VNU-HUS MAT1206E/3508: Introduction to AI

Limitations of Logic

In-class Discussion

Hoàng Anh Đức

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

Contents

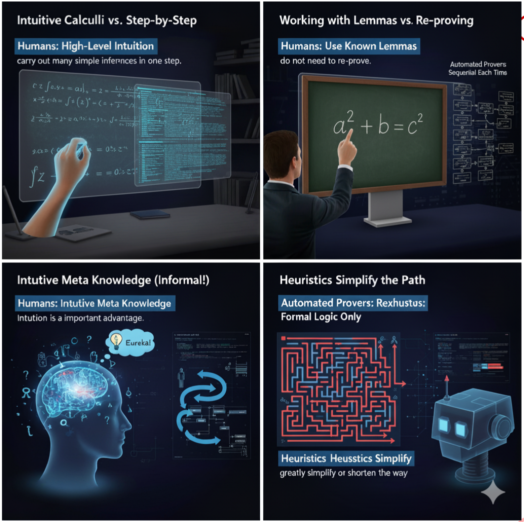
The Search Space Problem

- In the search for a proof, depending on the calculus, potentially *there are (infinitely) many ways to apply inference rules at each step*
- This is the main reason for the *explosive growth of the search space*

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

Human experts can prove theorems that are *out of reach* for *automated provers*



Human experts vs. Automated Provers. Image generated by Gemini AI

The Search Space Problem

Problems?

- Humans' *intuitive meta-knowledge* is difficult to formalize and integrate into automated provers.
- Humans acquire *heuristics through experience*, and these heuristics are hard to transfer to automated provers.

Possible Solutions?

- *Machine learning* techniques can be used to learn heuristics from large datasets of proofs.
- *Hybrid approaches* that combine automated reasoning with human expertise can be explored.

Decidability and Incompleteness

- *Decidability* refers to whether there exists an algorithm that can determine the truth or falsity of any statement in a given logical system.
- *Incompleteness* refers to the fact that in any sufficiently powerful logical system, there are statements that are true but cannot be proven within the system.

Examples?

- The *Halting Problem* is a classic example of an undecidable problem.
- In first-order logic, if a given formula is valid, there is a proof of it (Gödel's Completeness Theorem). However, if the formula is not valid, it may happen that the prover never halts (undecidability). On the other hand, propositional logic is decidable. **[Why?]**
- Gödel's *Incompleteness Theorems* show that in any consistent formal system that is capable of expressing basic arithmetic, there are true statements that cannot be proven within the system.

Decidability and Incompleteness

Note

The deeper background of Gödel's Incompleteness Theorem is that mathematical theories (axiom systems) and, more generally, *languages become incomplete if the language becomes too powerful* (e.g., PL1).

Example 1 ("Too powerful language")

- *Set theory* is so powerful that one can formulate paradoxes (= statements that contradict themselves) with it.
- For example, a paradox in set theory: "The set of all the barbers who all shave those who do not shave themselves"

Example: The Flying Penguin

Knowledge Base (KB)

(1) Tweety is a penguin

(2) Penguins are birds

(3) Birds can fly

(1) *penguin*(*Tweety*)

(2) $\forall x\,(\textit{penguin}(x) \Rightarrow \textit{bird}(x))$

(3) $\forall x\,(\textit{bird}(x) \Rightarrow \textit{fly}(x))$

Query (Q)

Can *Tweety fly*? ($Q = \textit{fly}(\textit{Tweety})$)

Note that $KB \vdash Q$, i.e., Q can be proven from KB **[How?]**

Example: The Flying Penguin

However, in reality, *penguins cannot fly*. That is, we should prevent the derivation of Q from KB

First Attempt

To mimics the reality, we try to add $\forall x\, (\textit{penguin}(x) \Rightarrow \neg \textit{fly}(x))$ to KB.

- After adding this new information, we have $KB \vdash \neg Q$, i.e., $\neg Q$ can be proven from KB **[How?]**
- But Q can still be derived from KB **[How?]**
- Both Q and $\neg Q$ can be derived from $KB \Rightarrow$ *contradiction* $\Rightarrow KB$ is *inconsistent* $\Rightarrow$ *everything can be proven from* KB (principle of explosion)
  - **Reason:** PL1 is *monotonic*, i.e., if $KB \vdash Q$, then $(KB \cup \{R\}) \vdash Q$ for any formula R.

$\Rightarrow$ Does not work!

Example: The Flying Penguin

Second Attempt

- Add “Penguins cannot fly” to the original KB
- Replace the non-realistic rule “Birds can fly” by “Birds except penguins can fly”.

Knowledge Base (KB_2)

(1)	Tweety is a penguin	(1)	*penguin*(*Tweety*)
(2)	Penguins are birds	(2)	$\forall x\,(\textit{penguin}(x) \Rightarrow \textit{bird}(x))$
(3)	Birds except penguins can fly	(3)	$\forall x\,(\textit{bird}(x) \land \neg\textit{penguin}(x) \Rightarrow \textit{fly}(x))$
(4)	Penguins cannot fly	(4)	$\forall x\,(\textit{penguin}(x) \Rightarrow \neg\textit{fly}(x))$

$\Rightarrow$ Problem solved! Now, $\neg Q$ can be derived from KB_2, but Q cannot

Example: The Flying Penguin

But ...

- Whenever we want to add a new bird, we have to specify whether it can fly or not
- This means we have to explicitly specify a rule saying that the newly added bird is not a penguin (because of the rule "Birds except penguins can fly"); otherwise, we have no conclusion about whether the new bird can fly or not
- For the construction of a knowledge base with all 9800 or so types of birds worldwide, this becomes a significant challenge.

In general, for every object in the knowledge base, in addition to its attributes, all of the attributes it does not have must be listed.

Example: The Flying Penguin

Another problem caused by the monotony is the so-called *frame problem*. This happens in complex planning problems in which the world can change

Example 2 (An example of the frame problem)

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

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

  one can derive $\mathit{color}(\mathit{house}, \mathit{red})$
- Additionally, *$\mathit{color}(\mathit{house}, \mathit{blue})$ is already in the knowledge base, which leads to the conclusion that, after painting, the house is both blue and red*

Example: The Flying Penguin

Some Resolving Ideas

- *Non-monotonic reasoning* allows us to withdraw conclusions in the light of new information (i.e., old knowledge can be removed).
- *Probability theory* can be used to handle uncertainty and make decisions based on incomplete information.

Modelling Uncertainty

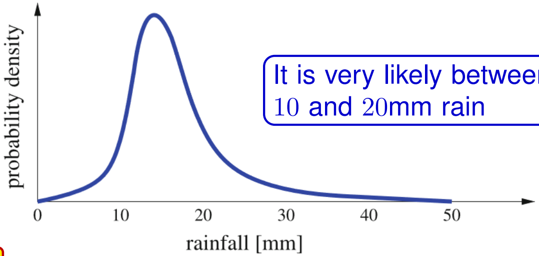
- *Two-valued logic* can and should only *model circumstances in which there is true, false, and no other truth values*.
- For many tasks in everyday reasoning, *two-valued logic is therefore not expressive enough*.
  - For example, the rule $\textit{bird}(x) \Rightarrow \textit{fly}(x)$ is *true for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

Modelling Uncertainty

- Probability theory also offers the possibility of *making statements about the probability of continuous variables*.

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

$$P(\textit{rainfall} = X) = Y$$



It is very likely between 10 and 20mm 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	**Number of truth values**	**Probabilities expressible**
Propositional logic	2	—
Fuzzy logic	∞	—
Discrete probabilistic logic	n	yes
Continuous probabilistic logic	∞	*yes*