The Game Description Language (GDL) Part 2 and Autoformalization

Agnieszka Mensfelt **Kostas Stathis** **Vince Trencsenyi**

https://dicelab-rhul.github.io/Strategic-AI-Autoformalization

ESSAI, 02/07/25

Outline

More advanced GDLs

Motivation

Key Constraints in GDL 1.0

Limitation	Impact
No arithmetic or numeric operations	Scores, counters, and basic math must be encoded using successor relations.
Finite domains only	Infinite or continuous games cannot be represented.
No support for real-time play	Only turn-based games are allowed; no modelling of time or concurrency.
No randomness or chance	Cannot represent dice rolls, shuffled decks, or probabilistic outcomes.
Perfect information only	Hidden information or partial observability (e.g., card games) is not supported.
Lacks metadata	No standard fields for game name, description, or UI elements.

Notable Extensions to GDL 1.0

Addressing GDL 1.0 Limitations

Extension	Added Features
GDL-II	Randomness, imperfect info (e.g.,hidden states, chance nodes)[1].
GDL-Z	Numerical variables, parameters, and numerical comparisons[2].
Temporal GDL	Real-time features[3].
GDL-III	Nested knowledge, common knowledge, and epistemic reasoning.[4]

[1] M. Thielscher, "A general game description language for incomplete information games," in *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, AAAI Press, 2010, pp. 994–999.

[2] M. Mittelmann and L. Perrussel, "Game description logic with integers: A gdl numerical extension," in *Foundations of Information and Knowledge Systems: 11th International Symposium, FoIKS 2020, Dortmund, Germany, February 17–21, 2020, Proceedings 11*, Springer, 2020, pp. 191–210.

[3] J. Kowalski and A. Kisielewicz, "Game description language for real-time games," in *Proceedings of the 8th International Conference on Agents and Artificial Intelligence (ICAART 2016)*, vol. 2, 2016, pp. 494–499.

[4] M. Thielscher, "Gdl-iii: A description language for epistemic general game playing," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pp. 1276–1282.

GDL 1.0

- Originally designed for complete information games.
- A simple extension[5] suffices to generalise it to arbitrary (discrete and finite) games involving **information asymmetry** and **randomness**.

Extension requires two keywords to be introduced (on top of those already seen).

Extended Vocabulary [6]	Explanation
`sees(R, P)`	Player R perceives percept P in the next state.
`random`	A special role used to represent a random player that selects among its legal actions nondeterministically.

[5] M. Thielscher, "A general game description language for incomplete information games," in *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, AAAI Press, 2010, pp. 994–999.

[6] Extensions introduced to support randomness and imperfect information.

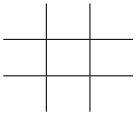
Imperfect-Information Tic-Tac-Toe in GDL-II

Game Description:

We illustrate how to model **imperfect-information games** in **GDL-II** using a variant of Tic-Tac-Toe inspired by *Kriegspiel*, where players cannot see each other's moves.

- Moves to already occupied cells have no effect.
- Players are notified whether their move *succeeded* or *failed*.
- Players act **simultaneously** each turn.
- If both pick the same empty cell, a **coin toss** (random player) resolves the conflict.

Initial State:



A Final State:

X	O	O
O	X	
		X

Blind Tic-Tac-Toe[7]: Indexing and Initialization

```
index(1).
index(2).
index(3).

base(cell(M,N,x)) :- index(M), index(N).
base(cell(M,N,o)) :- index(M), index(N).
base(cell(M,N,b)) :- index(M), index(N).
base(tried(white,M,N)) :- index(M), index(N).
base(tried(black,M,N)) :- index(M), index(N).

init(cell(1,1,b)). init(cell(1,2,b)). init(cell(1,3,b)).
init(cell(2,1,b)). init(cell(2,2,b)). init(cell(2,3,b)).
init(cell(3,1,b)). init(cell(3,2,b)). init(cell(3,3,b)).
```

[7] M. R. Genesereth and M. Thielscher, *General Game Playing* (Synthesis Lectures on Artificial Intelligence and Machine Learning). Morgan & Claypool Publishers, 2014.

Blind Tic-Tac-Toe: Inputs and Legal Moves


```
input(white,mark(M,N)) :- index(M), index(N)
input(black,mark(M,N)) :- index(M), index(N)
input(random,tiebreak(x))
input(random,tiebreak(o))

legal(white,mark(X,Y)):- index(X),index(Y),not true(tried(white,X,Y))

legal(black,mark(X,Y)):- index(X),index(Y),not true(tried(black,X,Y))

legal(random,tiebreak(x))
legal(random,tiebreak(o))
```

Blind Tic-Tac-Toe: Transition Rules (Part 1)

```
% any new attempt to mark a cell is recorded
next(tried(W,M,N)) :- does(W,mark(M,N)).

% all recorded attempts are remembered
next(tried(W,M,N)) :- true(tried(W,M,N)).
```

```
% white marks if black chose a
% different column
next(cell(M,N,x)) :-
  does(white,mark(M,N)),
  true(cell(M,N,b)),
  does(black,mark(J,K)),
  distinct(M,J).
```

```
% white marks if black chose a
% different row
next(cell(M,N,x)) :-
  does(white,mark(M,N)),
  true(cell(M,N,b)),
  does(black,mark(J,K)),
  distinct(N,K).
```

Blind Tic-Tac-Toe: Transition Rules (Part 2)

```
% black: white chose another col
next(cell(M,N,o)) :-
  does(black,mark(M,N)),
  true(cell(M,N,b)),
  does(white,mark(J,K)),
  distinct(M,J).

% black: white chose another row
next(cell(M,N,o)) :-
  does(black,mark(M,N)),
  true(cell(M,N,b)),
  does(white,mark(J,K)),
  distinct(N,K).

marked(M,N):- does(W,mark(M,N)).
```

```
% same cell: random decision
next(cell(M,N,W)) :-
  true(cell(M,N,b)),
  does(white,mark(M,N)),
  does(black,mark(M,N)),
  does(random,tiebreak(W)).

% markings persist
next(cell(M,N,x)):-true(cell(M,N,x)).
next(cell(M,N,o)):-true(cell(M,N,o)).

% blank if unmarked
next(cell(M,N,b)) :-
  true(cell(M,N,b)),
  not marked(M,N).
```

Blind Tic-Tac-Toe: Percepts

```
percept(white,ok).
percept(black,ok).
```

```
% R marks blank, O picks other col
sees(R,ok) :-
  does(R,mark(M,N)),
  true(cell(M,N,b)),
  does(O,mark(J,K)),
  distinct(M,J).

% R marks blankm O picks other row
sees(R,ok) :-
  does(R,mark(M,N)),
  true(cell(M,N,b)),
  does(O,mark(J,K)),
  distinct(N,K).
```

```
% white marks blank & wins
    tiebreak
sees(white,ok) :-
  does(white,mark(M,N)),
  true(cell(M,N,b)),
  does(random,tiebreak(x)).

% black marks blank & wins
    tiebreak
sees(black,ok) :-
  does(black,mark(M,N)),
  true(cell(M,N,b)),
  does(random,tiebreak(o)).
```

Blind Tic-Tac-Toe: Terminal and Goal Conditions

```
terminal :- line(x).
terminal :- line(o).
terminal :- not open.

open :- true(cell(M,N,b)).

goal(white,100) :- line(x), not line(o).
goal(white, 50) :- line(x), line(o).
goal(white, 50) :- not open, not line(x), not line(o).
goal(white, 0)  :- not line(x), line(o).

goal(black,100) :- not line(x), line(o).
goal(black, 50) :- line(x), line(o).
goal(black, 50) :- not open, not line(x), not line(o).
goal(black, 0)  :- line(x), not line(o).
```

Extending Our Framework for GDL-II

To faithfully capture the expressive power of **GDL-II**, our framework must support:

- **Simultaneous moves** — multiple players acting in the same turn.
- see/2 — to incorporate **percepts** and model information asymmetry.
- A **random player** — for representing stochastic elements like dice rolls or shuffled cards.

We begin by addressing how our solver can use simultaneous moves.

Revised Solver: Domain-Independent Part

Game Execution 'Semantics'

Revised solver:

```
game(F, F) :-
    final(F).

game(S, F) :-
    not final(S),
    legal(sim(Ms), S),
    game(do(sim(Ms), S), F).
```

Simultaneous moves:

`sim(Ms)` is now a list of simultaneous moves, defined specifically for each game.

What changes?

SC as before, what changes is:

- legal/2 - to generate simultaneous move combinations;
- effects/3 - to apply simultaneous move effects;
- abnormal/3 - to block state properties from persisting.

We will also need to use them in sees/3 definitions, that are new.

Blind Tic-Tac-Toe: Indexing and Initialization in our Framework

```
index(1).
index(2).
index(3).

initial(s0).
initially(cell(1,1,b), s0).
initially(cell(1,2,b), s0).
initially(cell(1,3,b), s0).
initially(cell(2,1,b), s0).
initially(cell(2,2,b), s0).
```

```
initially(cell(2,3,b), s0).
initially(cell(3,1,b), s0).
initially(cell(3,2,b), s0).
initially(cell(3,3,b), s0).
initially(player(p1), s0).
initially(player(p2), s0).
initially(player(p3), s0).
initially(role(p1,white), s0).
initially(role(p2,black), s0).
initially(role(p3,random), s0).
```

Blind Tic-Tac-Toe: Legal and Possible Moves in Our Framework

```
legal(sim([move(P1,M1),
           move(P2,M2),
           move(P3,M3)
   ]), S):-
   holds(role(P1, white), S),
   holds(role(P2, black), S),
   holds(role(P3, random), S),
   legal(move(P1,M1), S),
   legal(move(P2,M2), S),
   legal(move(P3,M3), S).

legal(move(P,mark(X,Y)), S):-
  possible(move(P, mark(X,Y)), S),
  not holds(tried(P,X,Y), S).
```

```
legal(move(P,tiebreak(M)),S):-
  possible(move(P,tiebreak(M)),S).

possible(move(P,mark(M,N)), S) :-
  index(M),
  index(N),
  holds(player(P), S),
  not holds(role(P, random), S).

possible(move(P,tiebreak(x)), S):-
  holds(role(P,random), S).

possible(move(P,tiebreak(o)), S):-
  holds(role(P,random), S).
```

Blind Tic-Tac-Toe: Transition Rules in our framework (Part 1)

```
% any new attempt to mark a cell is recorded and persist in next
%states as no abnormal/3 definitions for it.

effect(tried(P,M,N), move(P,mark(M,N)), S).
```

```
% black chose another col/row
effect(cell(M,N,x)), sim(Ms), S):-
  holds(role(P1, white), S),
  member(move(P1, mark(M,N)), Ms),
  holds(cell(M,N,b)), S),
  holds(role(P2, black), S),
  member(move(P2,mark(J,K)), Ms),
  (distinct(M,J); distinct(N,K)).
```

```
% white chose another col/row
effect(cell(M,N,o)), sim(Ms), S):-
  holds(role(P1, black), S),
  member(move(P1, mark(M,N)), Ms),
  holds(cell(M,N,b)), S),
  holds(role(P2, white), S),
  member(move(P2,mark(J,K)), Ms),
  (distinct(M,J); distinct(N,K)).
```

Blind Tic-Tac-Toe: Transition Rules (Part 2)

```
% same cell: random decision
effect(cell(M,N,W), sim(Ms), S) :-
  holds(cell(M,N,b)), S),
  holds(role(P1, white), S),
  member(move(P1, mark(M,N)), S),
  holds(role(P2, black), S),
  member(move(P2,mark(M,N)), S),
  holds(role(P3, random), S),
  member(move(P3,tiebreak(W)), S).

% blank does not persits if marked
abnormal(cell(M,N,b), sim(Ms), S) :-
  holds(cell(M,N,b)), S),
  member(move(P, mark(M, N)), Ms).
```

Blind Tic-Tac-Toe: Percepts in our framework

```
% players pick diff col
sees(percept(P1,ok),sim(Ms),S):-
  holds(player(P1), S),
  member(move(P1, mark(M,N)), Ms),
  holds(cell(M,N,b), S),
  holds(player(P2), S),
  distinct(P1, P2),
  member(move(P2, mark(J,K)), Ms),
  distinct(M, J).

% white wins tiebreaks
sees(percept(P1,ok),sim(Ms),S):-
  holds(role(P1, white), S),
  member(move(P1, mark(M,N)), Ms),
  holds(cell(M,N,b), S),
  holds(role(P, random), S),
  member(move(P,tiebreak(x)), Ms).
```

```
% players pick diff row
sees(percept(P1,ok), sim(Ms), S) :-
  holds(player(P1), S),
  member(move(P1, mark(M,N)), Ms),
  holds(cell(M,N,b), S),
  holds(player(P2), S),
  distinct(P1, P2),
  member(move(P2, mark(J,K)), Ms),
  distinct(N, K).

% black wins tiebreak
sees(percept(P1,ok), sim(Ms), S):-
  holds(role(P1, black), S),
  member(move(P1, mark(M,N)), Ms),
  holds(cell(M,N,b), S),
  holds(role(P, random), S),
  member(move(P,tiebreak(o)), Ms).
```

Blind Tic-Tac-Toe: Terminal and Goal Conditions

```
final(S) :- line(x,S).
final(S) :- line(o,S).
final(S) :- not open(S).

holds(goal(P,100),S) :-
  holds(role(P, white), S),
  line(x,S),
  not line(o,S).

holds(goal(P, 0), S) :-
  holds(role(P, white), S),
  not line(x,S),
  line(o,S).
```

```
open(S) :-
  holds(cell(M,N,b),S).

holds(goal(P,50),S) :-
  holds(player(P),S),
  not holds(role(P, random), S),
  not line(x,S),
  not line(o,S).

% Payoffs for black left as an
% exercise to the reader.
%
% ...
```

Reflection on extending our framework towards GDL-II

- We have shown how to extend our GDL-based framework to capture features of both **GDL 1.0** and **GDL-II**, enabling the modelling of games with **incomplete information** and **randomness**.
- However, a key difference is that in our framework, **moves are always hypothetical**, following the style of the **situation calculus**.
- For more complex interactions, we must eventually incorporate **temporal reasoning** over events, akin to what is provided by the **event calculus**.
- There is a simple **natural interpretation** of our formulation, using the event calculus ontology, which we illustrate in the next slide.

Event Calculus Version[10] of Solver

Game execution (temporal):

```
game(T, T) :-
    final(T).

game(T, F) :-
    not final(T),
    legal(M, T),
    game(T+1, F).
```

Time T: 0, 1, 2, ...

Event calculus like holds/2:

```
holds(F, T) :-
    initial(T),
    initially(F, T).

holds(F, T+1) :-
    happens(M, T),
    initiates(F, M, T).
holds(F, T+1) :-
    holds(F, T),
    not terminates(F, A, T).
```

Moves happen at the **same time step**, without assuming player ordering.

[10] R. Kowalski and F. Sadri, "Reconciling the event calculus with the situation calculus," *Journal of Logic Programming*, vol. 31, no. 1–3, pp. 39–58, 1997.

Why Extend to the Event Calculus?

- An **event calculus** formulation would bring several key advantages:
  - Natural support for **concurrent execution**.
  - Ability to maintain and query **histories of state properties**.
  - Better suited for **parallel implementations**.
  - Greater **generality** for modelling diverse dynamic systems.
- However, several challenges remain:
  - Some aspects require **further theoretical and implementation work**. E.g. past vs future like in LTL.
  - Tight integration will be needed with the existing **Game Manager**.

Recap

mainly on efficiency Considerations

- Focused on **simple games**, but more complex ones may use:
  - EC, and efficient versions of it[11],[12] or **depth-limited search** with max depth.
  - **ASP**[13] or **Model Checking**[14] if online use is feasible.
- Closer to **multi-agent adversarial planning**[15] than classical single-agent plans - worth further study.
- Did not address uncertainty - can **probabilistic logic programming**[16] help?
- Can agents use **LLMs to autoformalize games**?

[11] A. Artikis, M. Sergot, and G. Paliouras, "An event calculus for event recognition," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 4, pp. 895–908, 2015.

[12] Ö. Kafalı, A. E. Romero, and K. Stathis, "Agent-oriented activity recognition in the event calculus: An application for diabetic patients," *Computational Intelligence*, vol. 33, no. 4, pp. 899–925, 2017.

[13] M. Thielscher, "Answer set programming for single-player games in general game playing," in *Logic Programming*, Springer Berlin Heidelberg, 2009, pp. 327–341.

[14] T. Chen, V. Forejt, M. Kwiatkowska, *et al.*, "Prism-games: A model checker for stochastic multi-player games," in *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, vol. 7795, Springer, 2013, pp. 185–191.

[15] V. Vie, R. Sheatsley, S. Beyda, *et al.*, *Adversarial plannning*, 2022. arXiv: 2205.00566 [cs.CR].

[16] L. D. Raedt, A. Kimmig, and H. Toivonen, "Problog: A probabilistic prolog and its application in link discovery," in *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, 2007, pp. 2462–2467.

Autoformalization

What is Formalization?

What is Formalization?

Formalization

The process of translating informal language into a formal language with precisely defined syntax and rules of inference, suitable for mechanical verification or automated reasoning.

To Formalize or Not to Formalize

Why formalize?

To Formalize or Not to Formalize

Why formalize?

- Removes ambiguity

To Formalize or Not to Formalize

Why formalize?

- Removes ambiguity
- Ensures rigour

To Formalize or Not to Formalize

Why formalize?

- Removes ambiguity
- Ensures rigour
- Provides a foundation for formal verification, automated reasoning, and symbolic AI

To Formalize or Not to Formalize

Why formalize?

- Removes ambiguity
- Ensures rigour
- Provides a foundation for formal verification, automated reasoning, and symbolic AI

Why not formalize?

To Formalize or Not to Formalize

Why formalize?

- Removes ambiguity
- Ensures rigour
- Provides a foundation for formal verification, automated reasoning, and symbolic AI

Why not formalize?

- Some concepts may resist precise formalization

To Formalize or Not to Formalize

Why formalize?

- Removes ambiguity
- Ensures rigour
- Provides a foundation for formal verification, automated reasoning, and symbolic AI

Why not formalize?

- Some concepts may resist precise formalization
- Formalization can be labour-intensive and slow

To Formalize or Not to Formalize

Why formalize?

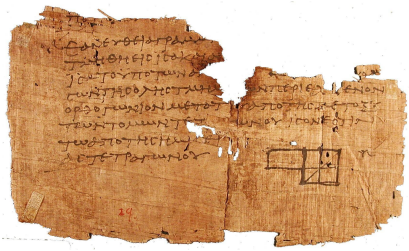
- Removes ambiguity
- Ensures rigour
- Provides a foundation for formal verification, automated reasoning, and symbolic AI

Why not formalize?

- Some concepts may resist precise formalization
- Formalization can be labour-intensive and slow
- Could accelerate advanced AI capabilities — raising alignment and safety concerns

Beginnings of Formalization

- **Babylonian and Egyptian mathematics (c. 3000–300 BCE)**: procedural, rule-based but not axiomatic
- **Greek mathematics (Euclid, c. 300 BCE)**: systematic axiomatic method in *Elements*
- **Indian logic (Nyaya, between 6th century BCE and 2nd century CE)** and **Islamic mathematics (Al-Khwarizmi, c. 800 CE)**: early algorithmic and logical frameworks
- **Chinese mathematics (c. 200 BCE – 1300 CE)**: rigorous algebraic methods, root extraction, early symbolic manipulation



A fragment of Euclid's Elements[a]

[a] en.wikipedia.org/wiki/Euclid%27s_Elements

What is a Formal System?

- A **formal system** is a precisely defined structure used to derive conclusions from axioms using rules of inference.
- It consists of:
  - **Alphabet**: a finite set of symbols
  - **Syntax**: rules for forming well-formed formulas (WFFs)
  - **Axioms**: formulas accepted as starting truths
  - **Inference rules**: logical rules to derive new formulas

What is a Formal System?

- A **formal system** is a precisely defined structure used to derive conclusions from axioms using rules of inference.
- It consists of:
  - **Alphabet**: a finite set of symbols
  - **Syntax**: rules for forming well-formed formulas (WFFs)
  - **Axioms**: formulas accepted as starting truths
  - **Inference rules**: logical rules to derive new formulas

Why it matters for AI

Formal systems are the backbone of automated theorem proving and symbolic reasoning.

From Logic to Formal Systems

- **Leibniz (17th c.)**: vision of a universal logical language (*characteristica universalis*) and method of calculation (*calculus ratiocinator*)
- **Boole (1854)**: an algebraic system for logic; first step toward symbolic logic
- **Frege (1879)**: first modern formal logical system in *Begriffsschrift*
- **Peano, Hilbert, Russell**: formalization of arithmetic, logic, and geometry

> *The only way to rectify our reasonings is to make them as tangible as those of the Mathematicians, so that we can find our error at a glance, and when there are disputes among persons, we can simply say: Let us calculate, without further ado, to see who is right.*[17]

[17] M. Kulstad and L. Carlin, "Leibniz's philosophy of mind,", 1997.

Hilbert's 23 Problems (1900)

- Published in 1900
- A list of **23 major open problems** in mathematics
- Profoundly influenced 20th-century mathematical research

"Wir müssen wissen. Wir werden wissen."
"We must know. We will know."
— David Hilbert, Königsberg, 1930

David Hilbert (1862–1943)

Hilbert's Program (1920s)

Hilbert's Program (1920s):

- Goal: Establish a complete and consistent foundation for all of mathematics by formalizing it within a symbolic system
- Represent all mathematical truths using a formal language with fixed syntax and inference rules
- Prove that the system is **consistent** — i.e., that it cannot derive both a statement and its negation
- Use **finitary methods** (concrete, algorithmic reasoning) to verify this consistency

Hilbert's Program (1920s)

Hilbert's Program (1920s):

- Goal: Establish a complete and consistent foundation for all of mathematics by formalizing it within a symbolic system
- Represent all mathematical truths using a formal language with fixed syntax and inference rules
- Prove that the system is **consistent** — i.e., that it cannot derive both a statement and its negation
- Use **finitary methods** (concrete, algorithmic reasoning) to verify this consistency

Kurt Gödel shattering Hilbert's dream, memeified.

Hilbert's Program (1920s)

Hilbert's Program (1920s):

- Goal: Establish a complete and consistent foundation for all of mathematics by formalizing it within a symbolic system
- Represent all mathematical truths using a formal language with fixed syntax and inference rules
- Prove that the system is **consistent** — i.e., that it cannot derive both a statement and its negation
- Use **finitary methods** (concrete, algorithmic reasoning) to verify this consistency

Gödel's Incompleteness Theorems (1931):

- **1st**: Any consistent formal system that is powerful enough to express basic arithmetic is incomplete — there are true statements in the system that cannot be proved within it.
- **2nd**: Such a system cannot demonstrate its own consistency from within itself.

Kurt Gödel shattering Hilbert's dream, memeified.

Formalization in the Origins of AI

John McCarthy:

- Saw intelligence as the ability to reason formally about actions, goals, and knowledge
- Advocated using mathematical logic to represent common sense and plan actions

Allen Newell and Herbert Simon:

- Pioneered the idea that intelligent behavior can be modeled as symbolic manipulation
- Created early theorem provers (e.g., Logic Theorist, 1956)

We shall therefore say that a program has common sense if it automatically deduces for itself a sufficiently wide class of immediate consequences of anything it is told and what it already knows[18]

[18] J. McCarthy, *Programs with common sense*, 1959.

Formalization in the Origins of AI

John McCarthy:

- Saw intelligence as the ability to reason formally about actions, goals, and knowledge
- Advocated using mathematical logic to represent common sense and plan actions

Allen Newell and Herbert Simon:

- Pioneered the idea that intelligent behavior can be modeled as symbolic manipulation
- Created early theorem provers (e.g., Logic Theorist, 1956)

We shall therefore say that a program has common sense if it automatically deduces for itself a sufficiently wide class of immediate consequences of anything it is told and what it already knows[18]

[18] J. McCarthy, *Programs with common sense*, 1959.

Proof Assistants

- **Proof assistants** are software tools that help users write and check formal proofs using formal logic and rules of inference.
- Common proof assistants:
  - **Coq** (started in 1984)
  - **Isabelle** (1986), **Isabelle/HOL** (early 1990s)
  - **HOL Light** (1996)
  - **Lean** (2013)
- Used in:
  - Formalizing mathematics
  - Software and hardware verification

Why Autoformalize?

- Speeds up adoption of formal methods in mathematics and engineering.
- **Formalization is rigorous, but slow and labour-intensive, e.g.:**
  - **Kepler Conjecture (Flyspeck project)**[19]: $\geqslant$ 7 years
  - **Odd-Order Theorem**[20]: 6 years
- Autoformalization could reduce this effort by orders of magnitude.
- **Enables more reliable AI**: Formal methods can make AI systems provably correct, interpretable, and safer.

[19] T. Hales, M. Adams, G. Bauer, *et al.*, "A formal proof of the kepler conjecture," in *Forum of mathematics, Pi*, Cambridge University Press, vol. 5, 2017, e2.

[20] G. Gonthier, A. Asperti, J. Avigad, *et al.*, "A machine-checked proof of the odd order theorem," in *International conference on interactive theorem proving*, Springer, 2013, pp. 163–179.

From Logic to Autoformalization: A Timeline

- **1950s–60s: John McCarthy**, **Newell & Simon** — logic as the foundation of intelligence; first theorem provers
- **1980s–2000s:** Rise of **automated reasoning**, **SAT/SMT solvers**, formal methods in software verification
- **2010s: Proof assistants** like **Coq** and **Isabelle** mature; used in math and engineering
- **2020s: Autoformalization** with LLMs — informal mathematics translated into formal code (e.g., Lean + GPT)

From Logic to Autoformalization: A Timeline

- **1950s–60s: John McCarthy**, **Newell & Simon** — logic as the foundation of intelligence; first theorem provers
- **1980s–2000s:** Rise of **automated reasoning**, **SAT/SMT solvers**, formal methods in software verification
- **2010s: Proof assistants** like **Coq** and **Isabelle** mature; used in math and engineering
- **2020s: Autoformalization** with LLMs — informal mathematics translated into formal code (e.g., Lean + GPT)

The long arc of symbolic AI is converging with modern machine learning.

Early Visions of Autoformalization: Hao Wang (1960s)

> *The original aim of the writer was to take mathematical textbooks such as Landau on the number system, Hardy-Wright on number theory, Hardy on the calculus, Veblen-Young on projective geometry, the volumes by Bourbaki, as outlines and make the machine formalize all the proofs (fill in the gaps).*[21]

[21] H. Wang, "Toward mechanical mathematics," *IBM Journal of research and development*, vol. 4, no. 1, pp. 2–22, 1960.

Modern approaches: Autoformalization with Large Language Models[22]

- Autoformalization of informal math statements to formal systems (Isabelle/HOL)
- Few-shot prompting of LLMs on Math Competition problems
- Evaluation: BLEU score, manual evaluation on a random sample of 150 problems
- Training a neural theorem prover on the autoformalized theorems

[22] Y. Wu, A. Q. Jiang, W. Li, *et al.*, "Autoformalization with Large Language Models," en, *Advances in Neural Information Processing Systems*, vol. 35, pp. 32 353–32 368, Dec. 2022.

Case Study[23]

Natural Language Statement:

Prove that there is no function f from the set of non-negative integers into itself such that $f(f(n)) = n + 1987$ for every n.

Codex Isabelle Output:

```
theorem
fixes f :: "nat \<Rightarrow> nat"
assumes "\<forall> n. f (f n) = n + 1987"
shows False
```

Analysis:

- Correct syntactic and semantic formalization.

[23] Y. Wu, A. Q. Jiang, W. Li, *et al.*, "Autoformalization with Large Language Models," en, *Advances in Neural Information Processing Systems*, vol. 35, pp. 32 353–32 368, Dec. 2022.

Case Study 2^{24}

Natural Language Statement:

When all the girls at Madeline's school line up in rows of eight, there are seven left over. If instead they line up in rows of four, how many are left over? The final answer is 3.

PaLM Isabelle Output:

```
theorem
fixes n :: nat
assumes "n mod 8 = 7"
shows "n mod 4 = 3"
```

Analysis:

- Correct formalization of a grade school mathematical problem.

[24] Y. Wu, A. Q. Jiang, W. Li, *et al.*, "Autoformalization with Large Language Models," en, *Advances in Neural Information Processing Systems*, vol. 35, pp. 32 353–32 368, Dec. 2022.

Experimental Setup[25]

- **Dataset:** A subset of miniF2F dataset of mathematical competition statements
- **Target formal system:** Isabelle/HOL
- **Models tested:** Codex, PaLM
- Evaluation: BLEU, manual evaluation

Prompting strategy:

- Few-shot prompt with informal-formal examples

[25] Y. Wu, A. Q. Jiang, W. Li, *et al.*, "Autoformalization with Large Language Models," en, *Advances in Neural Information Processing Systems*, vol. 35, pp. 32 353–32 368, Dec. 2022.

Quantitative Results[26]

- BLEU scores

Model	Algebra	Number Theory
PaLM 8B	31.49	22.10
PaLM 64B	43.13	31.43
PaLM 540B	50.30	36.16
Codex	57.13	43.33

- Manual evaluation:
  - Codex formalization accuracy: **25.3%**
- Fine-tuning prover on autoformalized examples
- MiniF2F benchmark accuracy improvement: 29.6% → 35.2%

[26] Y. Wu, A. Q. Jiang, W. Li, *et al.*, "Autoformalization with Large Language Models," en, *Advances in Neural Information Processing Systems*, vol. 35, pp. 32 353–32 368, Dec. 2022.

Conclusions[27]

- **LLMs can generate formal statements** from informal math problems (e.g., in Isabelle).
- **Larger models perform better**: BLEU scores and manual accuracy improve with scale.
- **Room for improvement**: 25% of statements were semantically correct
- **Useful for downstream provers**: Generated statements improved neural theorem proving performance.
- **Challenges remain**:
  - Majority of outputs still contain errors
  - BLEU does not capture semantic correctness

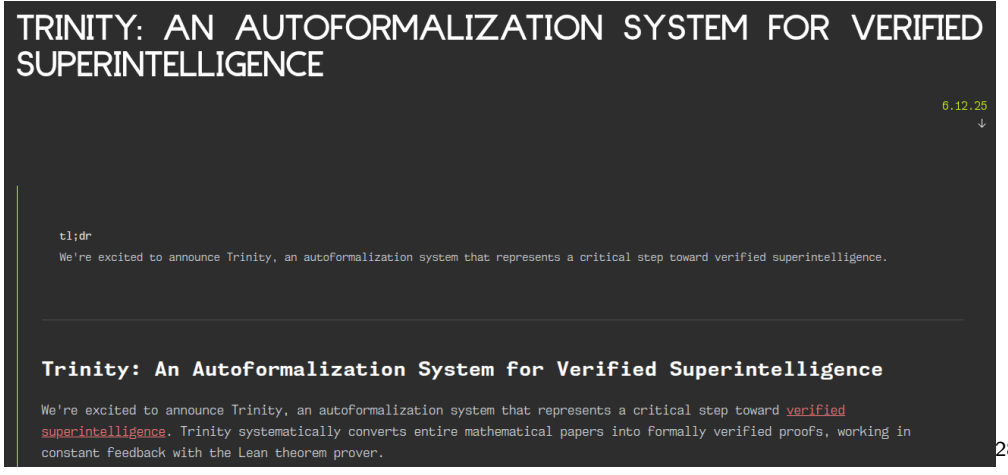
[27] Y. Wu, A. Q. Jiang, W. Li, *et al.*, "Autoformalization with Large Language Models," en, *Advances in Neural Information Processing Systems*, vol. 35, pp. 32 353–32 368, Dec. 2022.

Related (but Distinct) Work

- **Neural Theorem Proving**
  - Goal: Guide or automate proof search within formal systems using neural networks
  - Example:
    - *AlphaGeometry* (DeepMind, 2024): Solves formal geometry problems using neural-symbolic reasoning
  - **Focus:** Given formal input, generate a formal proof
- **Proof Synthesis**
  - Automatically generate formal proofs from given statements or specs
  - Can use symbolic search, program synthesis, or ML techniques
  - **Focus:** From formal spec to formal proof — assumes formal input

Autoformalization: starting from informal natural language

Trinity



TRINITY: AN AUTOFORMALIZATION SYSTEM FOR VERIFIED SUPERINTELLIGENCE

6.12.25

tl;dr

We're excited to announce Trinity, an autoformalization system that represents a critical step toward verified superintelligence.

Trinity: An Autoformalization System for Verified Superintelligence

We're excited to announce Trinity, an autoformalization system that represents a critical step toward verified superintelligence. Trinity systematically converts entire mathematical papers into formally verified proofs, working in constant feedback with the Lean theorem prover.

[28]

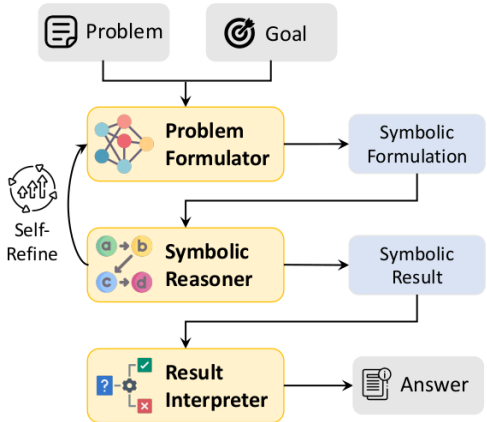
[28] https://www.morph.so/blog/trinity

LOGIC-LM[29]

- **LLMs struggle with complex logical problems** — prone to unfaithful reasoning
- **Logic-LM design:**
  1. **Problem Formulator:** LLM → symbolic representation
  2. **Symbolic Solver:** deterministic engine
  3. **Result Interpreter:** map symbolic output → final answer
  4. **Self-Refinement:** use solver errors to revise formulation

[29] L. Pan, A. Albalak, X. Wang, *et al.*, "Logic-lm: Empowering large language models with symbolic solvers for faithful logical reasoning," in *Findings of the Association for Computational Linguistics: EMNLP 2023*, 2023, pp. 3806–3824.

LOGIC-LM: Overview[30]



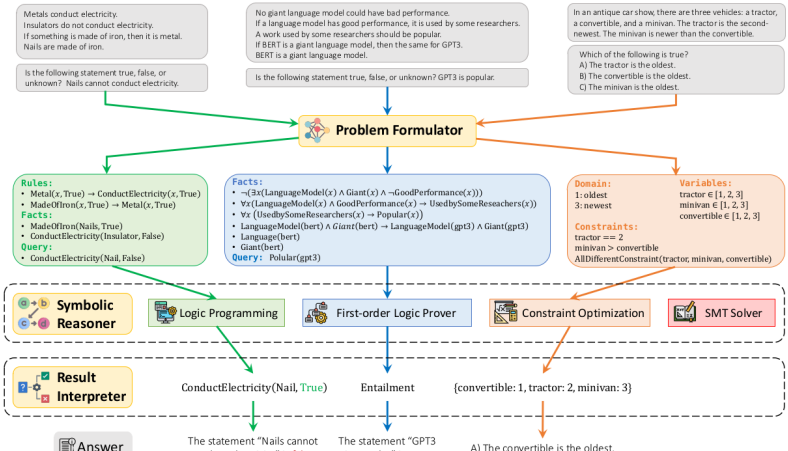
[30] L. Pan, A. Albalak, X. Wang, *et al.*, "Logic-lm: Empowering large language models with symbolic solvers for faithful logical reasoning," in *Findings of the Association for Computational Linguistics: EMNLP 2023*, 2023, pp. 3806–3824.

Datasets & Solvers[31]

Problem	Formulation	Example		Solver	Dataset
		NL Sentence	Symbolic Formulation		
Deductive Reasoning	LP	If the circuit is complete and the circuit has the light bulb then the light bulb is glowing.	`Complete(Circuit, True)`$\wedge$ `Has(Circuit, LightBulb)` $\rightarrow$ `Glowing(LightBulb, True)`	Pyke	ProntoQA, ProofWriter
First-Order Logic	FOL	A Czech person wrote a book in 1946.	$\exists x_2 \exists x_1$(`Czech`$(x_1) \wedge$ `Author`(x_2, x_1) $\wedge$`Book`$(x_2) \wedge$ `Publish`$(x_2, 1946)$)	Prover9	FOLIO
Constraint Satisfaction	CSP	On a shelf, there are five books. The blue book is to the right of the yellow book.	`blue_book` $\in \{1, 2, 3, 4, 5\}$ `yellow_book` $\in \{1, 2, 3, 4, 5\}$ `blue_book` $>$ `yellow_book`	python-constraint	LogicalDeduction
Analytical Reasoning	SAT	Xena and exactly three other technicians repair radios	`repairs(Xena, radios)` $\wedge$ `Count([t:technicians], t` $\neq$ `Xena` $\wedge$ `repairs(t, radios))) == 3)`	Z3	AR-LSAT

[31] L. Pan, A. Albalak, X. Wang, *et al.*, "Logic-lm: Empowering large language models with symbolic solvers for faithful logical reasoning," in *Findings of the Association for Computational Linguistics: EMNLP 2023*, 2023, pp. 3806–3824.

Workflow[32]



[32] L. Pan, A. Albalak, X. Wang, *et al.*, "Logic-lm: Empowering large language models with symbolic solvers for faithful logical reasoning," in *Findings of the Association for Computational Linguistics: EMNLP 2023*, 2023, pp. 3806–3824.

Self-Refinement Mechanism[33]

- When solver fails $\rightarrow$ extract error, feed back to LLM
- LLM rephrases symbolic representation $\rightarrow$ re-run solver

[33] L. Pan, A. Albalak, X. Wang, *et al.*, "Logic-lm: Empowering large language models with symbolic solvers for faithful logical reasoning," in *Findings of the Association for Computational Linguistics: EMNLP 2023*, 2023, pp. 3806–3824.

Quantitative Gains[34]

- Logic-LM vs. standard prompting: +39.2pp
- Logic-LM vs. Chain-of-Thought (CoT): +18.4pp

[34] L. Pan, A. Albalak, X. Wang, *et al.*, "Logic-lm: Empowering large language models with symbolic solvers for faithful logical reasoning," in *Findings of the Association for Computational Linguistics: EMNLP 2023*, 2023, pp. 3806–3824.

Summary[35]

Contributions

- Extension of autoformalization to logical reasoning tasks
- Modular integration of LLM and solver
- Applicable across multiple logic domains

Limitations

- Reliant on accurate NL→SL translation
- Limited to problems solvable by available symbolic solvers

[35] L. Pan, A. Albalak, X. Wang, *et al.*, "Logic-lm: Empowering large language models with symbolic solvers for faithful logical reasoning," in *Findings of the Association for Computational Linguistics: EMNLP 2023*, 2023, pp. 3806–3824.

Other Examples

- Answer Set Programming[36]
- Temporal Logic[37,38,39]

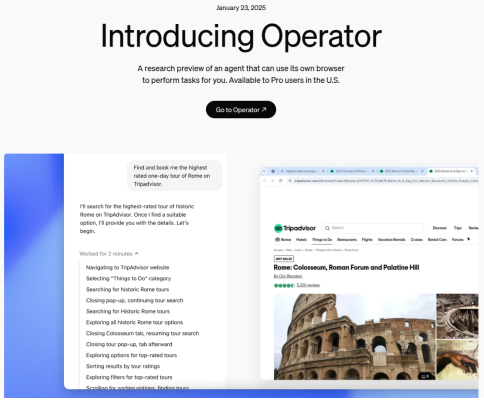
[36] Z. Yang, A. Ishay, and J. Lee, "Coupling large language models with logic programming for robust and general reasoning from text," *arXiv preprint arXiv:2307.07696*, 2023.

[37] M. Cosler, C. Hahn, D. Mendoza, *et al.*, "Nl2spec: Interactively translating unstructured natural language to temporal logics with large language models," in *International Conference on Computer Aided Verification*, Springer, 2023, pp. 383–396.

[38] Y. Chen, R. Gandhi, Y. Zhang, *et al.*, "Nl2tl: Transforming natural languages to temporal logics using large language models," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023, pp. 15 880–15 903.

[39] A. Mavrogiannis, C. Mavrogiannis, and Y. Aloimonos, "Cook2ltl: Translating cooking recipes to ltl formulae using large language models," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2024, pp. 17 679–17 686.

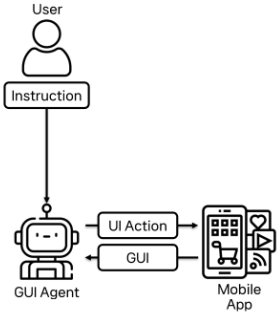
GUI Agents[40,41]



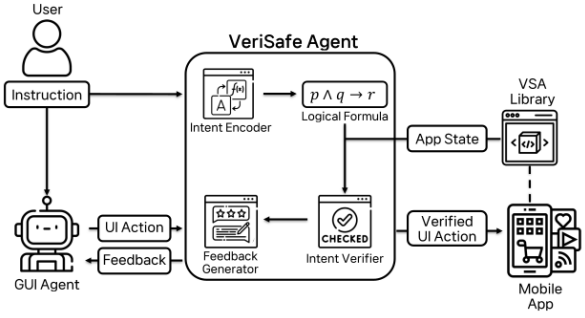
[40] https://www.anthropic.com/news/3-5-models-and-computer-use

[41] https://openai.com/index/introducing-operator/

Safeguarding Mobile GUI Agent[42]



(a) Traditional Mobile GUI Agent System



(b) Mobile GUI Agent System with VeriSafe Agent integration

[42] J. Lee, D. Lee, C. Choi, *et al.*, "Safeguarding mobile gui agent via logic-based action verification," *arXiv preprint arXiv:2503.18492*, 2025.

Challenges and Open Problems

Key Challenges

- **No guarantees of correctness:**
  - LLM-generated formal statements may be syntactically valid but semantically incorrect.
- **Evaluation is difficult:**
  - BLEU scores and syntactic matching don't capture semantic equivalence.
  - Manual review is costly and does not scale.

Open Problems

- How to ensure faithfulness of formalization?
- How to develop better, automated evaluation metrics?
- How to integrate formal verification during generation?

Autoformalization for Strategic Interaction Modelling

- Natural language strategic reasoning[43]
- Autoformalization into normal-form game representations[44]
- Autoformalization into extensive-form game representations[45]
- Streamlining simulation development from high-level descriptions[46]

[43] A. Mensfelt, K. Stathis, and V. Trencsenyi, "Logic-enhanced language model agents for trustworthy social simulations," *arXiv preprint arXiv:2408.16081*, 2024.

[44] A. Mensfelt, K. Stathis, and V. Trencsenyi, "Autoformalization of game descriptions using large language models," *First International Workshop on Next-Generation Language Models for Knowledge Representation and Reasoning*, 2024.

[45] S. Deng, Y. Wang, and R. Savani, "From natural language to extensive-form game representations," *arXiv preprint arXiv:2501.17282*, 2025.

[46] A. Mensfelt, K. Stathis, and V. Trencsenyi, "GAMA: Generative agents for multi-agent autoformalization," *arXiv preprint arXiv:2412.08805*, 2024.