An introduction to logic

Gabriele Puppis

First and foremost: interrupt!

Logic as a formal language

What is “a logic”?

A formal language to express and reason about properties of objects (numbers, graphs, computations, etc)

👍 a crucial tool for verifying, validating, synthesising safety-critical systems

What is “a logic”?

A formal language to express and reason about properties of objects (numbers, graphs, computations, etc)

👍 a crucial tool for verifying, validating, synthesising safety-critical systems

Two (in)famous examples where <u>logical formal methods</u> would have saved $$$:

- Pentium I floating point division bug (loss @Intel > $475,000,000)

$$\frac{4{,}195{,}835}{3{,}145{,}727} = 1.3338204491362410025$$

The world of mathematics.

$$\frac{4{,}195{,}835}{3{,}145{,}727} = 1.3337390689020375894$$

The world from the Pentium's point of view.

- Ariane 501 overflow bug: (loss @ESA > $500,000,000)

What is “a logic”?

What is “a logic”?

As a formal language, a logic requires:

- a vocabulary: list of symbols we can use to denote objects and properties
- a syntax: grammar for combining symbols into well-formed terms (formulas)
- a semantics: way of assigning a meaning to symbols and terms
- some algorithms: ways of reasoning automatically about the semantics (e.g. checking whether a given formula is valid)

Syntax and semantics

Q. “Are you two married?”
A. “Depends...”

Syntax and semantics

Q. “Are you two married?”
A. “Depends…”

Two possible relations:

- unary relation $R_1(x)$ = “x is married with someone”
- binary relation $R_2(x,y)$ = “x and y are married together”

Syntax and semantics

Q. "Are you two married?"
A. "Depends..."

Two possible relations:

- unary relation $R_1(x)$ = "x is married with someone"
- binary relation $R_2(x,y)$ = "x and y are married together"

Note: $R_2(x,y) \rightarrow R_1(x) \,\&\, R_1(y)$ but not the converse!

Syntax and semantics

"All humans are mortal.
Socrates is human.
So Socrates is mortal."

$$((\forall x\, A(x) \to B(x)) \wedge A(y)) \to B(y)$$

Syntax and semantics

"All humans are mortal.
Socrates is human.
So Socrates is mortal."

$$((\forall x\, A(x) \rightarrow B(x)) \wedge A(y)) \rightarrow B(y)$$

The above formula is *valid* (i.e. true in every model). We call it a tautology

Syntax and semantics

"All humans are mortal.
Socrates is human.
So Socrates is mortal."

"All beatles have 6 legs.
John Lennon is a beatle.
So John Lennon has 6 legs.

$$((\forall x\, A(x) \to B(x)) \wedge A(y)) \to B(y)$$

The above formula is *valid* (i.e. true in every model). We call it a tautology

Syntax and semantics

"There is a barber who shaves all and only those men who do not shave themselves."

$$\exists x\ \forall y\ C(x,y) \leftrightarrow \neg C(y,y)$$

Syntax and semantics

"There is a barber who shaves all and only those men who do not shave themselves."

$$\exists x\ \forall y\ C(x,y) \leftrightarrow \neg C(y,y)$$

The formula is *not satisfiable* (i.e. false in every model). We call it a <u>contradiction</u>

Syntax and semantics

"There exists a set that
contains all and only
the sets that do not
contain themselves."

"There is a barber who
shaves all and only
those men who do not
shave themselves."

$$\exists x\ \forall y\ C(x,y) \leftrightarrow \neg C(y,y)$$

The formula is *not satisfiable* (i.e. false in every model). We call it a <u>contradiction</u>

Syntax and semantics

"Every incoming order is eventually processed."

$$\forall o\ \forall t\quad A(o,t)\ \rightarrow\ \exists t'\ \ t<t'\ \wedge\ B(o,t')$$

Syntax and semantics

"Every incoming order is eventually processed."

$$\forall o\ \forall t \quad A(o,t) \rightarrow \exists t' \ \ t<t' \wedge B(o,t')$$

The formula has *variables of different sorts* (representing objects of different types)

Syntax and semantics

"If a non-deterministic program
can reach infinitely many configurations,
it has an infinite execution."

Syntax and semantics

"If a finitely branching tree
has infinitely many nodes,
it contains an infinite path."

"If a non-deterministic program
can reach infinitely many configurations,
it has an infinite execution."

Syntax and semantics

"If a finitely branching tree
has infinitely many nodes,
it contains an infinite path."

"If a non-deterministic program
can reach infinitely many configurations,
it has an infinite execution."

$$
\begin{array}{llll}
 & \forall x\, \forall y\, \forall z & A(y,x) \wedge A(z,x) \rightarrow y=z \vee A(y,z) \vee A(z,y) & \text{// structure is a tree} \\
\wedge & \forall x\, \neg\exists^\infty y & A(x,y) \wedge \forall z\, A(x,z) \rightarrow y=z \vee A(y,z) & \text{// is finitely branching} \\
\wedge & \exists^\infty x & & \text{// is infinite} \\
\rightarrow & & & \text{// so} \\
 & \exists S\, \exists^\infty x & x \in S & \text{// there is an infinite set} \\
\wedge & \forall x\, \forall y & x \in S \wedge y \in S \rightarrow x=y \vee A(x,y) \vee A(y,x) & \text{// which is a path}
\end{array}
$$

Syntax and semantics

"If a finitely branching tree
has infinitely many nodes,
it contains an infinite path."

"If a non-deterministic program
can reach infinitely many configurations,
it has an infinite execution."

$$
\begin{array}{llll}
 & \forall x\, \forall y\, \forall z & A(y,x) \wedge A(z,x) \rightarrow y=z \vee A(y,z) \vee A(z,y) & \text{// structure is a tree} \\
\wedge & \forall x\, \neg\exists^\infty y & A(x,y) \wedge \forall z\, A(x,z) \rightarrow y=z \vee A(y,z) & \text{// is finitely branching} \\
\wedge & \exists^\infty x & & \text{// is infinite} \\
\rightarrow & & & \text{// so} \\
 & \exists S\, \exists^\infty x & x \in S & \text{// there is an infinite set} \\
\wedge & \forall x\, \forall y & x\in S \wedge y \in S \rightarrow x=y \vee A(x,y) \vee A(y,x) & \text{// which is a path}
\end{array}
$$

The formula uses *one relational symbol* A (for the "ancestor-of" relation)
and *different types of quantifiers* ($\forall x$, $\exists x$, $\exists^\infty x$, $\exists S$)

Algorithms

Evaluation of a formula
Validity / satisfiability
Logical equivalence
Logical consequence
Definability in a logical fragment
...

What can be mechanized? $\rightsquigarrow$ decidable/undecidable

How hard is it to mechanise? $\rightsquigarrow$ complexity classes

Algorithms

Evaluation of a formula
Validity / satisfiability
Logical equivalence
Logical consequence
Definability in a logical fragment
…

Domino H's 10th PCP K $\vdots$

What can be mechanized? $\rightsquigarrow$ decidable/undecidable

How hard is it to mechanise? $\rightsquigarrow$ complexity classes

usage of resources:
- time
- memory

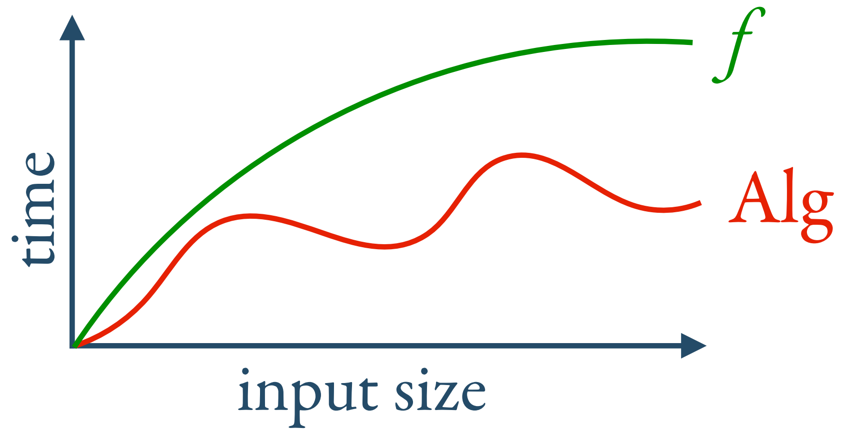
Algorithms

Domino H's 10th PCP K ⋮

What can be mechanized? $\rightsquigarrow$ decidable/undecidable

How hard is it to mechanise? $\rightsquigarrow$ complexity classes

usage of resources:
- time
- memory



Algorithms

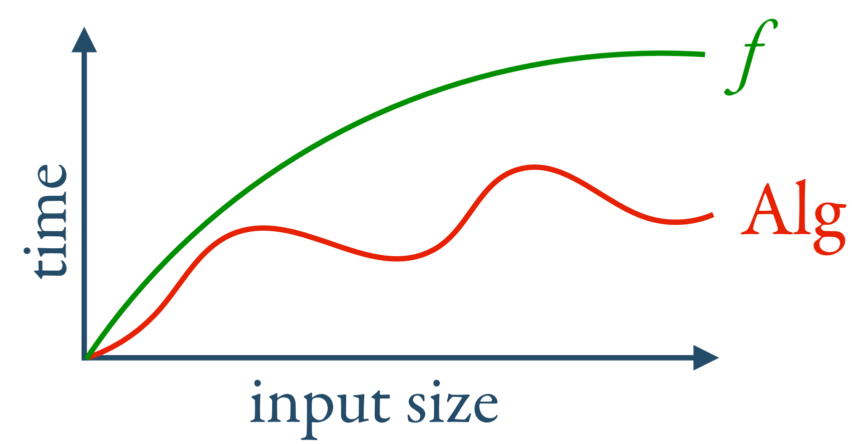
Domino H's 10th PCP K ⋮

What can be mechanized? $\rightsquigarrow$ decidable/undecidable

How hard is it to mechanise? $\rightsquigarrow$ complexity classes

usage of resources:
- time
- memory

Algorithm **Alg** is TIME-bounded by a function $\mathbf{f}: \mathbb{N} \longrightarrow \mathbb{N}$ if **Alg**(*input*) uses less than $\mathbf{f}(|\mathit{input}|)$ units of TIME.



Algorithms

Domino H's 10th PCP K ⋮

What can be mechanized? $\rightsquigarrow$ decidable/undecidable

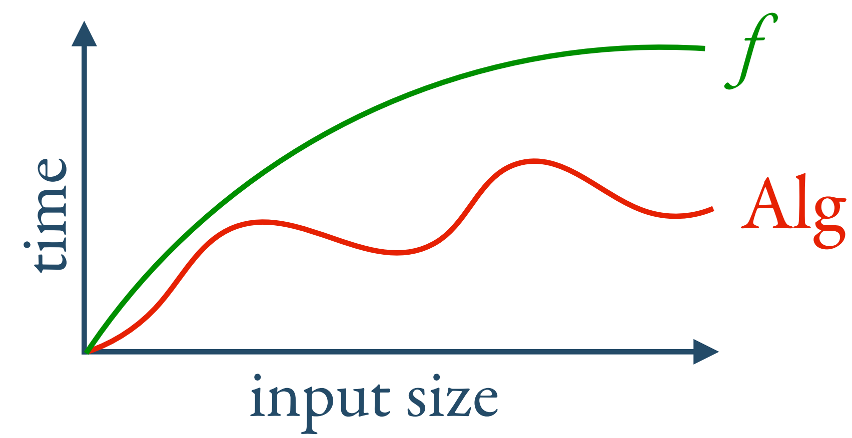
How hard is it to mechanise? $\rightsquigarrow$ complexity classes

usage of resources:
- time
- memory

Algorithm **Alg** is ~~TIME~~ **SPACE**-bounded

by a function **f**: $\mathbb{N} \longrightarrow \mathbb{N}$ if

Alg(*input*) uses less than **f**(|*input*|) units of ~~TIME.~~ SPACE.



Algorithms

Domino H's 10th PCP K ⋮

What can be mechanized? $\rightsquigarrow$ decidable/undecidable

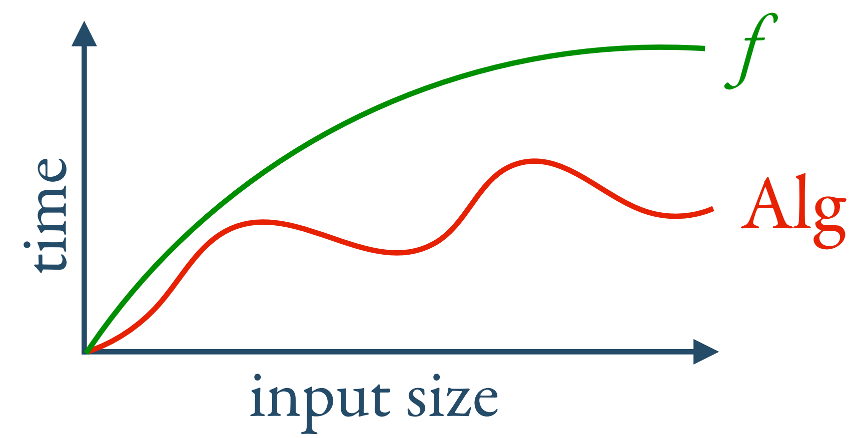
How hard is it to mechanise? $\rightsquigarrow$ complexity classes

usage of resources:
- time
- memory

Algorithm **Alg** is ~~TIME~~ **SPACE**-bounded

by a function **f**: $\mathbb{N} \longrightarrow \mathbb{N}$ if

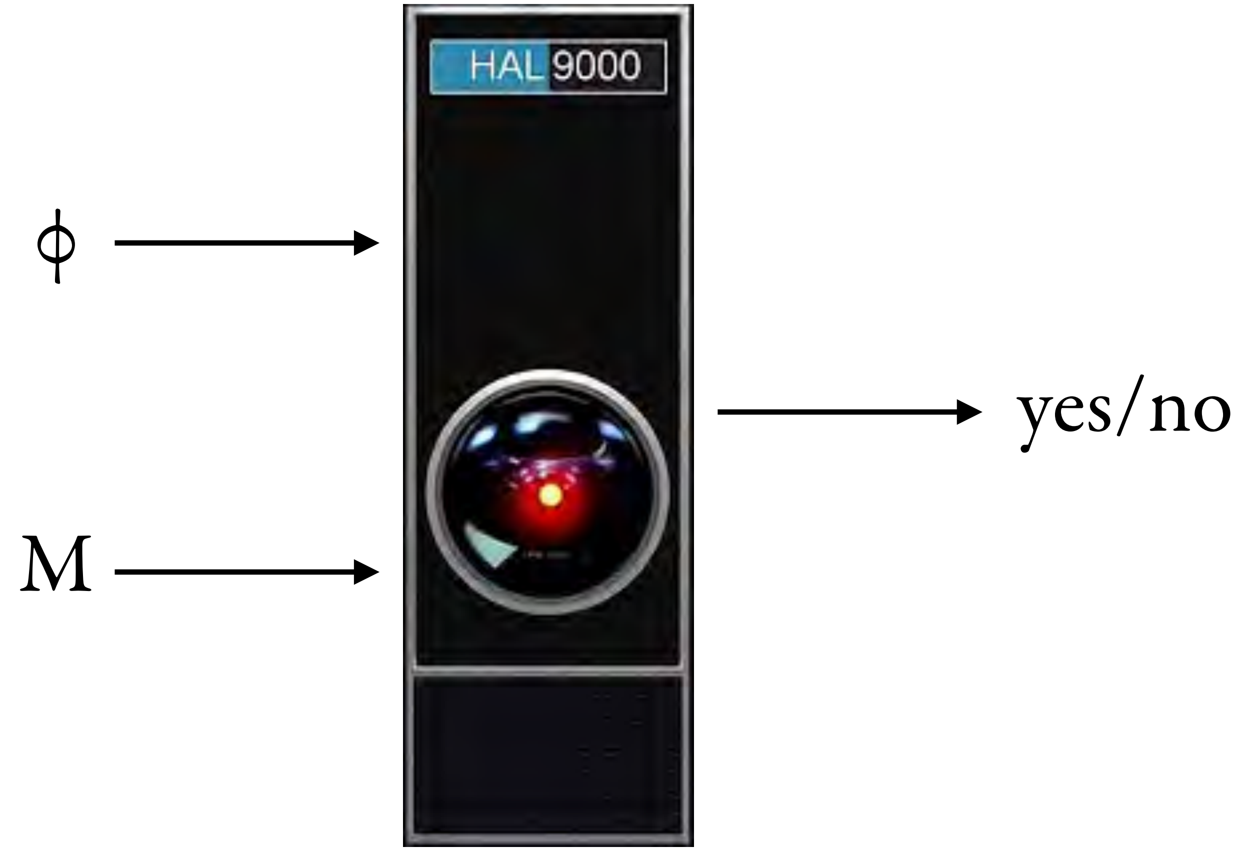
Alg(*input*) uses less than **f**(|*input*|) units of ~~TIME.~~ SPACE.



P $\subseteq$ **NP** $\subseteq$ **PSPACE** $\subseteq$ **EXP** $\subseteq \cdots$

- P: Deterministic and TIME-bounded by a polynomial
- NP: Non-deterministic and TIME-bounded by a polynomial
- PSPACE: SPACE-bounded by a polynomial

Algorithms

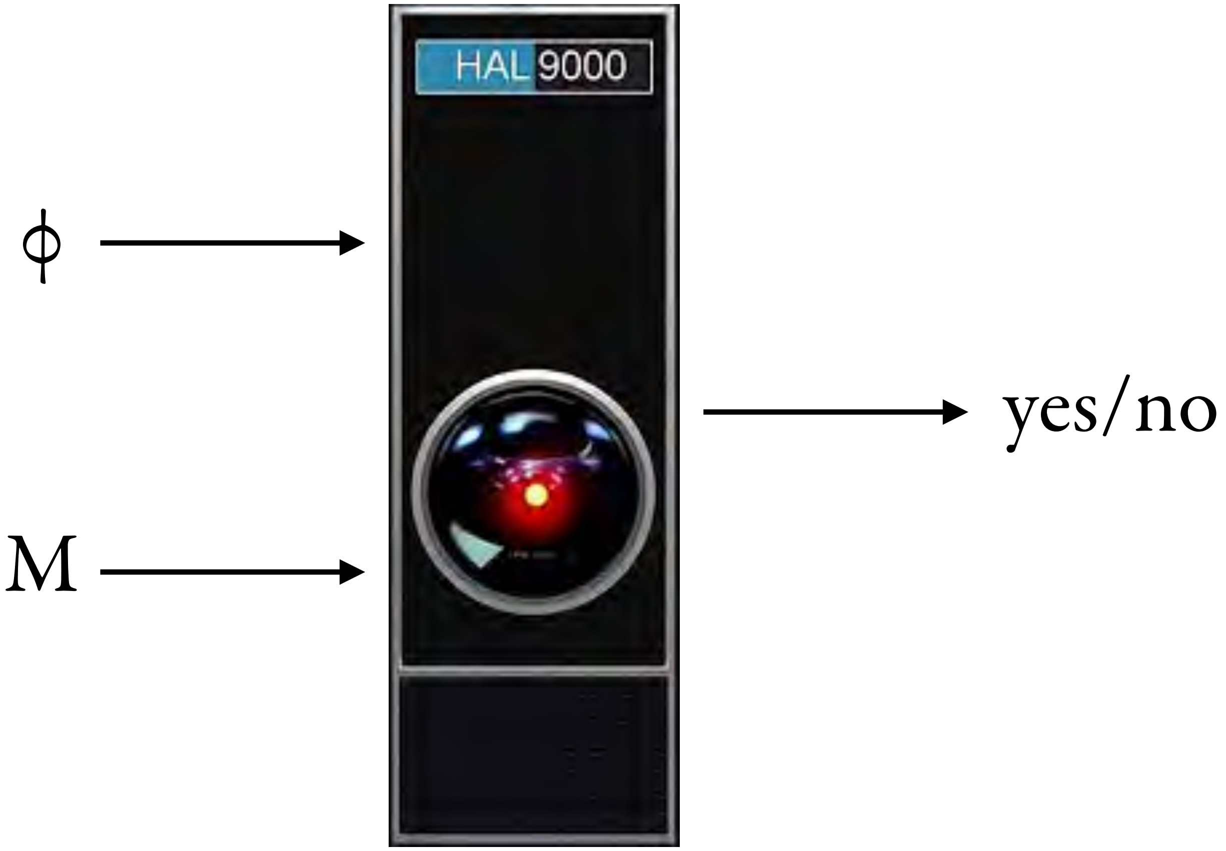


<u>Model-checking</u> problem

input: formula ϕ + model M

output: yes iff ϕ holds on M $(M \vDash \phi)$

Algorithms



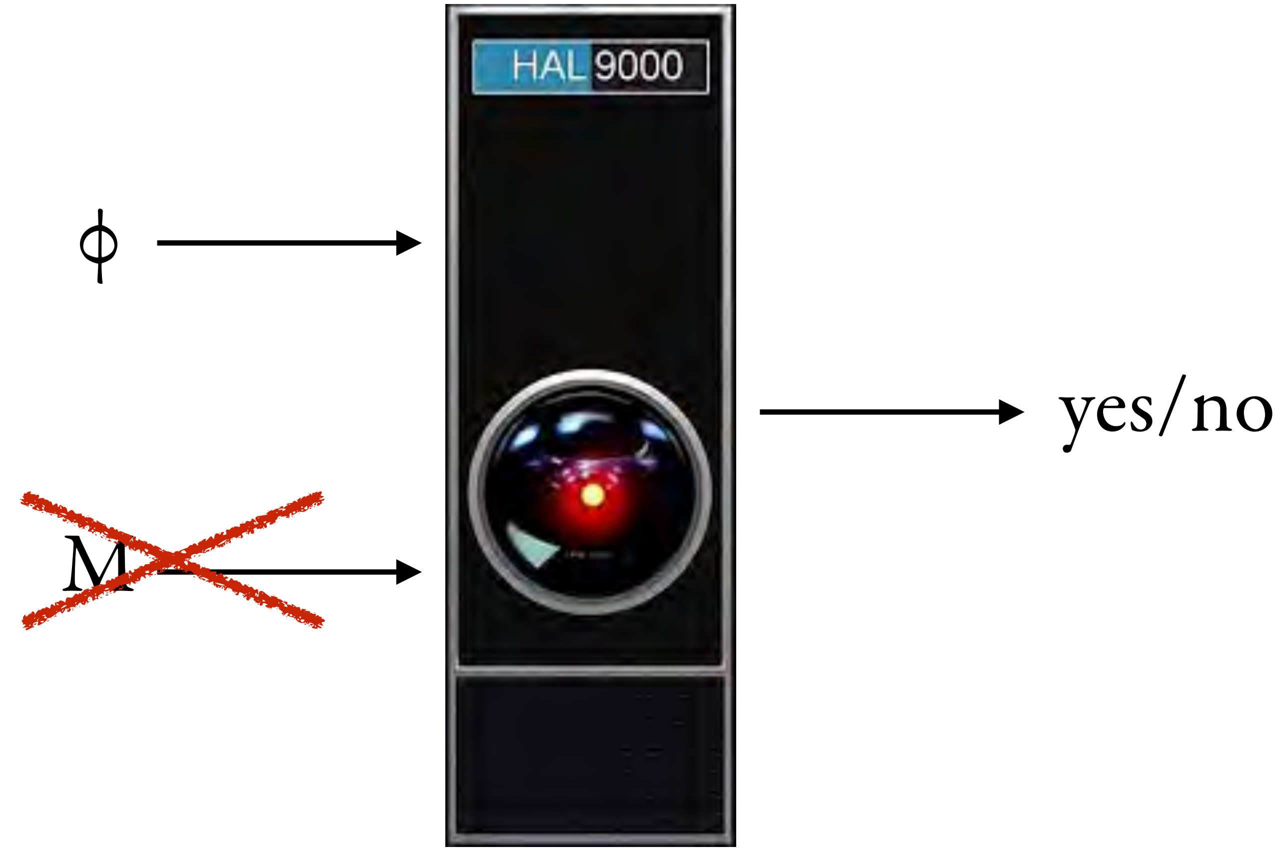
<u>Model-checking</u> problem

input: formula ϕ + model M

output: yes iff ϕ holds on M ($M \vDash \phi$)

Variant: sometimes M is fixed (e.g. when M is infinite), and the only input is ϕ

Algorithms

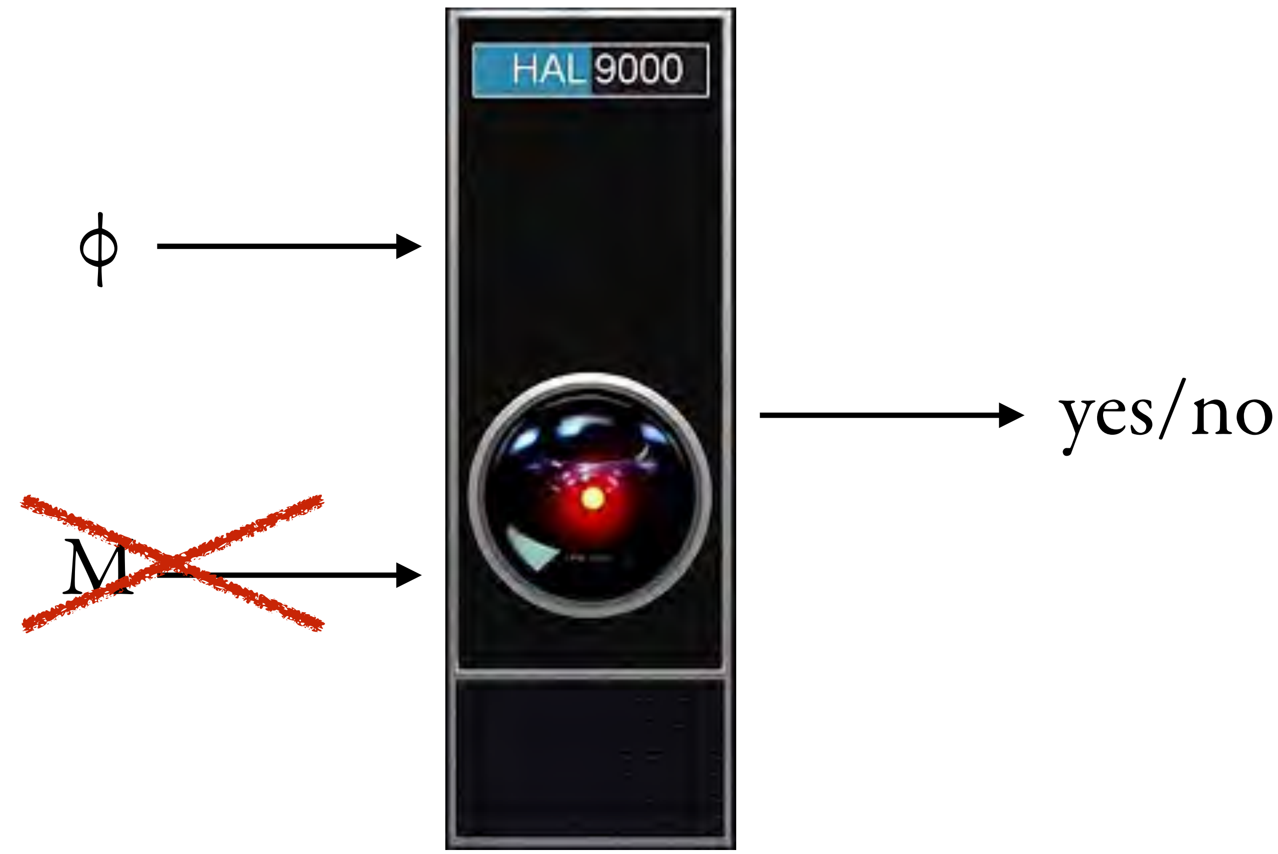


[Validity]{.underline} problem

input: formula ϕ

output: yes iff ϕ holds on *every* model M

Algorithms



<u>Validity</u> problem

input: formula ϕ

output: yes iff ϕ holds on *every* model M

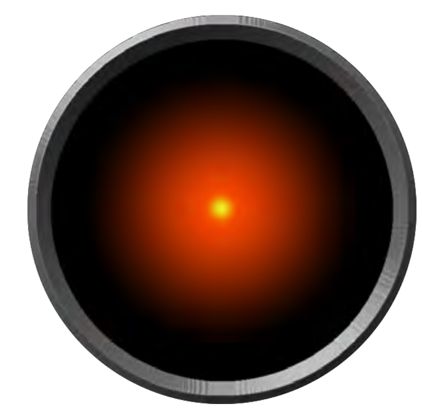
Variant: sometimes M is restricted to range over specific class (e.g. finite models)

Algorithms

B or not B? ϕ $\longrightarrow$

$\longrightarrow$ yes/no

Yes, Dave,
that's valid



<u>Validity</u> problem

input: formula ϕ

output: yes iff ϕ holds on *every* model M

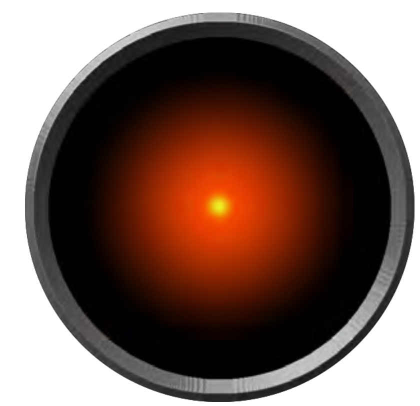
Variant: sometimes M is restricted to range over specific class (e.g. finite models)

Algorithms

B and not B? ϕ →

→ yes/no

No, Dave,
that's not possible



[Satisfiability]{.underline} problem

input: formula ϕ

output: yes iff ϕ holds on *some* model M

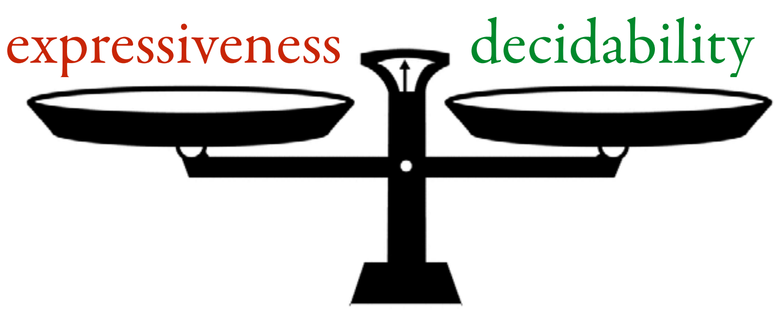
Variant: sometimes M is restricted to range over specific class (e.g. finite models)

Algorithms ...but not only

When studying logics, algorithms are not the only interesting part:

- Expressive power

  Which kinds of properties can be expressed in a given logic?
  Is this logic more/less expressive than this other logic?
  Does it express undecidable properties?



- Succinctness

  How complex it is to express a family of properties?
  Which logic is more succinct?
  Which logic has more efficient algorithms?

- Normal forms, algebraic representations, ...

  Ways of matching syntax and semantics, ease automatic reasoning, etc.