VNU-HUS MAT1206E/3508: Introduction to AI

Limitations of Logic

Hoàng Anh Đức

Bộ môn Tin học, Khoa Toán-Cơ-Tin học
Đại học KHTN, ĐHQG Hà Nội
hoanganhduc@hus.edu.vn

Textbook Basis

These lecture slides were constructed based primarily on the contents of:

- Wolfgang Ertel (2025). *Introduction to Artificial Intelligence*. 3rd. Springer. DOI: 10.1007/978-3-658-43102-0

Contents

The Search Space Problem

- In the search for a proof, depending on the calculus, potentially *there are (infinitely) many ways to apply inference rules at each step*
- Each choice creates a branch; after several steps, *the search space grows explosively*

Because of the search space problem, *automated provers* today can *only prove relatively simple theorems in special domains with few axioms*

The Search Space Problem

	Automated Provers	Human Experts
Number of inferences performed per second	20000	1
Solve difficult problems	slow	fast

Reasons

- Humans use *intuitive calculi* that work on a higher level
  - These intuitive calculi often *carry out many of the simple inferences of an automated prover in one step*
- Humans work with *lemmas*
  - We already know the lemmas are true and *do not need to re-prove* them each time
- Humans use *intuitive meta knowledge* (informal!)
  - Intuition is an important advantage of humans. Without it, we could not solve any difficult problems
- Humans use *heuristics*
  - Heuristics often *shorten the way to the goal*; rarely, they can make the search longer

The Search Space Problem

Problems

- Often humans are *unable to formulate intuitive meta-knowledge verbally*!
- Humans *learn heuristics by experience*

Solution 1

Try to “copy nature”: learn heuristics using machine learning techniques

Solution 2: Interactive Systems

- Computer algebra systems such as Mathematica, Maple, or Maxima can automatically carry out difficult symbolic mathematical manipulations, but proof search is left fully to the human
- Interactive theorem provers, for example Isabelle (`https://isabelle.in.tum.de/`), check and support user-guided proof search

The Search Space Problem

Example 1 (Resolution + Machine Learning Techniques)

A *resolution prover* has, during the search for a proof, *hundreds or more possibilities for resolution steps at each step*, but *only a few lead to the goal*. It would be *ideal* if the prover could *ask an oracle which two clauses it should use in the next step to quickly find the proof*

(1) A *proof-directing module evaluates* the different alternatives for the next step heuristically and *chooses* the alternative with the *best rating*
   - Rating the available clauses by *a function that calculates a value based on the number of literals, the number of positive literals, the complexity of the terms, etc.* for every pair of resolvable clauses

(2) How to *compute* such a *function*?
   - Use machine learning algorithms to learn from successful proofs
   - For each candidate resolution step, describe the two resolvable clauses by *features* such as the number of literals, number of positive literals, and term complexity
   - Feature descriptions from clause pairs used in successful resolution steps are stored as *positive training examples*
   - Feature descriptions from unsuccessful clause pairs are stored as *negative training examples*
   - The machine learning system generates a program for rating future clause pairs heuristically

Decidability and Incompleteness

- There are *correct and complete calculi and theorem provers*
- Any *theorem (i.e., a true statement)* can be *proved in a finite amount of time*
- What if *the statement is not true*?
  - There is *no process* that can *prove or refute any formula from PL1 in finite time*

Theorem 1

The set of valid formulas in first-order predicate logic is semidecidable

- This theorem implies that there are programs (theorem provers) which, given a true (valid) formula as input, determine its truth in finite time
- If the formula is not valid, however, it may happen that the prover never halts

Decidability and Incompleteness

Note

- *Propositional logic decidable* because *the truth table method provides all models of a formula in finite time*
- Evidently predicate logic with quantifiers and nested function symbols is a language somewhat too powerful to be decidable

Exercise 1 ([Ertel 2025], Exercise 4.1, p. 75)

(a) With the following (false) argument, one could claim that PL1 is decidable: *"We take a complete proof calculus for PL1. With it we can find a proof for any true formula in finite time. For every other formula ϕ I proceed as follows: I apply the calculus to $\neg\phi$ and show that $\neg\phi$ is true. Thus ϕ is false. Thus I can prove or refute every formula in PL1."* Find the mistake in the argument and change it so it becomes correct (**Hint:** Separate the cases where ϕ is valid, unsatisfiable, and satisfiable but not valid)

(b) Construct a decision process for formulas in PL1 that are either valid or unsatisfiable

Decidability and Incompleteness

Higher-order Logic

- A *first-order logic* can only *quantify over variables*
- A *second-order logic* can also *quantify over predicates, relations, or functions*
- A *third-order logic* can *quantify over such higher-order objects*
- and so on

Example 2 (Second-order Logic Formula)

- Induction over predicates/properties of natural numbers is already second-order in this sense
- Inductive step: $\forall p\ (p(n) \Rightarrow p(n+1))$ (if p holds for n, then p also holds for $n+1$, where p is a predicate)

Decidability and Incompleteness

Theorem 2 (Gödel incompleteness theorem)

If a formal theory is consistent and as strong as Peano arithmetic, then there are true statements that have no proof within the theory.

- The theorem follows Russell and Norvig's phrasing: "any formal theory as strong as Peano arithmetic" and "true statements that have no proof within the theory" [Russell and Norvig 2021], Sec. 1.2, p. 9, and the consistency condition "If F is consistent" [Russell and Norvig 2021], Sec. 27.1.3, p. 983.
- Recall: *consistent* means no contradiction is provable; *inconsistent* means some A and $\neg A$ are both provable.
- More precisely: if such a theory has mechanically checkable axioms and proofs, then some true statement about the natural numbers is not derivable within that theory.
- "Not provable" is relative to the theory; a stronger theory may prove it.

Decidability and Incompleteness

Note

The deeper background of Theorem 2 is that the more expressive a formal language is, the easier it becomes to write statements that talk about themselves or lead to paradox-like situations.

Example 3 ("Too powerful language")

- A warning example is *naive set comprehension*: if every condition is allowed to define a set, then $M = \{x \mid x \notin x\}$ leads to Russell's paradox.
- The barber story has the same self-reference pattern: a barber shaves exactly those people who do not shave themselves.

Exercise 2 ([Ertel 2025], Exercise 4.2, p. 75)

(a) Given the statement "There is a barber who shaves precisely those people who do not shave themselves." Consider whether this barber shaves himself.

(b) Let $M = \{x \mid x \notin x\}$. Describe this set and consider whether M contains itself.

Dilemma: Very expressive languages can formulate self-reference and paradoxes; sufficiently strong consistent formal systems still face incompleteness.

Example: The Flying Penguin

Goal

We present an example to demonstrate a fundamental problem of logic and possible solution approaches.

1. Tweety is a penguin
2. Penguins are birds
3. Birds can fly

Formalized in PL1, the *knowledge base* KB is:

$$\mathit{penguin}(\mathit{tweety})$$
$$\mathit{penguin}(x) \Rightarrow \mathit{bird}(x)$$
$$\mathit{bird}(x) \Rightarrow \mathit{fly}(x)$$

Exercise 3

(a) Show that $KB \vdash \mathit{fly}(\mathit{tweety})$ (for example, with resolution)

(b) Show that if we add $\mathit{penguin}(x) \Rightarrow \neg \mathit{fly}(x)$ to the knowledge base KB, then $KB \vdash \neg \mathit{fly}(\mathit{tweety})$

Example: The Flying Penguin

- From Exercise 3, both *fly*$($*tweety*$)$ and $\neg$*fly*$($*tweety*$)$ can be derived $\Rightarrow$ *The knowledge base is inconsistent*
- Although we explicitly state that penguins cannot fly, the opposite can still be derived. This is because of *the monotony of PL1*.

Monotonic Logic

A logic is called *monotonic* if, for an arbitrary knowledge base KB and an arbitrary formula ϕ, the set of formulas derivable from KB is a subset of the formulas derivable from $KB \cup \phi$.

Recap

- Evidently the formalization of the flight attributes of penguins is insufficient
- To prevent the formula *fly*$($*tweety*$)$ from being derived, our first attempt is to add new formulas to the KB. This does not work

Example: The Flying Penguin

- We continue by replacing the obviously false statement "(all) birds can fly" in KB with the more exact statement "(all) birds except penguins can fly" and obtain as KB_2 the following clauses:

$$\begin{aligned} \textit{penguin}(\textit{tweety}) \\ \textit{penguin}(x) \Rightarrow \textit{bird}(x) \\ \textit{bird}(x) \wedge \neg\textit{penguin}(x) \Rightarrow \textit{fly}(x) \\ \textit{penguin}(x) \Rightarrow \neg\textit{fly}(x) \end{aligned}$$

- **Problem solved!** We *can now derive* $\neg\textit{fly}(\textit{tweety})$ *but not* $\textit{fly}(\textit{tweety})$, because deriving $\textit{fly}(\textit{tweety})$ would require instantiating the rule with $x = \textit{tweety}$ and proving $\neg\textit{penguin}(\textit{tweety})$, which is not derivable

Example: The Flying Penguin

- A problem arises when we want to add a new bird, say the raven Abraxas (from the German book “The Little Witch”), and obtain KB_3

$$
\begin{aligned}
&\mathit{raven}(\mathit{abraxas})\\
&\mathit{raven}(x) \Rightarrow \mathit{bird}(x)\\
&\mathit{penguin}(\mathit{tweety})\\
&\mathit{penguin}(x) \Rightarrow \mathit{bird}(x)\\
&\mathit{bird}(x) \wedge \neg \mathit{penguin}(x) \Rightarrow \mathit{fly}(x)\\
&\mathit{penguin}(x) \Rightarrow \neg \mathit{fly}(x)
\end{aligned}
$$

- At the moment, we cannot say anything about the flight attributes of Abraxas because we forgot to formulate that ravens are not penguins. Thus we extend KB_3 to KB_4

$$
\begin{aligned}
&\mathit{raven}(\mathit{abraxas})\\
&\mathit{raven}(x) \Rightarrow \mathit{bird}(x)\\
&\mathit{raven}(x) \Rightarrow \neg \mathit{penguin}(x)\\
&\mathit{penguin}(\mathit{tweety})\\
&\mathit{penguin}(x) \Rightarrow \mathit{bird}(x)\\
&\mathit{bird}(x) \wedge \neg \mathit{penguin}(x) \Rightarrow \mathit{fly}(x)\\
&\mathit{penguin}(x) \Rightarrow \neg \mathit{fly}(x)
\end{aligned}
$$

Example: The Flying Penguin

- The fact that ravens are not penguins, which is self-evident to humans, must be explicitly added here
- For the construction of a knowledge base with all 9800 or so types of birds worldwide, it must therefore be specified for every type of bird (except for penguins) that it is not a member of penguins
- More generally, classical PL1 does not treat an unstated fact as false. If a rule needs $\neg P(a)$, the knowledge base must contain enough information to derive $\neg P(a)$ (here, the flying rule needs $\neg \mathit{penguin}(\mathit{abraxas})$, so KB_4 adds $\mathit{raven}(x) \Rightarrow \neg \mathit{penguin}(x)$ to combine with $\mathit{raven}(\mathit{abraxas})$). In large domains, many such negative facts or rules may have to be stated explicitly.

Exercise 4 ([Ertel 2025], Exercise 4.3, p. 75)

Use an automated theorem prover (for example E [Schulz 2002]) for the five Tweety knowledge bases above: the initial KB, the inconsistent extension from Exercise 3(b), KB_2, KB_3, and KB_4. Validate the example's statements.

Example: The Flying Penguin

The *frame problem* asks how a knowledge base can avoid listing, for every action, all facts that stay unchanged unless the action affects them.

Example 4 (An example of the frame problem)

- A blue house is painted red, then afterwards it is red
- However, with the knowledge base

$$\mathit{color}(\mathit{house}, \mathit{blue})$$
$$\mathit{paint}(\mathit{house}, \mathit{red})$$
$$\mathit{paint}(x, y) \Rightarrow \mathit{color}(x, y)$$

  one can derive $\mathit{color}(\mathit{house}, \mathit{red})$
- The old fact $\mathit{color}(\mathit{house}, \mathit{blue})$ is still in the knowledge base, so the house is represented as both blue and red after painting.

Example: The Flying Penguin

- For the *frame problem*, a standard treatment is to represent time or situations explicitly, for example with *situation calculus* [Ertel 2025], p. 73.
  - Plain idea: separate what is true before an action from what is true after it.
- For the default-assumption problem in examples such as Tweety, *non-monotonic logics* have been developed [Ertel 2025], p. 73.
  - Previously drawn conclusions or *default assumptions* can be *withdrawn when new information is added*.
- Despite great effort, these approaches have had *semantic and practical difficulties* as *general-purpose solutions*.
- Another interesting approach for modeling problems such as the Tweety example is *probability theory*.
  - The statement "all birds can fly" is false.
  - A statement something like "almost all birds can fly" is correct.
  - This statement becomes more exact if we give a probability for "birds can fly".

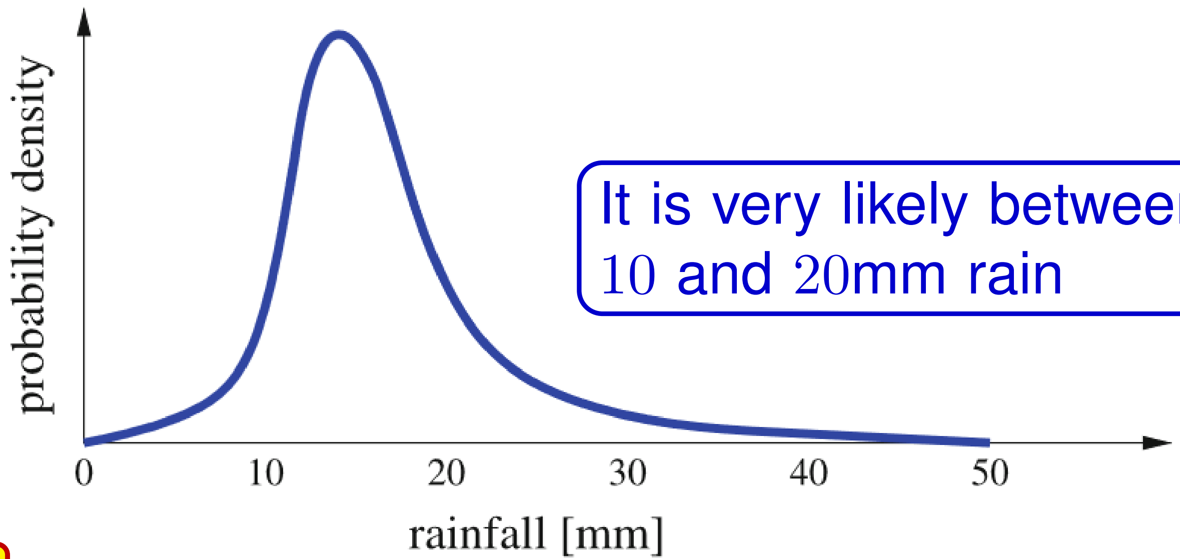
Modelling Uncertainty

- *Two-valued logic* can and should only *model circumstances in which there is true, false, and no other truth values*.
- For many tasks in everyday reasoning, *two-valued logic is therefore not expressive enough*.
  - For example, the rule *bird*$(x) \Rightarrow$ *fly*(x) is *true for almost all birds, but for some it is false*.
- As we already mentioned, to formulate uncertainty, we can use *probability theory*.
  - Informal shorthand: $P($*bird*$(x) \Rightarrow$ *fly*$(x)) = 0.99$ means *"about 99% of birds can fly"*.
  - Later, we will see that here it is better to work with *conditional probabilities* such as $P($*fly*$|$*bird*$) = 0.99$. With the help of *Bayesian networks*, complex applications with many variables can also be modelled.
  - *Fuzzy logic* is one model for *"The weather is nice"*, where strict true/false values are often unnatural.
  - The variable *weather_is_nice* is *continuous with values in* $[0, 1]$. *weather_is_nice* $= 0.7$ then means *"The weather is fairly nice"*.

Modelling Uncertainty

- Probability theory can also model *continuous variables* using *probability densities*.

"There is a high probability that there will be some rain"



Note

This very *general and even visualizable representation* of both types of uncertainty we have discussed, together with *inductive statistics* and the theory of *Bayesian networks*, makes it possible, in principle, *to answer arbitrary probabilistic queries*.

Modelling Uncertainty

Comparison of different formalisms for the modelling of uncertain knowledge

Formalism	Values used	Meaning
Propositional logic	2	truth values: true/false
Fuzzy logic	∞	degrees of truth
Discrete probabilistic logic	n	discrete probability values
Continuous probabilistic logic	∞	*probability values in* $[0, 1]$

- These formalisms are not directly comparable to predicate logic; here we compare simple propositional-style uncertainty formalisms.

References

Ertel, Wolfgang (2025). *Introduction to Artificial Intelligence*. 3rd. Springer. DOI: 10.1007/978-3-658-43102-0.

Russell, Stuart J. and Peter Norvig (2021). *Artificial Intelligence: A Modern Approach*. 4th. Pearson.

Schulz, Stephan (2002). “E–A Brainiac Theorem Prover.” In: *AI Communications* 15.2–3, pp. 111–126. URL: https://wwwlehre.dhbw-stuttgart.de/~sschulz/E/E.html.