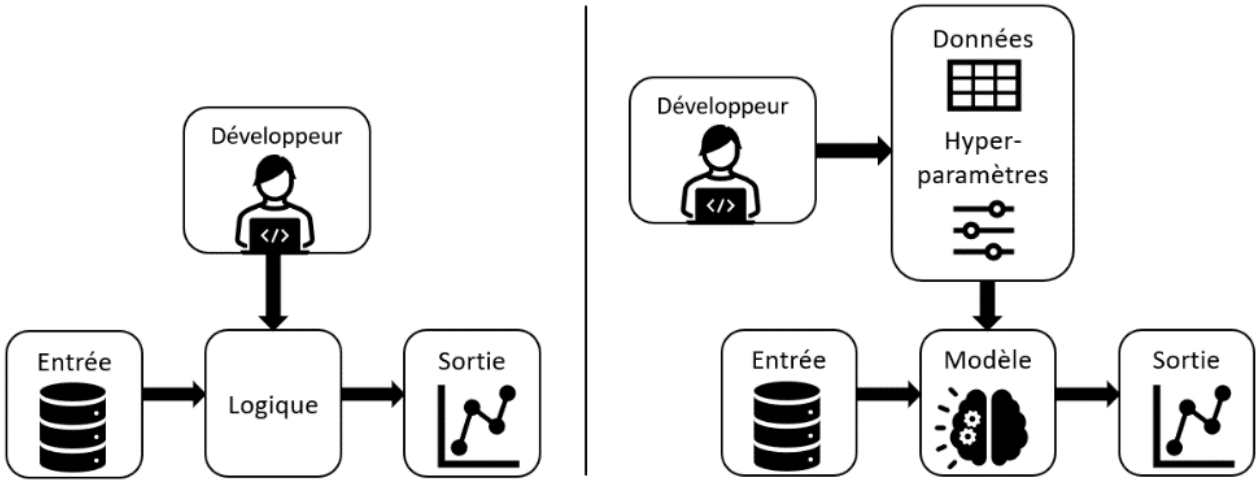
*Ambiente* para desarrollar pruebas ✚ Verificador de pruebas

- ➔ un programa interactivo
- ➔ sirve de guía al usuario para obtener demostraciones
- ➔ además verifica las demostraciones y
- ➔ si son correctas, potencialmente son transformadas en programas.




# Verificación formal para IAs [4]

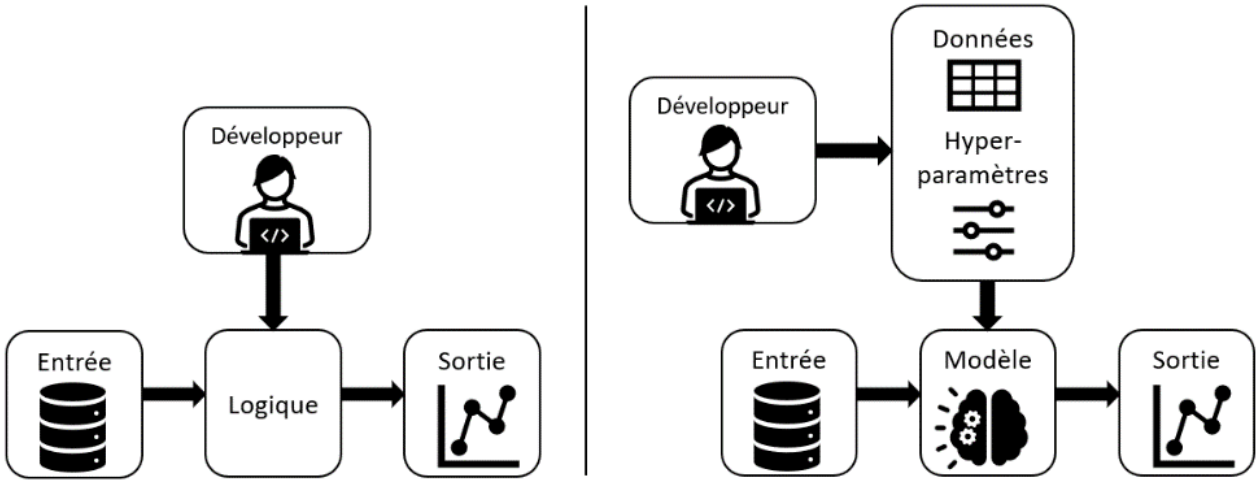
Las herramientas en las IAs, como ChatGPT o Deepseek, son programas y sistemas que manejan grandes datos, realizan reconocimiento de patrones, obtienen algunas deducciones, etc.




4 https://www.kereval.com/formal-verification-of-artificial-intelligence/

# Verificación formal para IAs [4]

Las herramientas en las IAs, como ChatGPT o Deepseek, son programas y sistemas que manejan grandes datos, realizan reconocimiento de patrones, obtienen algunas deducciones, etc.




Se puede incluir la verificación formal en estos sistemas ya que como cualesquiera otros son simples implementaciones.

4 https://www.kereval.com/formal-verification-of-artificial-intelligence/

# Verificación Formal para IAs[5]

Existen diferentes formas de hacerlo:

- ★ Resolvedores SMT (e.g., Z3, CVC4): herramientas para verificar la satisfacibilidad de fórmulas lógicas al cooperar en la verificación de propiedades de modelos de IAs.
- ★ Verificadores de Modelos (e.g., SPIN, PRISM): herramientas que verifican automáticamente si un sistema satisface ciertas especificaciones.
- ★ Verificadores para Redes Neuronales (e.g., Reluplex, AI2): herramientas diseñadas específicamente para verificar el comportamiento de redes neuronales, en particular propiedades de robustez contra entradas *adversarias*.

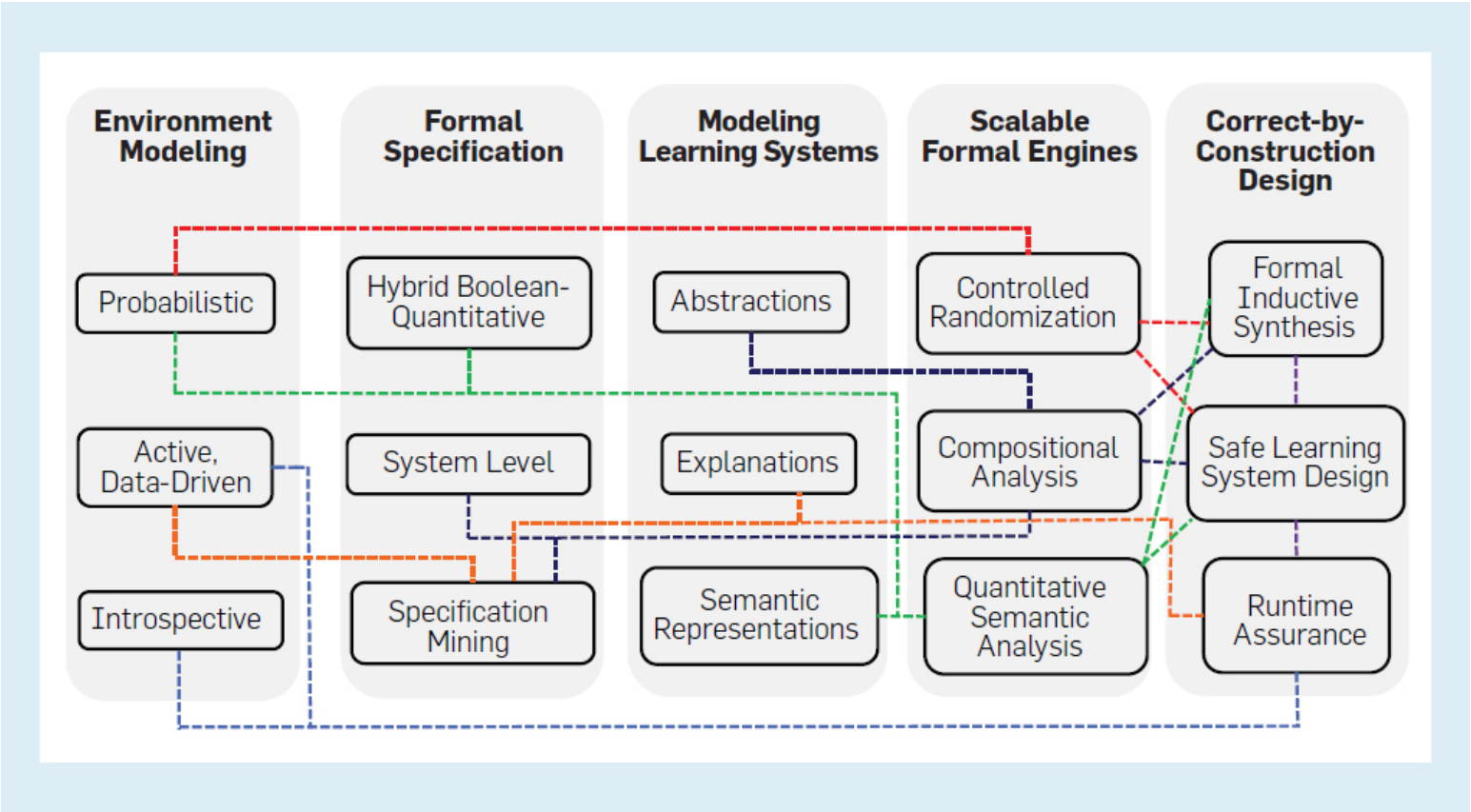

5 Liang, Warren, Taofeek, Adedokun and Johnson Mary, Britney. (2024). Formal Methods and Verification Techniques for Secure and Reliable AI.

# Retos de la VF en IAs [6]

- Desarrollo y fundamentos de lenguajes de programación aptos para el modelado y abstracción de datos.
- Representaciones adecuadas para componentes y sistemas de *Machine Learning*.
- Nuevos formalismos para especificar propiedades de las IAs.
- Herramientas computacionales que puedan ser escalables para razonamiento automatizado, técnicas algorítmicas y diseños confiables desde su concepción y construcción.


6 https://cacm.acm.org/research/toward-verified-artificial-intelligence/

# Retos de la VF en IAs [7]




7 https://cacm.acm.org/research/toward-verified-artificial-intelligence/

# Verificación formal para IAs

Se puede verificar la correspondencia entre especificaciones e implementaciones de un sistema de IA, de esta manera se aseguran las garantías como confiabilidad y seguridad.

Hay varias empresas que se dedican a la verificación formal, y ahora en específico para IAs:

- https://www.galois.com/
- https://www.kereval.com/home/

# Verificación formal para IAs

Se puede verificar la correspondencia entre especificaciones e implementaciones de un sistema de IA, de esta manera se aseguran las garantías como confiabilidad y seguridad.

Hay varias empresas que se dedican a la verificación formal, y ahora en específico para IAs:

- https://www.galois.com/
- https://www.kereval.com/home/

Pero el otro lado de la moneda es la perspectiva de la usuaria final de estas IAs:

se debe ser “muy buena” usando o postulando preguntas a las IAs

# Creatividad e IA

Usualmente se deja la parte creativa a las IAs:

delegamos tanto las preguntas como las respuestas
esperando que “lean nuestra mente” y
se comporten como un interlocutor de
nuestras propias necesidades.

# Creatividad e IA

Usualmente se deja la parte creativa a las IAs:

delegamos tanto las preguntas como las respuestas
esperando que “lean nuestra mente” y
se comporten como un interlocutor de
nuestras propias necesidades.

Si requerimos realizar una pregunta a una IA, ésta se debe refinar para no obtener respuestas incorrectas o incompletas:

la pregunta “perfecta”

contextualizada, sin ambigüedad, detallada y bien redactada

# Retos de la VF en IAs

*proofs-as-programs vs proofs-as-answers*

Inspirándose en un método formal indispensable en el diseño de lenguajes de programación: la Correspondencia Curry-Howard, en donde todo término o expresión tiene un tipo.

$$\begin{aligned} \text{fórmulas} &\iff \text{tipos} \\ \text{pruebas} &\iff \text{términos} \\ \text{verificación de pruebas} &\iff \text{verificación de términos} \end{aligned}$$

Proponemos una correspondencia semejante para refinar las preguntas que hacemos a una IA, en miras de tener preguntas y respuestas ideales:

$$\begin{aligned} \text{preguntas} &\iff \text{especificaciones} \\ \text{respuestas} &\iff \text{demostraciones} \\ \text{verificación de respuestas} &\iff \text{verificación de demostraciones} \end{aligned}$$

# Comentarios finales

## Verificación Formal

★ La verificación formal se está volviendo una práctica importante en todos los ámbitos.

✓ Los asistentes de prueba expanden nuestras capacidades de verificación y mecanización, pero también de razonamiento formal.

Pero aún falta mucho:

- facilitar el acceso a los asistentes de prueba
- incluir la verificación de programas en el desarrollo de software
- formalización de modelos matemáticos

# Comentarios finales[8]

## Verificación Formal & IAs

Proponemos dos líneas de investigación que conectan ambas áreas computacionales:

- Incorporar los métodos formales y la verificación formal en el desarrollo de herramientas de las IAs
  área que ya se encuentra vigente tanto a nivel académico como industrial

8 Todas las imágenes usadas en la presentación se usaron bajo la licencia Creative Commons.

# Comentarios finales[8]

## Verificación Formal & IAs

Proponemos dos líneas de investigación que conectan ambas áreas computacionales:

- Incorporar los métodos formales y la verificación formal en el desarrollo de herramientas de las IAs
  área que ya se encuentra vigente tanto a nivel académico como industrial
- Elevar el nivel de formalidad a la interacción con una IA particular mediante el refinamiento de preguntas
  oportunidad para usar IAs en áreas sensibles que requieren rigor en las especificaciones


8 Todas las imágenes usadas en la presentación se usaron bajo la licencia Creative Commons.

# Comentarios finales[8]

## Verificación Formal & IAs

Proponemos dos líneas de investigación que conectan ambas áreas computacionales:

- Incorporar los métodos formales y la verificación formal en el desarrollo de herramientas de las IAs
  área que ya se encuentra vigente tanto a nivel académico como industrial
- Elevar el nivel de formalidad a la interacción con una IA particular mediante el refinamiento de preguntas
  oportunidad para usar IAs en áreas sensibles que requieren rigor en las especificaciones

**Gracias**


8 Todas las imágenes usadas en la presentación se usaron bajo la licencia Creative Commons.