Strategic AI: Bridging Game Theory and Multi-Agent Systems via Autoformalization

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

https://github.com/dicelab-rhul/Strategic-AI

AAMAS, 20/05/25

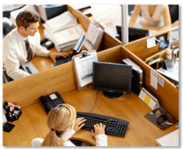
Outline

Introduction

Motivation

Why strategic interactions matter in AI

Many applications are naturally multi-agent:



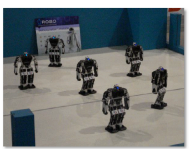
Human teams and companies

Markets and economies

Transportation networks

Distributed software systems

Communication networks



Robotic teams

How should agents act in the presence of other agents?

Strategic interaction

Why reasoning is essential

One way to answer this question is by expecting agents to think strategically.

- **Interdependent Decisions:** Agent choices affect and depend on others — strategic reasoning enables intelligent interaction.
- **Lack of information:** Strategic reasoning allows agents to model beliefs about others and act under information that is incomplete or imperfect.
- **Dynamic environment:** Strategic reasoning allows agents respond effectively to these ongoing changes.
  - **Collaboration vs Competition**

Key Message

To build intelligent and interactive agents, strategic reasoning is indispensable.

Games as Microcosms of Intelligence

Games provide a natural setting for studying and evaluating strategy.

- Success in games captures key elements of **strategic reasoning**:
  - **Problem-solving** in dynamic, rule-based environments
  - **Plan** sequences of actions to achieve long-term objectives
  - **Adaptation** to opponents' tactics and evolving situations
  - **Learning** from feedback to improve future decisions
- Quantifiable performance in games enables a **comparative assessment of strategies** and, as a result, **intelligence**

Why Game Playing Matters to AI Research

Game Playing as a Testbed for strategic AI

- Games have long been used to test AI:

Deep Blue (Chess)

AlphaGo (Go)

- These systems were successful but **not general** - they only played one game.
- **Can we build AI that plays *any* game?**

- **Goal:** Build agents capable of **General Game Playing (GGP)** - excel in multiple games from their formal rules alone.
- **Key Idea:** Use a shared, declarative formalism for representing a game - the **Game Description Language (GDL)**.
- **Approach:**
  - Provide agents with new game rules expressed in GDL, without additional code or training.
  - When encountering a new game:
    - The agent parses and understands its rules and objectives.
    - It uses this understanding to plan, adapt, and compete strategically.
  - Shift away from game-specific logic toward **domain-general intelligence**.

Game Theory and Multi-Agent Systems

Game Theory Foundations

A *game* models strategic interactions between decision-makers (players) where outcomes are interdependent on players' choices.[1]

[1] R. B. Myerson, "An introduction to game theory," Northwestern University, Center for Mathematical Studies in Economics and Management Science, Discussion Papers 623, 1984.

Games: Definition and Components

$$G = (N, \{A_i\}_{i \in N}, \{U_i\}_{i \in N})$$

- **Players** ($N = \{1, \ldots, n\}$): Strategic decision-makers involved in the game.
- **Actions** (A_i): The set of immediate choices available to player i.
- **Strategy Profile** ($o = (a_1, \ldots, a_n)$): A tuple representing the selected strategy of each player – the game's *outcome*.
- **Utility Functions** ($U_i : A_1 \times \cdots \times A_n \to \mathbb{R}$): An ordinal/cardinal representation of player i's preferences over outcomes.

Strategies

A strategy is a complete specification of play; a plan for a course of actions.

- A **pure strategy** for a player i is a single deterministic action $a_i \in A_i$.
- A **mixed strategy** is a probability distribution over pure strategies $\sigma_i \in \Delta(A_i), \quad \sum_{a_i \in A_i} \sigma_i(a_i) = 1$, where $\Delta(A_i)$ is the set of all probability distributions over A_i.

Representative Forms

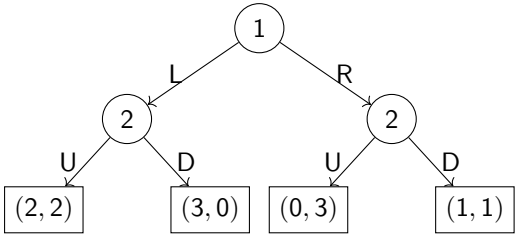
Normal Form

- Captures simultaneous actions
- Captures strategic structure
- $(\pi_{\text{row}}, \pi_{\text{col}})$ are the payoffs of an o

	L	R
U	(2,2)	(0,3)
D	(3,0)	(1,1)

Extensive Form

- Captures sequential moves
- Captures temporal/hierarchical structure



Payoff Structures

Zero-Sum Game

	L	R
U	(A, -A)	(-C, C)
D	(-B, B)	(D, -D)

Non-Zero-Sum Game

	L	R
U	(A, A)	(C, C)
D	(B, B)	(D, D)

Symmetric Payoffs

	L	R
U	(A, A)	(C, C)
D	(B, B)	(D, D)

Asymmetric Payoffs

	L	R
U	(A_i, A_j)	(C_i, C_j)
D	(B_i, B_j)	(D_i, D_j)

Payoff Structures

Matching Pennies

	Head	Tails
Head	(1, -1)	(-1, 1)
Tails	(-1, 1)	(1, -1)

Stag Hunt

	Stag	Hare
Stag	(5, 5)	(0, 3)
Hare	(3, 0)	(1, 1)

Prisoner's Dilemma

	C	D
C	(3, 3)	(0, 5)
D	(5, 0)	(1, 1)

Battle of the Sexes

	Ballet	Football
Ballet	(2, 1)	(0, 0)
Football	(0, 0)	(1, 2)

Solution Concepts

Dominant Strategies

A dominant strategy is one that always yields a higher payoff, no matter what the opponent does.

$a_i \in A_i$ is a strictly dominant action for player i if $\forall a_{-i} \in A_{-i}$,

$U_i(a_i, a_{-i}) > U_i(a'_i, a_{-i}) \quad \forall a'_i \neq a_i$

Prisoner's Dilemma

	C	D
C	(3, 3)	(0, **5**)
D	(**5**, 0)	(**1**, **1**)

Solution Concepts

Nash Equilibrium

A Nash equilibrium is a strategy profile where no player can benefit by unilaterally deviating.

$(a_1^*, \ldots, a_n^*)$ is a Nash equilibrium if $\forall i \in N$, $U_i(a_i^*, a_{-i}^*) \geqslant U_i(a_i, a_{-i}^*) \quad \forall a_i \in A_i$

Prisoner's Dilemma

	C	D
C	(3, 3)	(0, 5)
D	(5, 0)	(**1**, **1**)

Solution Concepts

Pareto Efficiency

An outcome is Pareto efficient if no player can be made better off without making another worse off.

$o \in A_1 \times \cdots \times A_n$ is Pareto efficient if $\nexists o'$ such that

$U_i(o') \geqslant U_i(o)\ \forall i$ and $\exists j,\ U_j(o') > U_j(o)$

Prisoner's Dilemma

	C	D
C	(**3**, **3**)	(0, 5)
D	(5, 0)	(1, 1)

Solution Concepts

Social Welfare

Social welfare is the sum of all players' utilities. An outcome is socially optimal if it maximizes this sum.

$o^* = \arg\max_{o \in A_1 \times \cdots \times A_n} \sum_{i \in N} U_i(o)$

Prisoner's Dilemma

	C	D
C	(**3**, **3**)	(0, 5)
D	(5, 0)	(1, 1)

Total payoff is maximized at (C, C):

$3 + 3 = 6$

Participatory Game

2 person game instruction:

"Your task is to guess how many balls are hidden in this black box. The box can be empty and can hold up to 100 balls. However, there is a twist! The winning guess is not the one closest to the actual number of balls hidden in the box. The closest guess to the two players' choices' mean multiplied by 0.66 will be the winner."

menti.com : 7130 6836

Strategic Reasoning in the p-Beauty Contest

The p-Beauty Contest[2,3]

- Each player selects a number in a known interval (e.g., [0, 100]).
- The winner is the player whose guess is closest to p times the average of all guesses.

Why is it interesting?

- Models *strategic anticipation* through **recursive reasoning**:
  - "What others think I will think..."
  - Formally captured by k-level theory
- No strictly dominant strategy
- Nash equilibrium: all players choose 0
- Special case[4] with $n = 2$

[2] J. M. Keynes, "The general theory of employment," *The quarterly journal of economics*, vol. 51, no. 2, pp. 209–223, 1937.

[3] R. Nagel, "Unraveling in guessing games: An experimental study," *The American Economic Rev.*, vol. 85, no. 5, pp. 1313–1326, 1995.

[4] B. Grosskopf and R. Nagel, "The two-person beauty contest," *Games and Economic Behavior*, vol. 62, no. 1, pp. 93–99, 2008.

Game Mechanisms

Cooperative vs Non-Cooperative

- Cooperative: binding agreements and coalition formation possible
- Non-cooperative: players act individually without enforceable contracts

Simultaneous vs Sequential

- Simultaneous: players act without knowledge of others' choices (normal form)
- Sequential: players move in turns, observing prior actions (extensive form)

Information Structures

Complete vs Incomplete Information

- Can players fully observe others' payoffs?

Perfect vs Imperfect Information

- Can players fully observe others' actions?

Symmetric vs Asymmetric Information

- Can players access the same quality/quantity of information?

Intelligent Agents and Strategic Reasoning

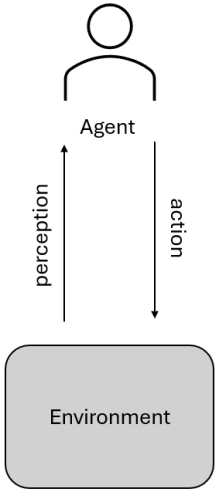
Real-world conflicts entail practical scenarios, where societies of actors are involved in cooperative or competitive interactions. For such cases, agent-based approaches are deemed a natural metaphor[5].

[5] M. Wooldridge, *An introduction to multiagent systems*. John wiley & sons, 2009.

Agents and the Environment

What is an agent?

An agent perceives the environment and produces actions that affect it.

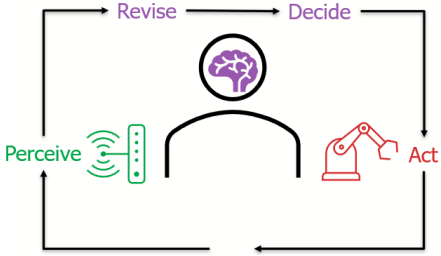


Agents and the Environment

What is an agent?

Agents are characterized by[6]:

- Autonomy (operating without direct intervention)
- Social ability (communicating with other agents)
- Reactivity (responding to environmental changes)
- Proactivity (exhibiting goal-directed behaviour)



OODA Agent[7]

[6] M. Wooldridge and N. R. Jennings, "Intelligent agents: Theory and practice," *The Knowledge Engineering Review*, vol. 10, no. 2, pp. 115–152, 1995

[7] V. Trencsenyi, A. Mensfelt, and K. Stathis, "Approximating human strategic reasoning with llm-enhanced recursive reasoners leveraging multi-agent hypergames," *arXiv preprint arXiv:2502.07443*, 2025

Agent Theories

How do agents work internally?

- **Reactive Architectures** Rely on simple stimulus-response mechanisms without internal models or planning.
- **Deliberative Architectures** Maintain internal representations, and support reasoning, memory, and planning, often inspired by cognitive processes.
- **Hybrid Architectures** Combine reactive and deliberative layers to balance reactive responsiveness with symbolic representations and goal-driven behaviour.

BDI Agents: Human-like Practical Reasoning

Belief–Desire–Intention (BDI) agents model human-like deliberative processes by decoupling reasoning into:

- **Beliefs (B)**: informational state (agent's internal model of the world)
- **Desires (D)**: motivational state (goals the agent might want to achieve)
- **Intentions (I)**: deliberative commitments to plans of action

BDI architecture functions:

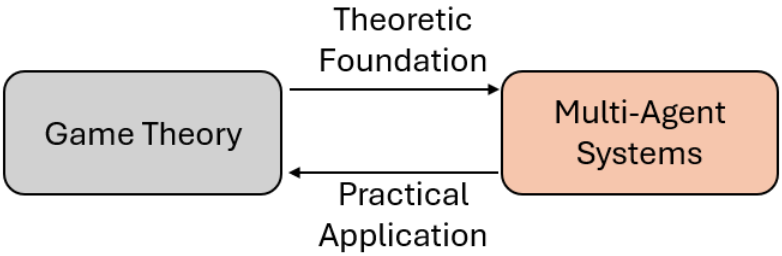
- $\mathcal{B}$: Belief revision from new percepts
- $\mathcal{O}$: Option generation based on B, I
- $\mathcal{F}$: Intention filtering from B, D, I
- $\mathcal{P}$: Planning over B, I to action sequence Π

BDI agents cycle through these functions to continuously update beliefs, deliberate, and act — paralleling human practical reasoning.

Multi-Agent Systems

Multi-Agent Systems (MAS) are societies of autonomous agents interacting in a shared environment.

- Each agent pursues individual or shared goals
- Agents coordinate, compete, and communicate
- Strategic interactions naturally emerge



Uncertainty in MAS

Realistic MAS must handle uncertainty beyond ideal game-theoretic assumptions.

- **Environmental uncertainty**: Partial observability, dynamic environments, non-stationarity
- **Action and Strategic uncertainty**: Non-deterministic effects of actions, unknown opponent strategies
- **Bounded rationality and belief misalignment**: Cognitive limits, mismatched models of the world or other agents

The Game Description Language

What is GDL?

The **Game Description Language**[6] **(GDL)** is a formal language used to define the rules of any game in a machine-readable format.

- It is based on **logic programming**, using a Datalog-style syntax.

Why GDL?

- Allows AI agents to **read and interpret new games** without manual reprogramming.
- Considered as the foundation for **General Game Playing**[7] **(GGP)**.

[6] N. Love, M. Genesereth, and T. Hinrichs, "General game playing: Game description language specification," Stanford University, Logic Group, Technical Report LG-2006-01, 2006.

[7] M. R. Genesereth, N. Love, and B. Pell, "General game playing: Overview of the AAAI competition," *AI Mag.*, vol. 26, no. 2, pp. 62–72, 2005.

Digression: Logic Programs

Syntax and Structure

A **logic program**[8] is a set of Horn clauses, each written as:

$$A_0 \leftarrow A_1 \wedge A_2 \wedge \cdots \wedge A_n \quad \text{(where } n \geqslant 0\text{)}$$

- Each A_i is an **atomic formula** $p(t_1, \ldots, t_m)$:
  - p is a predicate symbol
  - t_i are **terms** (constants, variables, or $f(t_1, \ldots, t_m)$)
  - f is a function symbol
- Variables in terms are **universally quantified**, and the scope is the clause in which the variable occurs
- Notation:
  - $\leftarrow$ means "if"
  - $\wedge$ means "and"

[8] R. A. Kowalski, "Logic programming," in *Computational Logic*, ser. Handbook of the History of Logic, J. H. Siekmann, Ed., vol. 9, Elsevier, 2014, pp. 523–569.

Digression: Logic Programs

Clause Types

- In a clause:

$$A_0 \leftarrow A_1 \wedge \cdots \wedge A_n$$

  - A_0 is called the **head** (or conclusion)
  - $A_1 \wedge \cdots \wedge A_n$ is the **body** (or conditions)
- If $n = 0$, the body is equivalent to `true`, and the clause becomes a **fact**:

$$A_0 \leftarrow \text{true} \quad \text{is abbreviated as } A_0$$

- If $n > 0$, the clause is a **rule**.
- If the head A_0 is `false`, the clause is a **goal clause**, abbreviated as:

$$\leftarrow A_1 \wedge \cdots \wedge A_n$$

That is, deny that $A_1 \wedge \cdots \wedge A_n$ has a solution, and refute it by finding one.

Digression: Normal Logic Programs

Horn clauses fail to capture **non-monotonic reasoning**, so we extend them to clauses of the form:

$$A_0 \leftarrow A_1 \wedge \cdots \wedge A_n \wedge \text{not } B_1 \wedge \cdots \wedge \text{not } B_m \quad (\text{where } n \geqslant 0 \text{ and } m \geqslant 0)$$

- Each A_i and B_j is an **atomic formula**.
- not is interpreted as **negation by failure (NBF)**.
- Atomic formulas and their negations are called **literals**:
  - A_i are **positive literals**
  - *not* B_j are **negative literals**
- Clause sets are called **normal logic programs** (or *logic programs*).

Syntax Variants of GDL and Examples

Notations for logic programming clauses:

$$A_0 \leftarrow A_1 \wedge \cdots \wedge A_n \wedge \text{not } B_1 \wedge \cdots \wedge \text{not } B_m$$

Prefix Notation (Lisp-like)

(<= A_0 A_1 ... A_n (not B_1) ... (not B_m))

E.g.

(<= (flies ?x) (bird ?x)(not (penguin ?x)))

Infix Notation (Prolog-like)

A_0 :- A_1, ..., A_n, not B_1, ... , not B_m.

E.g.:

flies(X) :- bird(X), not penguin(X).

For more, see here[9],[10].

[9] R. Jones, C. Maynard, and I. Stewart, *The Art of Lisp Programming*. Springer London, 1990.

[10] L. S. Sterling and E. Y. Shapiro, *The Art of Prolog: Advanced Programming Techniques*, Second. MIT Press, 1994.

GDL Specification Process for Games

Question: How do we specify programs in GDL?

- GDL defines a **shared vocabulary** to support game representation:
  - Establishes a fixed, **game-independent vocabulary** - shared meaning for key terms across all games.
  - Allows use of **game-specific vocabulary** - authors can define their own predicates and constants for individual games.
- Includes **predefined object constants** (0 to 100):
  - Often used to define **utility values** for game outcomes (e.g., 0 = worst, 100 = best)

Game Independent Vocabulary

Core Vocabulary [11]	Explanation
`role(R)`	R is a role in the game.
`init(F)`	Fact F is true at the beginning of the game.
`true(F)`	Fact F is true in the current state.
`does(R, A)`	Action A taken by role R.
`legal(R, A)`	Action A is legal for R in the current state.
`next(F)`	Fact F is true in the next state.
`goal(R, U)`	Role R gets payoff U in the current state.
`terminal`	Declares terminal states.

Represent a game in terms of this vocabulary, plus any additional predicates.

[11] We omit base(F) for possible state propositions F in a state, input(R, A) for actions A possible for role R, and distinct(X,Y) if $X \neq Y$.

PD as a concrete GDL Formalisation

We illustrate next how to instantiate the game independent vocabulary by formalizing a specific game:

- We will use the **Prisoner's Dilemma** (PD) as our example.
- We have already discussed PD previously.
- The goal is to see how represent PD as a GDL instance (not to run it).

PD Formalization: Roles and Initial State

Defined Roles:

```
% --- Player roles: row and col
% --- from the matrix

    role(row).
    role(col).
```

Initial State:

```
% --- What holds initially ---

    init(control(row)).
    init(control(col)).
```

PD Formalization: Legal Moves and State Transitions

Legal Moves:

```
% --- legal moves ---

    legal(R, coop) :- true(control(R)).
    legal(R, defect) :- true(control(R)).
```

State Transitions:

```
% --- state transitions ---

    next(did(R, M)) :- does(R, M), true(control(R)).
```

PD Formalization: Payoff Rules


```
% --- Both cooperate ---

goal(R1, 3) :-
    true(did(R1, coop)),
    true(did(R2, coop)),
    distinct(R1, R2).

% --- Both defect ---

goal(R1, 1) :-
    true(did(R1, defect)),
    true(did(R2, defect)),
    distinct(R1, R2).
```

```
% --- Sucker's Payoff ---

goal(R1, 0) :-
    true(did(R1, coop)),
    true(did(R2, defect)),
    distinct(R1, R2).

% --- Temptation Payoff ---

goal(R1, 5) :-
    true(did(R1, defect)),
    true(did(R2, coop)),
    distinct(R1, R2).
```

PD Formalization: Termination Condition


```
% --- Termination ---

terminal :-
    true(did(R1, M1)),
    true(did(R2, M2)),
    distinct(R1, R2).
```

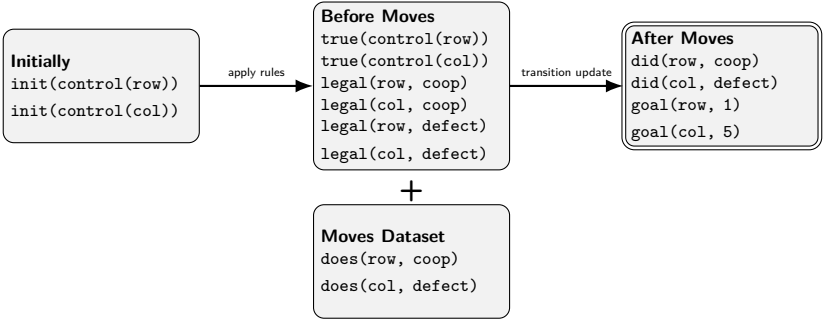
- This rule defines when the game ends: both players have made a move.
- We have now completed the formalization of the Prisoner's Dilemma.
- Emphasis has been on **declarative representation**, not execution.

Question:
What would a `Datalog`-style reasoning engine derive from this?

PD State Transitions (GDL Trace)

How does it work?

- Reason forwards from initial state until terminal state is reached[12].



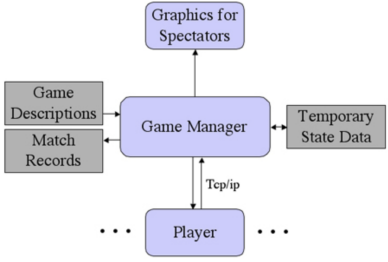
- GDL 1.0 uses a *Game Manager* for state and action coordination.

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

PD Formalization: Reflection

Key Observations:

- Designed for turn-taking games with perfect information.
- Prefix syntax (Lisp-style) for rules not very user-friendly.
- Some modelling questions:
  - Why use `role(row)` instead of:
    - `player(p1).`
    - `role(p1, row).`
  - input/2 vs possible/2 moves.



Extensions of GDL 1.0: GDL-II and rtGDL

Several formal extensions of GDL 1.0 exist, including:

- **GDL-II**[13] - *Games with Incomplete Information*
  - Adds `sees/2` to model **observations**.
  - Adds `random/1` for **stochastic outcomes**.
  - Enables modelling of imperfect-information and simultaneous-move games.
- **rtGDL**[14] - *Real-Time General Game Description Language*
  - A time parameter $T \in \mathbb{R}^+ \cup \{\infty\}$ is associated with a currently true fact F, encoding the *lifetime* of that fact - i.e., the duration for which F remains true.
  - Once T has elapsed, the auxiliary relation `expired`(F) is asserted to indicate that F has expired.
  - Enables asynchronous gameplay.

[13] 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.

[14] 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.

A Custom Solver for GDL-II and rtGDL

Motivation

- While implementations of **GDL-II** and **rtGDL** exist, they are often:
  - Not publicly available, or
  - Difficult to access, modify, or extend
- Our approach: develop a **logic programming solver in Prolog** that:
  - Simulates the `Datalog`-style reasoning of GDL 1.0
  - Provides a more manageable and extensible foundation
  - Enables rapid experimentation with new extensions/semantics
- This enables easier prototyping of GDL extensions in a familiar declarative environment.

Solver: Domain-Independent Part[15]

Game Execution 'Semantics'

Top-level of our solver:

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

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

States and Moves:

Abstract transition function `do(M, S)`.

Observations:

- Defines all legal **execution traces** from a starting state `S` to a final state `F`.
- Can be used as a **generator** to enumerate possible plays.
- Can be used as a **tester** to verify reachability of a final state.

[15] 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.

Solver: Domain-Independent Part

Reasoning about game transitions in the Situation Calculus[16]

Key Design Decisions:

- Must define S and M:
  - State labelled as:

    $\underline{s_0}$, $\underline{do(m_1, s_0)}$, $\underline{do(m_2, do(m_1, s_0))}$, ...
  - A move is a term:
    `move(P, A)`
    `P` is player ID, `A` the action.
- We use the **situation calculus** to reason about state transitions.

Situation calculus for our solver:

```
holds(F, S) :-
    initially(F, S).

holds(F, do(M, S)) :-
    effect(F, M, S).

holds(F, do(A, S)) :-
    holds(F, S),
    not abnormal(F, A, S).
```

[16] J. McCarthy and P. J. Hayes, "Some philosophical problems from the standpoint of artificial intelligence," in *Machine Intelligence 4*, B. Meltzer and D. Michie, Eds., Edinburgh University Press, 1969, pp. 463–502.

PD Formalization revisited: Roles and Initial State

Initial Situation:

```
initial(s0).

initially(player(p1), s0).
initially(player(p2), s0).
initially(role(p1, row), s0).
initially(role(p2, col), s0).
initially(control(p1), s0).
initially(control(p2), s0).
```

Explanation:

- Initial situation denoted by `s0`.
- There are two players with unique ids: `p1` and `p2`.
- Each player is assigned a role: `row` or `col`.
- Both players are initially in control - allowed to act.

PD Formalization revisited: Legal Moves and State Transitions

Legal, Possible, and Effect Rules:

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

possible(move(P, defect), S) :-
    holds(player(P), S).
possible(move(P, coop), S) :-
    holds(player(P), S).

effect(did(P, M), move(P, M), S).

abnormal(control(P), move(P, M), S).
```

Explanation:

- A move is `legal` if it is `possible` and player is currently in control.
- It is always possible for players to either cooperate or defect.
- The `effect/3` clause captures how the move alters the state.
- The `abnormal/3` clause indicates that executing a move ends the player's control.

PD Formalization revisited: Payoff Rules (v1)


```
% --- Both cooperate ---
holds( goal(P1, 3), S):-
    holds(did(P1, coop), S),
    holds(did(P2, coop), S),
    distinct(P1, P2).

% --- Both defect ---
holds( goal(P1, 1), S):-
    holds(did(P1, defect), S),
    holds(did(P2, defect), S),
    distinct(P1, P2).
```

```
% --- Sucker's Payoff ---
holds(goal(P1, 0), S) :-
    holds(did(P1, coop), S),
    holds(did(P2, defect), S),
    distinct(P1, P2).

% --- Temptation Payoff ---
holds(goal(P1, 5), S) :-
    holds(did(P1, defect), S),
    holds(did(P2, coop), S),
    distinct(P1, P2).
```

PD Formalization Revisited: Payoff Rules (v2)


```
% Row player payoff
holds(goal(P1, U1), S) :-
    holds(role(P1, row), S),
    holds(did(P1, M1), S),
    holds(role(P2, col), S),
    holds(did(P2, M2), S),
    payoff(M1, M2, U1, U2).

% Column player payoff
holds(goal(P2, U2), S) :-
    holds(role(P1, row), S),
    holds(did(P1, M1), S),
    holds(role(P2, col), S),
    holds(did(P2, M2), S),
    payoff(M1, M2, U1, U2).
```

```
% --- Payoff matrix ---
payoff(defect, defect, 1, 1).
payoff(coop, defect, 0, 5).
payoff(defect, coop, 5, 0).
payoff(coop, coop, 3, 3).
```

Notes:

- Payoff matrix externally defined and decoupled from the rules.
- Can extend with game G, e.g.,

  `payoff(G, M1, M2, U1, U2).`

PD Formalization Revisited: Termination Condition


```
% --- Termination Condition ---
final(S) :-
    holds(did(R1, M1), S),
    holds(did(R2, M2), S),
    distinct(R1, R2).

finally(F, S) :- final(S), holds(F, S).
```

GDL-like formalization of the PD in Prolog completed.

- `final/1` plays a similar role to `terminal/0` in GDL, but is defined via `holds/2` to reflect a state.
- `finally/2` extracts all facts that hold in terminal states.

Remarks on our Formalization

- Query below returns **8** traces instead of **4**:

  ?- game(s0, F).
  F = do(move(p2, defect), do(move(p1, defect), s0)) ;
  F = do(move(p2, coop), do(move(p1, defect), s0)) ;
  F = do(move(p1, defect), do(move(p2, defect), s0)) ;
  ...

- In 4 `p1` plays and `p2` second, and in 4 vice-versa.
- Why? Because situation term `do(M, S)` allows one move at a time.
- This reflects **interleaved semantics**, not true simultaneity.
- To model **simultaneous moves**, we shift to the **Event Calculus**[17].

[17] R. Kowalski and M. Sergot, "A logic-based calculus of events," *New Generation Computing*, vol. 4, pp. 67–95, 1986.

Event Calculus Version[18] 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) :-
    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.

[18] 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.

Example of Strategic Reasoning

Using Situation Calculus based Solver

Best move (based on opponent's move in previous game[19])

```
select(P, O, S, M) :-
    not holds(last_move(O, _), S),
    holds(default_move(P, M), S).
select(P, O, S, M) :-
    holds(last_move(O, LMo), S),
    findall(Ui-Mi,
        ( game(S, F),
          finally(outcome(P, Mi, Ui, O, LMo, Uo), F),
          Ui >= Uo),
        Options),
    best_move(Options, M).
```

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

Final Remarks on Our Approach

- Focused on **simple games**, but more complex ones may use:
  - efficient versions of EC[20],[21] or **depth-limited search** with max depth.
  - **ASP**[22] or **Model Checking**[23] if online use is feasible.
- Closer to **multi-agent adversarial planning**[24] than classical single-agent plans - worth further study.
- Did not address uncertainty - can **probabilistic logic programming**[25] help?
- Can agents use **LLMs to autoformalize games**?

[20] 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.

[21] 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.

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

[23] 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.

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

[25] 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, Simulation Framework, and Case Studies

Autoformalization

- Early ideas: 1960s[26].

  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).[27]

- Renewed interest with deep learning[28,29].
- Recent breakthroughs with large language models (LLMs)[30].

[26] J. McCarthy, "A basis for a mathematical theory of computation," in *Studies in Logic and the Foundations of Mathematics*, vol. 26, Elsevier, 1959, pp. 33–70.

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

[28] C. Szegedy, "A promising path towards autoformalization and general artificial intelligence," in *Intelligent Computer Mathematics: 13th International Conference, CICM 2020, Bertinoro, Italy, July 26–31, 2020, Proceedings 13*, Springer, 2020, pp. 3–20.

[29] Q. Wang, C. Kaliszyk, and J. Urban, "First experiments with neural translation of informal to formal mathematics," in *Intelligent Computer Mathematics: 11th International Conference, CICM 2018, Hagenberg, Austria, August 13-17, 2018, Proceedings 11*, Springer, 2018, pp. 255–270.

[30] 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.

Autoformalization

- **Mathematics**: Formalizing mathematical proofs and textbook content into interactive theorem prover languages (e.g., Lean)[31,32,33]
- **Logic**:
  - Answer Set Programming[34]
  - First-Order Logic[35]
  - Temporal Logic[36]

[31] 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.

[32] J. He-Yueya, G. Poesia, R. E. Wang, *et al.*, "Solving math word problems by combining language models with symbolic solvers," *arXiv preprint arXiv:2304.09102*, 2023.

[33] A. Q. Jiang, S. Welleck, J. P. Zhou, *et al.*, "Draft, sketch, and prove: Guiding formal theorem provers with informal proofs," *arXiv preprint arXiv:2210.12283*, 2022.

[34] 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.

[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.

[36] 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, 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.

Autoformalization for Strategic Interaction Modelling

- Natural language strategic reasoning[37]
- Autoformalization into normal-form game representations[38]
- Autoformalization into extensive-form game representations[39]
- Streamlining simulation development from high-level descriptions[40]

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

[38] 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.

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

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

Generative Agents for Multi-Agent Autoformalization[41]

Natural language interaction description

Two rival political candidates must decide whether to engage in mudslinging during their campaign. If both engage in mudslinging, they degrade the political discourse and might turn off voters, each receiving a payoff of 3. If one mudslings and the other runs a clean campaign, the mudslinger might tarnish the opponent's reputation and win more votes, receiving a payoff of 8, while the clean campaigner receives a payoff of 2. If neither engages in mudslinging, they uphold higher moral standards but miss the opportunity to undermine their opponent, each candidate getting a payoff of 5.

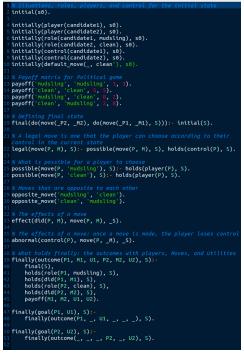
Interaction description

Natural language strategy description

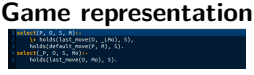
Start with a default move. Then, mirror the opponent's move in the previous round.

Strategy description

→



Game representation



Strategy representation

→

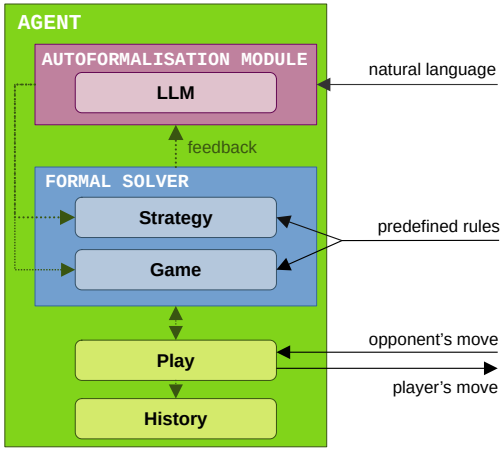
MAS

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

Generative Agents for Multi-Agent Autoformalization

- Multi-agent framework for generating formal representations of games and strategies.
- Agents equipped with autoformalization module.
- Three-level code validation:
  - Syntactic: using formal solver.
  - Runtime: validating through playing.
  - Semantic: comparing target vs. achieved payoffs.

Agent Model

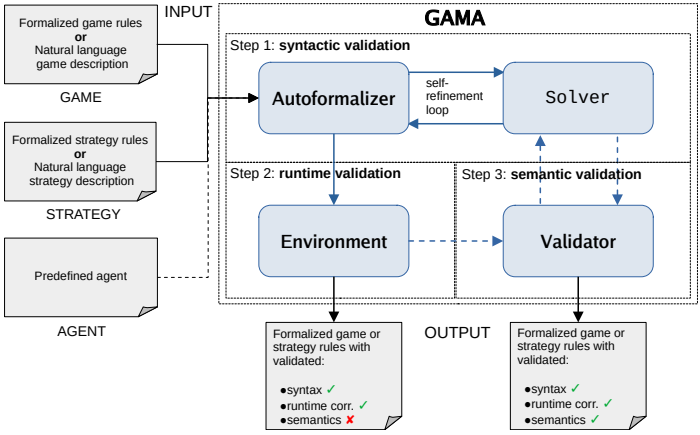


Autoformalisation Module[42]

Input:
Γ: game-independent predicates, NL_{PD}: natural language description of PD, ξ_{PD}: game-specific predicates of PD, NL_{NG}: natural language description of a new game.
Output:
ξ_{NG}: game-specific predicates for the new game.
Parameter:
`max_attempts`: maximum correction attempts.
`attempts` $\leftarrow 0$
`trace` $\leftarrow \varnothing$
while `attempts` $<$ `max_attempts` **do**
  $\xi_{NG} \leftarrow$ `LLM.translate`$(\Gamma, NL_{PD}, \xi_{PD}, NL_{NG})$
  `is_valid` $\leftarrow$ `solver.check_predicates`(ξ_{NG})
  if `is_valid` **then**
    return ξ_{NG}
  else
    `trace` $\leftarrow$ `solver.get_trace()`
    $\xi_{NG} \leftarrow$ `LLM.self_correct`$(\xi_{NG}, \texttt{trace})$
  end if
  `attempts` $\leftarrow$ `attempts` $+ 1$
end while
return `Unable to generate valid predicates within maximum attempts.`

[42] 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.

Overview



Input: Data Set of Game Descriptions

Prisoner's Dilemma: standard description

Two suspects are arrested and interrogated separately. Each can either confess or remain silent. If both suspects remain silent, they each receive a minor sentence of 1 year in prison. If one confesses and the other remains silent, the confessor goes free while the silent one receives a heavy sentence of 10 years. If both confess, they each receive a moderate sentence of 5 years. This situation, known as the Prisoner's Dilemma, demonstrates how individual rationality can lead to a worse collective outcome.

Input: Data Set of Game Descriptions

Prisoner's Dilemma: non-standard description

Two rival political candidates must decide whether to engage in mudslinging during their campaign. If both engage in mudslinging, they degrade the political discourse and might turn off voters, each receiving a payoff of 3. If one mudslings and the other runs a clean campaign, the mudslinger might tarnish the opponent's reputation and win more votes, receiving a payoff of 8, while the clean campaigner receives a payoff of 2. If neither engages in mudslinging, they uphold higher moral standards but miss the opportunity to undermine their opponent, each candidates getting a payoff of 5.

Input: Data Set of Game Descriptions

Prisoner's Dilemma: non-standard description without numerical payoffs

Two employees are working on a joint project and must decide whether to share all their innovative ideas or keep some to themselves for credit. If both share openly, the project flourishes and they achieve great results, earning joint recognition. If one shares while the other withholds, the sharer contributes more but feels exploited, while the withholder benefits more and gains more recognition. If neither shares openly, the project suffers, and they both receive mediocre evaluations.

Input: Data Set of Game Descriptions

Payoffs/Description	Standard	Non-standard
Numerical	5	50
Non-numerical	5	50

Table: Number of natural language game descriptions for each variant. **Standard** refers to example employing a typical metaphor for a given game. Conversely, **non-standard** is a newly invented example of a situation that can be modelled by a given game. **Numerical** refers to descriptions containing numerical payoffs, and **non-numerical** to descriptions without numerical payoffs.

Strategies

Strategy	Description
anti-default-move	Always select the move that is the opposite of the default move.
anti-tit-for-tat	Start with a default move. Then, select the move that is the opposite of the opponent's move in the previous round.
best-response	Start with a default move. Then, select a move that would give you the highest payoff in response to the opponent's move in the previous round.
default-move	Always select the default move.
random	Select one of the possible moves with uniform probability.
tit-for-tat	Start with a default move. Then, mirror the opponent's move in the previous round.

Table: Strategies utilised in the evaluation.

Experimental Parameters

Parameter	Value
LLM	*Claude 3.5S* *GPT-4o*
temperature	1
maximum output tokens	2048
maximum attempts number	5

Table: Common experimental parameters.

Experimental Parameters

Parameter	Exp. 1	Exp. 2	Exp. 3	Exp. 4
`games`	55	55	5	5
`agents in game`	5	5	1	1
`strategies`	1	1	6	6
`rounds`	4	4	10	10
`match-maker`	S	S	RR	S

Table: Experiment-specific parameters. *S* stands for a mode in which an agent plays against its clone with an opposite strategy, while *RR* stands for round robin.

Results: Attempts Distribution

	1	2	3	4	5
	Numeric payoffs				
Claude 3.5 Sonnet	100.00%	0.00%	0.00%	0.00%	0.00%
GPT-4o	96.77%	2.87%	0.00%	0.00%	0.36%
	Non-numeric payoffs				
Claude 3.5 Sonnet	100.00%	0.00%	0.00%	0.00%	0.00%
GPT-4o	97.64%	2.02%	0.34%	0.00%	0.00%

Table: Percentage distribution of attempts at creating syntactically correct code.

Results: Correctness

	Numeric Payoffs				
	BS	**HD**	**MP**	**PD**	**SH**
Syntactic	100.0%	100.0%	100.0%	100.0%	100.0%
Runtime	89.1%	96.4%	68.5%	96.4%	87.3%
Semantic	89.1%	96.4%	68.5%	89.1%	87.3%
	Non-Numeric Payoffs				
	BS	**HD**	**MP**	**PD**	**SH**
Syntactic	100.0%	100.0%	100.0%	100.0%	100.0%
Runtime	82.1%	85.7%	74.0%	70.7%	83.1%
Semantic	82.1%	77.8%	34.2%	64.0%	83.1%

Table: Correctness by game for Claude 3.5S.

Results: Correctness

	Numeric Payoffs				
	BS	**HD**	**MP**	**PD**	**SH**
Syntactic	98.3%	100.0%	100.0%	100.0%	100.0%
Runtime	83.1%	89.1%	96.4%	89.1%	90.9%
Semantic	79.7%	89.1%	90.9%	85.5%	83.6%
	Non-Numeric Payoffs				
	BS	**HD**	**MP**	**PD**	**SH**
Syntactic	100.0%	100.0%	100.0%	100.0%	100.0%
Runtime	93.2%	88.1%	87.3%	96.5%	91.5%
Semantic	93.2%	67.8%	30.2%	73.7%	76.3%

Table: Correctness by game for GPT-4o.

Summary and Future Work

- Conclusions
  - High syntactic correctness.
  - Room for improvement in runtime and semantic correctness.
  - Successful generalization for structurally similar games.
- Future Work
  - A more general game representation.
  - A feedback loop for runtime and semantic evaluation stages.
  - A learning mechanism.
- **How can we model more human-like, sophisticated reasoners?**

Belief Hierarchies and Recursive Reasoning

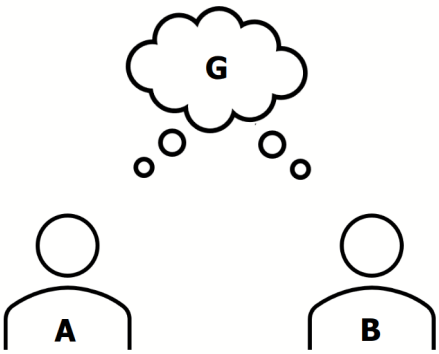
Game-theoretic Assumptions

Players are assumed to share a concept of rationality:

- players act by their free will
- players maximize utility
- players expect the same of others

Standard models also assume players':

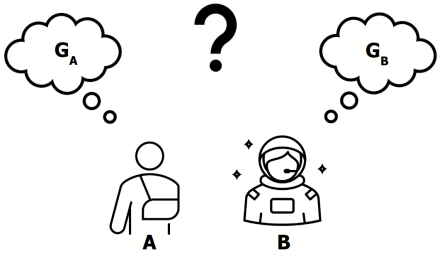
- common understanding of the conflict
- common concept of utility
- common cognitive and physical ability



Game-theoretic Assumptions[43,44,45,46,47]

Game theory makes simplifying assumptions to make models tractable.

- Individual factors affect utility
- Social context influences behaviour
- Players perceive and interpret differently
- Game-theoretic assumptions restrict agents' internal models



[43] T. Burns, "A structural theory of social exchange," *Acta Sociologica*, vol. 16, no. 3, pp. 188–208, 1973.

[44] D. D. Johnson, P. Stopka, and J. Bell, "Individual variation evades the prisoner's dilemma," *BMC Evolutionary Biology*, vol. 2, p. 15, 2002.

[45] T. Burns and L. D. Meeker, "Structural properties and resolutions of the prisoners' dilemma game," in *Game Theory as a Theory of a Conflict Resolution*, A. Rapoport, Ed. Springer Netherlands, 1974, pp. 35–62.

[46] P. G. Bennett, "Hypergames: Developing a model of conflict," *Futures*, vol. 12, no. 6, pp. 489–507, 1980.

[47] Y. Sasaki, "Multi-agent decision system," in *Handbook of Systems Sciences*, G. S. Metcalf, K. Kijima, and H. Deguchi, Eds. Singapore: Springer Singapore, 2021, pp. 337–352.

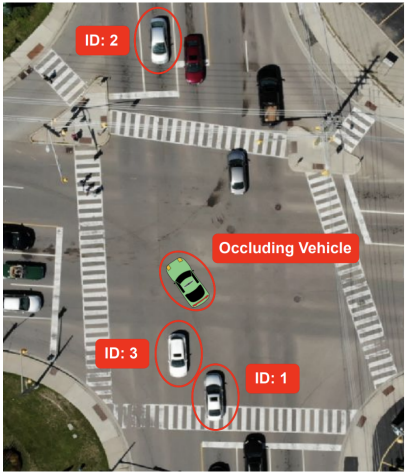
Humans Reason Recursively

The theory of mind (ToM) is an agent's cognitive capacity to attribute mental states – beliefs, desires, intentions, and knowledge – to oneself and others while understanding that others' mental states may differ from one's own[48].

[48] D. Premack and G. Woodruff, "Does the chimpanzee have a theory of mind?" *Behavioral and Brain Sciences*, vol. 1, no. 4, pp. 515–526, 1978.

Recursive Reasoning in Multi-agent Systems[49]

- Environments are non-stationary and not fully observable
- Agents are heterogeneous, with misaligned
  - Abilities
  - Objectives
  - Preferences



[49] M. Kahn, A. Sarkar, and K. Czarnecki, *I know you can't see me: Dynamic occlusion-aware safety validation of strategic planners for autonomous vehicles using hypergames*, 2021. arXiv: 2109.09807 [cs.RO].

Theory of Mind as a Benchmark

The theory of mind is a framework of nested beliefs: *Alice believes, that Bob believes that, Cecil believes...*

- Levels of recursion measure agent cognition[50,51]
- ToM addresses goal/perception misalignment in human-agent interaction[52,53,54]
- ToM for evaluating LLM reasoning[55,56]

[50] T. Bosse, Z. A. Memon, and J. Treur, "A recursive bdi agent model for theory of mind and its applications," *Applied Artificial Intelligence*, vol. 25, no. 1, pp. 1–44, 2011.

[51] M. Rocha, H. H. da Silva, A. S. Morales, *et al.*, "Applying theory of mind to multi-agent systems: A systematic review," in *Intelligent Systems*, M. C. Naldi and R. A. C. Bianchi, Eds., Cham: Springer Nature Switzerland, 2023, pp. 367–381.

[52] E. Erdogan, F. Dignum, R. Verbrugge, *et al.*, "Toma: Computational theory of mind with abstractions for hybrid intelligence," *Journal of Artificial Intelligence Research*, vol. 82, pp. 285–311, 2025.

[53] M. Mechergui and S. Sreedharan, "Goal alignment: Re-analyzing value alignment problems using human-aware ai," in *Adaptive Agents and Multi-Agent Systems*, 2024.

[54] A. Kierans, A. Ghosh, H. Hazan, *et al.*, *Quantifying misalignment between agents: Towards a sociotechnical understanding of alignment*, 2024. arXiv: 2406.04231 [cs.MA].

[55] J. W. Strachan, D. Albergo, G. Borghini, *et al.*, "Testing theory of mind in large language models and humans," *Nature Human Behaviour*, pp. 1–11, 2024.

[56] T. Ullman, "Large language models fail on trivial alterations to theory-of-mind tasks," *arXiv preprint arXiv:2302.08399*, 2023.

Game-theoretic Theory of Mind

How can we relax standard game-theoretic restrictions?

- Cognitive Hierarchy Model[57]
- Player Types and Nature[58]
- Epistemic Game Theory[59]

[57] C. F. Camerer, T.-H. Ho, and J.-K. Chong, "A Cognitive Hierarchy Model of Games*," *The Quarterly Journal of Economics*, vol. 119, no. 3, pp. 861–898, Aug. 2004.

[58] J. Harsanyi, "Games with incomplete information played by 'bayesian' players, parts i-iii," *Management Science*, vol. 14, pp. 159–182, 320–334, 486–502, 1967-1968.

[59] E. Dekel and M. Siniscalchi, "Epistemic game theory," in *Handbook of game theory with economic applications*, vol. 4, Elsevier, 2015, pp. 619–702.

Explicit Models of Subjective Games

"But in real life – and even in some laboratory games – it is clear that decision-makers' perceptions of the situation may differ radically."[60]

[60] P. Bennett, "Toward a theory of hypergames," *Omega*, vol. 5, no. 6, pp. 749–751, 1977.

Simple Hypergames

Hypergames are composites of each player's perceptual (subjective) games.

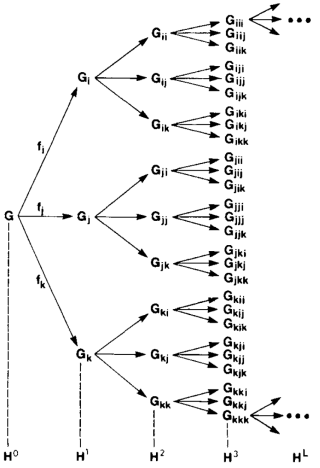
$$H = (N, \{G_i\}_{i \in N})$$

- $N = \{1, 2, \ldots, n\}$ is the set of agents
- For each agent $i \in N$, G_i represents agent i's perceived game, where $G_i = (N_i, A_i, U_i)$
  - $N_i \subseteq N$ is the set of agents as perceived by i
  - $A_i = \times_{j \in N_i} A_{ij}$ is the joint action space as perceived by i, where:
    - A_{ij} is i's perception of j's available actions
  - $U_i = \{U_{ij}\}_{j \in N_i}$ is the set of preferences as perceived by i, where:
    - $U_{ij} \subseteq A_i \times A_i$ is i's perception of j's preference

Hierarchical Hypergames

Wang et al. introduce an extended, concretized formalization[57].

- H^0: same as G, no perceptual differences
- H^1: misperceptions, players unaware
- H^2: at least 1 player is aware of H^1
- H^3: at least 1 player is aware of H^2

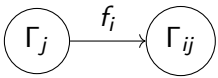


[57] M. Wang, K. W. Hipel, and N. M. Fraser, "Modeling misperceptions in games," *Behavioral Science*, vol. 33, no. 3, pp. 207–223, 1988

Recursive Reasoning via Game Hierarchies

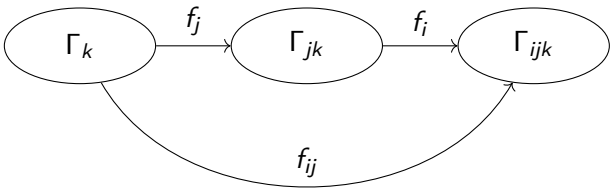
A *perspective* is a player's holistic interpretation of the base conflict, composed of *images* of the game's components, generated by the perceptual mapping $f_i : \Gamma_j \to \Gamma_{ij}$, where:

- Γ_j represents any component from player j's game
- Γ_{ij} denotes player i's interpretation of player j's component;
- For any element $\gamma \in \Gamma_j$, the image $\varphi = f_i(\gamma)$ represents player i's perception of that element.



Recursive Reasoning via Game Hierarchies

When a player i considers another j's perception of k's game, it is said to have a *higher-order expectation*, captured by a product mapping $f_i \circ f_j : \Gamma_k \to \Gamma_{ijk}$:



Then, the image φ is the product of individual perceptual functions,
$\varphi = f_i(f_j(\gamma)) = f_i \circ f_j(\gamma) = f_{ij}(\gamma)$

Recursive Reasoning via Game Hierarchies

- Higher-order expectations Γ_{ijk} are captured by equal-order subjective games G_{ijk}
- Given $\beta_i(\beta_j(\beta_k))$[61] corresponds to i's beliefs about player j's beliefs about player k's reasoning, we assume $\beta_i(\beta_j(\beta_k)) \cong G_{ijk}$.

[61] E. Dekel and M. Siniscalchi, "Epistemic game theory," in *Handbook of game theory with economic applications*, vol. 4, Elsevier, 2015, pp. 619–702.

Perceptual Beauty Contest Games[62]

$G = (N, A, U, \Psi)$

- **Players** $N = \{1, \ldots, n\}$
- **Actions** $A = A_i \times A_j, \quad A_i, A_j \subseteq \mathbb{Z}$
- **Utility Functions** $(U_i = U_j)$

$$U_i(a_i, a_{-i}) = \begin{cases} 1 & \text{if } |a_i - p\mu| < |a_{-i} - p\mu| \\ 0.5 & \text{if } |a_i - p\mu| = |a_{-i} - p\mu| \\ 0 & \text{otherwise} \end{cases}$$

- **Perspective** $\Psi = (\psi_1, \psi_2, \ldots, \psi_\kappa)$: an ordered sequence of κ perception steps, ψ_1 denotes the interpreter (creator) of Ψ

[62] V. Trencsenyi, A. Mensfelt, and K. Stathis, "Approximating human strategic reasoning with llm-enhanced recursive reasoners leveraging multi-agent hypergames," *arXiv preprint arXiv:2502.07443*, 2025.

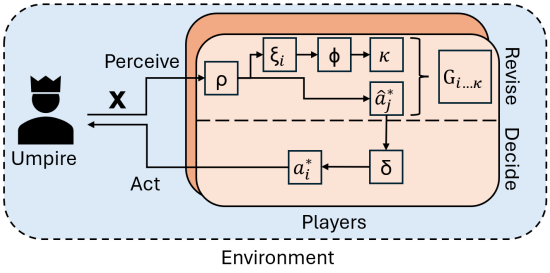
Multi-agent Centralized Hypergames

The umpire υ is a pseudo-player, who

- defines the **true game** $G^* = (N, A, U, \Psi = (\upsilon))$
- facilitates games
- validates player actions
- translates between subjective games

Multi-agent Centralized Hypergames

- υ is tasked with conducting an experiment
- υ derives G^*, then
  1. requests players to play
  2. validates responses
  3. provides feedback



Artificial Recursive Reasoners Towards Human-like Strategic Reasoning[63]

- Analysis of agentic sophistication:
  - Strongly conceptualized, structured architectures (BDI) vs. looser definitions (LLM-as-agent)
  - Human-cognition-inspired context models
- Analysis of population-wise and role-specific performance
  - Average k-level sophistication
  - Matching human guess distribution
  - Frequency of zero guesses
- Out-of-sample validation
  - Obfuscated game description
  - Shifted guess interval

[63] V. Trencsenyi, A. Mensfelt, and K. Stathis, *The influence of human-inspired agentic sophistication in llm-driven strategic reasoners*, 2025. arXiv: 2505.09396 [cs.AI].

Artificial Recursive Reasoners Towards Human-like Strategic Reasoning

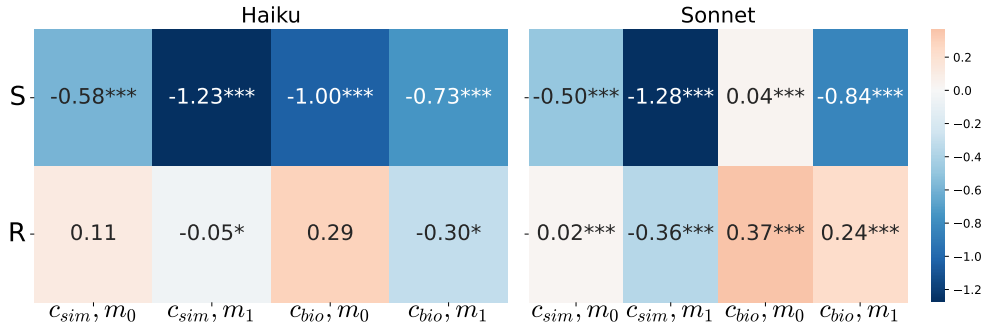


Figure: Delta between agents' and humans' cohort-wise mean k-level difference with significance levels: * $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$.

Summary

ROYAL HOLLOWAY UNIVERSITY OF LONDON

Summary

- Multi-agent systems exhibit strategic complexity beyond what classical game theory typically models.
- Strategic reasoning becomes more challenging in this general setting due to diverse agent goals, partial observability, and dynamic interactions.
- Game representations should be **formal**, **expressive**, and **computationally usable**.
- **Autoformalization** offers a promising route to scalable, general-purpose strategic reasoning in game-like settings.