

# On Teaching Logic Programming in the Era of Generative AI

Paul Tarau

University of North Texas

PEG'2024

# The Generative AI Disruption of Education

- the Generative AI disruption hits the gaps that have accumulated over the last 2000 years in the educational processes
  - replicating “canned” content
  - emphasis on quantity vs. depth
  - lack of emphasis on actionable generalization and creativity
  - optimizations to reduce the cost of grading rather than feedback on learning
  - outdated evaluation mechanisms of educational processes
- How can we actually benefit from the disruption and improve the educational processes?

# AI disruption of student testing

- The advanced capabilities of AI models make it nearly impossible to detect AI-generated essays or multiple choice problems, rendering traditional plagiarism detection methods ineffective.
- AI-assistance on students' work will significantly disrupt traditional homework practices, raising concerns about cheating and the need for educators to rethink assignments.
- Educators will need to reconsider the purpose and structure of assignments, including essays and problem sets, in light of AI's ability to generate content and solve complex problems.

# The Domain Specific Disruption : Prolog

- LLMs write nice Prolog code when prompted in plain natural language:

*Write a Prolog program that finds the first 5 pairs of twin primes and show the Result it returns.*

```
% Define a predicate to check if a number is prime
is_prime(2).
is_prime(3).
is_prime(N) :-
    N > 3,
    N mod 2 =\= 0,
    \+ has_factor(N, 3).

% Helper predicate to check if a number has a factor
has_factor(N, F) :-
    N mod F =:= 0.
has_factor(N, F) :-
    F * F < N,
    F2 is F + 2,
    has_factor(N, F2).
```

# AI authored Prolog twin primes

```
% Define a predicate to find twin primes
twin_prime(X, Y) :-
    is_prime(X),
    Y is X + 2,
    is_prime(Y).

% Find the first N pairs of twin primes
find_twin_primes(0, _, []) :- !.
find_twin_primes(N, X, [(X, Y) | Rest]) :-
    N > 0,
    twin_prime(X, Y),
    N1 is N - 1,
    X2 is X + 1,
    find_twin_primes(N1, X2, Rest).
find_twin_primes(N, X, Result) :-
    X2 is X + 1,
    find_twin_primes(N, X2, Result).

% Predicate to get the first 5 pairs of twin primes
first_5_twin_primes(Result) :-
    find_twin_primes(5, 2, Result).
```

# The Disruption is here: thoughts on the logic of joining it

- Generative AI-denial is a natural, partly rational, partly emotion-driven reaction of practitioners of symbolic AI, Natural Language Processing and classic Machine Learning that are prime disruption targets of it, as it suddenly deprecates years of hard work on significant academic and industrial achievements.
- But the disruption is here and socially recognized in major ways
  - Nobel prizes in Physics and Chemistry
  - 17K submissions to ICLR'25
  - multi-trillion market caps of Generative AI related companies
- On the positive side: there are natural synergies that Logic Programming and Generative AI have together!
- With focus on the teaching of Prolog and related languages we sketch next a pragmatic path for joining the disruption.

# Engagement strategies with AI Assistants

AI assistants seen as:

- on the spot helpers
- tutors
- code readers
- code writers
- evaluators
- playmates in a gamified learning world

# A basic AI engagement strategy

- The basic engagement
  - Learn underlying concepts, try to solve the problem.
  - If stuck, iterate a few times with alternative ways to look at it.
  - If still stuck, call the AI-assistant.
  - After reading the solution, answer to a set of multiple choice questions about the AI's code.
- This could be automated by the instructor's code asking the LLM to evaluate the answers
- In the case of Prolog, this AI-assisted coding pipeline can be refined by requiring the students to run the AI's code and explain the results of a few spy points by tracing the execution.
- More interactive online engagement on platforms like Canvas engaging in discussion groups on the AI generated code can provide a richer learning experience that is also gradable by the instructor or an AI assistant.



# Gamification

Gamification relies on borrowing elements from the gaming world students are naturally exposed to.

- an assignment requests a small group of students to compare a Prolog problem given by each to a different AI of their choice
- the students will be timed on testing and possibly debugging the AI-provided solutions and gain points, banners etc. for their performance
- the students discuss the merits and demerits of each in a forum with a partial grade originating from their participation in the forum
- a refinement to the grading: allow AI-assistance (seen as the equivalent of “magic spells” in a game) will have a point cost
- like in the gaming world, some extra work to refill the player’s “strength level”, with clear learning benefits in this case!

# Engagement with Natural Language and Logic Programming

- as logic notation originates in natural language, there are conspicuous mappings between verbs and predicates and nominal groups as their arguments, especially relevant when solving knowledge representation problems
- that hints toward learning methods and language constructs easily mapped syntactically and semantically to natural language equivalents

# Natlog as a lightweight natural language inspired Prolog dialect

- A Our Natlog system enforces closeness to Natural Language by a simplified, flat Prolog syntax

*mother of X M: parent of X M, female M.*

*father of X M: parent of X M, male M.*

*grand parent of X GP: parent of X P, parent of P GP.*

*ancestor of X A : parent of X P, parent or ancestor P A.*

*parent or ancestor P P.*

*parent or ancestor P A : ancestor of P A.*

# Natlog as a highly expressive coding tool

- Natlog opens access to Python's rich finite set and coroutining primitives with minimal syntax annotations
- example of Natlog code defining at source level the primitive `findall/3`:

```
stack S : `list S.  
  
push S X : #meth_call S append (X).  
pop S X : `meth_call S pop () X.  
  
findall X G Xs: listof X G S, to_cons_list S Xs.  
  
listof X G S: stack S, collect_ X G S.  
  
collect_ X G S : call G, `copy_term X CX, push S CX, fail.  
collect_ _X _G _S.
```

# DeepLLM: AI-assistance results in the form of a Propositional Horn Clause program

- DeepLLM is a system that automates deep step-by step reasoning in an LLM dialog thread by recursively exploring alternatives (OR-nodes) and expanding details (AND-nodes) up to a given depth
- starting from a single succinct task-specific initiator we steer the automated dialog thread to stay focussed on the task by synthesizing a prompt that summarizes the depth-first steps taken so far
- semantic similarity to ground-truth facts or oracle advice from another LLM instance is used to restrict the search space and validate the traces of justification steps returned as focussed and trustable answers
- Use cases:
  - consequence predictions
  - causal explanations
  - step by step guidance to achieve a goal
  - topic-focussed exploration of scientific literature
  - supporting arguments for a thesis
  - recommendation systems

# Exploring a topic with DeepLLM

Topic to explore:

inductive definitions

Prompter:

- ☒ scientific\_concept\_explorer
- ☐ consequence\_prediction
- ☐ causal\_inference
- ☐ step\_by\_step\_guidance\_to\_achieve\_a\_goal
- ☐ recommender\_system
- ☐ supporting\_arguments\_for\_a\_thesis
- ☐ cons\_and\_pros\_for\_a\_thesis
- ☐ explorer\_of\_negative\_consequences

Activate LLM!

STARTING!

CLAUSES

```
'inductive definitions':-
  'Base case specification';
  'Inductive step formulation';
  'Recursive structure'.
'Base case specification':-
  'Recursive structure initialization',
  'Minimal instance configuration',
  'Initial condition formulation',
  'Boundary value specification',
  'Fundamental case analysis'.
'Inductive step formulation':-
  'Recursive hypothesis application',
  'Base case verification',
  'Mathematical induction principles',
  'Proof structure complexity',
  'Inductive assertion method'.
'Recursive structure':-
  'Hierarchical decomposition models',
  'Recursive function theory',
  'Nested relational structures',
  'Infinite tree automata',
  'Fixed-point semantics'.
```

Figure: Exploring a concept as a Propositional Horn Clause Program

# The Associated Relation Graph

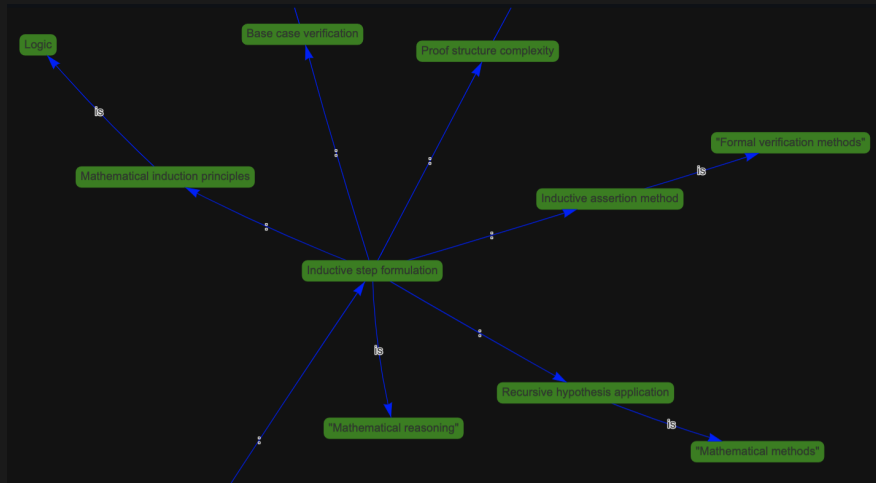


Figure: Visualization of the concept graph related to “inductive definitions”

# Teaching with multi-agent AI systems

- a trend in today's large scale, industrial strength AI API based applications is the the use of multi-agent architectures chaining together:
  - custom code
  - vector store and database calls
  - LLM API calls
- a society of specialized AI agents:
  - Prolog code generator assistant
  - Prolog code critique and bug finder
  - natural language to/from executable Prolog translator
  - conversation animator AI bot as part team projects or forums
  - instructor representative as performance evaluator
- the agents watch and react to unblock students based on the Agents' specific skills and available tools using a publish-subscribe architecture while keeping track of each student progress in a student-specific memory track



# AI-assisted teaching with human in the loop

- real time or asynchronous presence of the human instructor can be naturally integrated when unexpected events are likely to occur in the process
- in a large scale online teaching system this could be an instant real-time intervention by an instructor, possibly assisted by an AI-generated summary of the blockers a given student faces
- at a smaller scale, help ticket-style event queues and blackboards could be made available soliciting instructor intervention with well-defined status details

# Some “out of The box” ideas on on reshaping upcoming AI-assisted Teaching Methodologies

- **Transition from penalty-based to reward-based teaching systems:** In the “training loops” of the today’s user-facing LLMs reward-based engagement has plenty of room to replace or complement today’s penalty-based student-evaluation mechanisms.
- **Encourage emergence of spontaneous social network study groups** among remote online friends exposed to a diversity of teaching systems
- **Facilitate curiosity-driven AI-assisted self study:** with personalized, user aware generative AI (e.g., Character AI’s agents), individual curiosity on a school topic can be explored directly and turned into a learning process while interacting with the customized agent aware of past interactions with its owner.

- **Industry-inspired project-driven learning with AI assistance:**
  - It is by now standard practice in industrial code development to use tools (e.g., Github Copilot) that revise or even create and inject new code in the coder's development pipeline.
  - This is particularly useful in project-based courses like terminal undergrad classes or in-depth graduate classes involving software artifacts as deliverables.
- **Focus on spending teaching time on the “why” besides the “how” in problem solving**
- **Extend the narrow domain-specific teaching to encompass the spectrum of related concepts to which students might have been already exposed**
- **Moving from canned routines and memorized factoids to actionable knowledge outcomes**

# Conclusion

- Generative AI is a major disruptor not just of industrial fields and research fields, including symbolic AI as we know it
- two-fold impact on teaching Prolog: impact on teaching itself + impact on the domain
- in particular, with LLMs able to write clear and well-explained Prolog code, it makes sense to invoke them as helpers at various stages of the instructional process
- this motivates our effort to explore their use including for automated online teaching
- some of our suggested teaching techniques and also some of our “out of the box” ideas will take some time to migrate from the realm of the possible to the realm of the real, but the road to get there is wide open!