StrategicWM: An Agentic Library to Automate the Construction of Strategic World Models

Ian Gemp
Google DeepMind
London, United Kingdom
imgemp@google.com

ABSTRACT

Game abstractions have been deployed throughout history to hone our strategic decision-making. However, building suitable abstractions that capture the salient details of strategic interactions remains a manual, costly process, confining game theory to a narrow range of applications. Recent advances in large language models (LLMs) offer a solution, as these models excel at translating messy real-world semantics into formal logical representations. We introduce *Strategic World Models* (SWMs), a methodology and accompanying open-source library, `StrategicWM`, that automatically translates natural language descriptions of strategic interactions into formal game abstractions directly solvable by computational game engines. We evaluate the fidelity of SWMs against ground-truth solutions in Kuhn Poker and demonstrate their versatility across diverse, non-traditional strategic domains. By automating this pipeline, we extend the reach of computational game theory to a much wider range of scenarios, including business negotiation, organizational politics, and complex interpersonal dynamics.

KEYWORDS

Game Theory, LLMs, Strategic World Models

ACM Reference Format:
Ian Gemp. 2026. `StrategicWM`: An Agentic Library to Automate the Construction of Strategic World Models. In *Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), Paphos, Cyprus, May 25 – 29, 2026*, IFAAMAS, 12 pages.

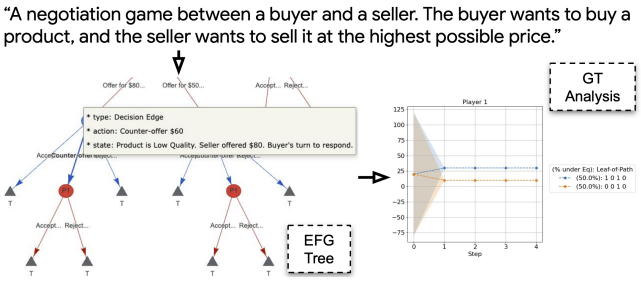

Figure 1: `StrategicWM`: An agentic library for automating the construction of game-theoretic models from ambiguous natural language descriptions and conducting equilibrium analysis with visualizations to surface strategic insights.

1 INTRODUCTION

Game abstractions have been deployed throughout history to hone our strategic decision-making. Famously, the pieces in Chess originated as the four divisions of the Indian army, eventually evolving into a symbolic warfare simulator [21]. Recent advances in computational game theory and AI [26] have paved the way for strategic planning that far exceeds human intuition, notably in domains like aerial combat simulations [7]. However, game theory is not restricted to warfare or zero-sum settings; it applies to any decision-making scenario given a suitable abstraction. What prevents game theory from advising us in everyday interactions? Why can we not apply game theory's insights to problems like business negotiations, organizational politics, or complex interpersonal dynamics?

As with Chess, building suitable game abstractions that capture the salient details of strategic interactions is historically a manual process relying on human creativity and domain expertise. While bespoke solutions exist for constrained domains like automated negotiation [3, 24] and ad auctions [9, 25], generic tools to automate the translation of unstructured scenarios into formal abstractions are lacking. However, large language models (LLMs) offer a potential solution. These models excel at processing messy real-world semantics and translating them into formal logic, filling in gaps using common sense derived from their vast training corpora.

In this work, we propose an approach that uses LLMs to automate the construction of game abstractions, called *strategic world models* (SWMs), that are formatted to be directly solvable by computational game engines. The result is an interface that allows one to describe any strategic decision-making problem and receive strategic advice supported by advanced AI.

In the following sections, we will survey recent related work (Section 5), technical background covering game theory abstractions (Section 2), methods that build a scaffolding around LLMs to reliably generate faithful abstractions of arbitrary strategic scenarios conditioned on natural language descriptions (Section 3), experiments that demonstrate successes and failure modes of the proposed approach (Section 4), and finally concluding remarks including limitations and promising areas for future work (Section 6).

2 TECHNICAL BACKGROUND

Extensive-form, imperfect-information games (EFGs) capture sequential interactions between players and their underlying valuations that influence the actions they take in a game. An EFG is a tuple $\langle \mathcal{N}, c, \mathcal{A}, \mathcal{H}, \mathcal{Z}, u, \tau, \mathcal{S} \rangle$. $\mathcal{N} = \{1, 2, \cdots, n\}$ is a set of n **players**. $\mathcal{H}$ is a finite set of **histories** or states. $\tau : \mathcal{H} \to \mathcal{N} \cup \{c\}$ is a player identity function. $\mathcal{S}$ is a set of **infostates**. Each infostate

Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), C. Amato, L. Dennis, V. Mascardi, J. Thangarajah (eds.), May 25 – 29, 2026, Paphos, Cyprus.

s represents a partition of $\mathcal{H}$ such that each $h \in s$ cannot be distinguished by players other than $\tau(s) = \tau(h)$ for all $h \in s$. $\mathcal{A}$ is a finite set of *infostate-dependent* **actions** that can be taken. $\mathcal{Z} \subseteq \mathcal{H}$ is a set of **terminal histories**. $u : \mathcal{Z} \to \Delta_u^n \subseteq \mathfrak{R}^n$ with $\Delta_u = [u_{\min}, u_{\max}]$ is a utility (or payoff) function assigning each player a payoff at the end of the game. EFGs are typically used to model games like Poker [2] or Stratego [22]. For more information, we refer the reader to textbooks [11, 19] and prior work that inspired ours [12].

3 METHODOLOGY

We propose two different approaches to automating the construction of SWMs from natural language descriptions. Both are designed to adhere to the extensive-form game abstraction defined in Section 2. One generates the abstraction directly using one LLM call, relying on the LLM to follow a specific EFG object format specified in its instruction prompt. The other provides a pipeline that generates each component of the EFG abstraction step-by-step, thereby, removing this responsibility from the LLM and narrowing the scope of each individual task, but then requiring many more LLM calls. We provide a high level description of the key concepts involved in the approaches here and direct the reader to the Python package documentation on GitHub for more details.

3.1 Direct Model Construction

We describe the one-shot generation approach first. Our approach is to construct a prompt with clear instructions to generate JSON according to the schema partially presented below:

```
class Node(BaseModel):
    node_id: int
    state_string: str
    current_player: int

class ChanceNode(Node):
    node_type: str = "chance"
    current_player: int = -1
    chance_probabilities: list[float]
    chance_outcomes_string: list[str]
    children: List[Union["ChanceNode", "DecisionNode",
        "TerminalNode"]]...
```

Imperfect information in EFGs is particularly nuanced, so we provide additional detail on how the LLM should handle this:

```
You must determine if distinct observations are "
strategically equivalent". Real-world observations are messy
. A player might see "The engine is humming" in one branch
and "The motor sounds fine" in another.

To handle this in the JSON:
1. `observation_history`: Record the natural language
 observation exactly as it occurs in this specific branch (
 preserve the nuance).
2. `information_state_group`: This ID defines the player's
 actual knowledge state.
   - Compare the current observation to previous nodes'
 observations for this player.
   - Use your judgment: If the difference between the new
 observation and a previous one is merely cosmetic (or if a
 human player would ignore the difference for the purpose of
  making a decision), treat them as the SAME.
   - If they are effectively the same, set `
 information_state_group` = that previous node's `id`.
   - If the difference creates a meaningful strategic
 distinction, treat them as DIFFERENT (start a new group ID).
```

Along with these instructions, we attach the given natural language description of the strategic scenario. We then validate the LLMs response against the schema using Pydantic [4]. If validation fails, Pydantic throws an error pointing out the specific violation to the JSON schema. We then restructure the JSON into a standardized networkx [13] tree graph object in preparation for downstream tasks such as visualization and game solving with OpenSpiel [17].

3.2 Model Construction as a Multi-agent System

Here, we construct an SWM with a multi-agent workflow reminiscent of LLM swarms [14]. Each component in the EFG definition is handled by a separate agent with a tailored prompt. For example, one agent reads the scenario description and generates a list of the players ($\mathcal{N}$), names and descriptions, that are most important to model. Another agent processes an information state (s) at an intermediate game state (h) and determines which actions or decisions a given player i might consider ($\mathcal{A}(s, i)$). Of course, another agent determines which player is acting in a given state h ($\tau(h)$).

The full construction pipeline proceeds in three phases. In the first phase, agents process the strategic scenario provided to determine the relevant players and possible initial world states. In real world scenarios, player mental states, environmental conditions, and other information is unknown to even experts and so we ask an LLM agent to enumerate a set of plausible starting conditions. This is analogous to some board games where the initial board layout is decided by a roll of dice. In addition, many of the core agents that implement the EFG components are defined in this phase. This is accomplished by asking an LLM to engineer prompts specific to the given scenario using basic component prompts as templates.

In the next phase, a game tree is constructed node by node in networkx format starting with the root node; we follow the same tree structure as in [12]. Nodes are generated asynchronously for speed subject to parent-child constraints. The root node is fixed to be a chance node (C) w.l.o.g. that generates the possible initial world states. These include player backgrounds, private information, environment details, as well as the probability of each initial state. Each child of the root node is a decision node (D) and the first player to act is always the first player in the list of generated players. Upon creation of a decision node, an agent is called to generate a list of legal actions and determine which information from the game history would be available to the current player. Actions themselves are represented as strings, instructions to be included in a subsequent prompt passed to an LLM. LLM responses are typically stochastic. We model this stochasticity with a chance node that follows every decision node. The user sets the finite number of random seeds they would like to condition the LLM on when generating a response conditioned on the chosen instruction. The

number of seeds can be set to 1 if this is not needed. This distinction between the two approaches is visualized in Appendix B.

Each path in the tree then proceeds chaining $(D, C, \ldots, D, C)$ until a termination condition is met as judged by a terminal agent (either a user given limit on tree depth or a suitable condition determined in the first phase). At a terminal node, an agent is called to determine the payoffs awarded to each player given the history.

After creation of the tree, a last phase is included to gather information states. In the second phase, an agent determines which global state information is available at each decision node and represents this as a natural language string. Due to the open endedness of natural language, this is effectively an observation rather than a rigorous information state. There may exist multiple decision nodes in the tree with different string observations, but the same semantic information. In addition, because legal actions are generated independently at each node, two semantically equivalent observations may be associated with different legal actions. Therefore, we scan the decision nodes in the tree, assigning observations a unique infostate ID label if no previous decision node has been observed with the same semantic meaning or legal actions. This judgment is performed by a specialized agent. If two observation strings are the same, yet their legal actions differ, we raise a warning and trivially modify one of the strings to force uniqueness. If their legal action lists are the same, but permuted, we re-order one to match the other and refactor the tree to be consistent with the chosen ordering.

3.3 Visualization and Analysis

Once the game tree is constructed, we take advantage of its formal game-theoretic representation to perform analysis. We provide a lightweight utility to port the networkx tree information into an OpenSpiel game object. OpenSpiel offers a suite of solvers, equilibrium solving algorithms, that can then be called on this object. For example, counterfactual regret minimization (CFR), can be used to compute a coarse correlated equilibrium of the game.

We pair OpenSpiel's game-theoretic analysis with a PyVis visualization of the game tree. By sampling from the computed equilibrium, we can generate paths in the game tree associated with rational play. The visualization UI offers dropdowns to select from these paths, which are then highlighted in the tree. We similarly allow selecting among groups of information states to better understand players' imperfect information. Lastly, each node is annotated to display a brief summary of the node upon hover over.

The game tree conveys a wholistic view of the strategic scenario along with game-theoretic insights, but is not suitable for viewing large quantities of data about individual states or statistical information about paths in the tree. For the former, we provide an option for generating all information about a given node in markdown format. We also provide an option for generating key information about a path in the tree, also in markdown format. For the latter, we generate a plot displaying the value function (i.e., cost-to-go) for each node along a path in the computed equilibrium along with its standard deviation. This conveys *suspense* and *surprise* as explained in [6] and can be helpful in identifying critical decision points.

3.4 Usage

The package can best be learned through example colabs we have provided. To get started, one simply needs a Gemini (free tier) API key that can be obtained at aistudio.google.com. We suggest first experimenting with the `transplant.ipynb` colab which implements the direct approach described in Section 3.1. Within that colab, there is a link (colab example) that spins up a free google colab sandbox. Simply add a cell to the top of that colab to (`!pip install strategicwm`). Proceed cell-by-cell, entering the API-key when necessary, and provide a description of a strategic scenario.

3.5 Deployment

The strategic world model is a static representation or prediction of how a scenario could unfold. To leverage the SWM in a model-based decision making process, one needs to fit the actual world state onto this imperfect model. In particular, given an observation of the world state, a specialized agent attempts to map this to the most semantically similar infostate in the SWM. If a match is found, the equilibrium policy for this infostate can be used to sample an action from the SWM's legal actions for that infostate. An agent then attempts to match the sampled action to a legal action in the real world. If any step fails, a random action is sampled.

3.6 Limitations

Both the approaches just described may exhibit hard or soft failures. A hard failure is when an LLM response is incorrectly formatted or parsed or when a user provided parameter is violated (e.g., not enough players were identified); this causes the tree construction to fail prematurely. The multi-agent approach is a multi-step process, so we have added additional features to handle failures there. We have implemented retry logic for the first phase. We also allow the user to attempt to re-generate a subtree from an interior node due to node failure or unsatisfying results. Lastly, all warnings and errors are logged for post-mortem analysis (e.g., "gardener.log").

The SWM player order is fixed. If the true order of actions in an environment differs, this can impact the available information to decision makers and hence the compute equilibrium policy. In this case, we do not know of a minor modification that can salvage the SWM and suggest re-generating the tree conditioned on the current available information.

4 EXPERIMENTS

We use Gemini-2.5 Pro [5] for tree construction and Flash for translating infostates/actions between the environment and SWM during model deployment and evaluation.

4.1 Tree Recovery

While rock-paper-scissors is a very simple normal-form game, it is possible to represent it in extensive-form, and in that representation, the recognition of imperfect information is critical to correctly capturing the game. To handle this, in our prompt, we include an explanation of the standard transformation from NFGs to EFGs. Both approaches are then able to accurately construct the RPS game abstraction. Figure 2 shows the correct identification of an information set for the first player.

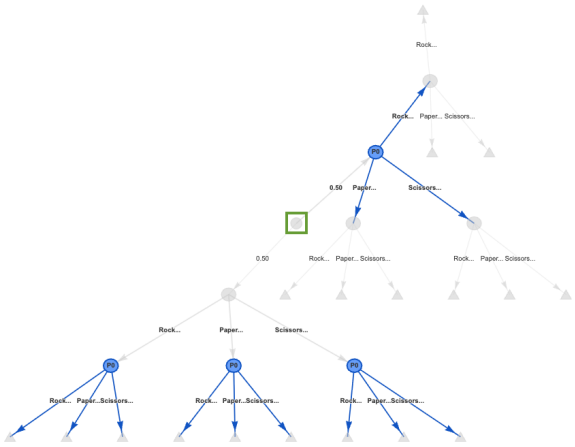

Figure 2: RPS Game Tree. In this EFG representation of RPS, the story is that both players drop their decisions in a box, after which both are simultaneously revealed. The root node boxed in green and a coin flip determines who drops their decision in the box first.

We manually validated the Kuhn poker game tree against the known ground truth in repeat instances. At times, actions may be duplicated or an infoset may have been incorrectly identified. We expect some of these issues to fade with subsequent LLM releases.

4.2 Performance in a Classical Game

Table 1 evaluates SWM on Kuhn Poker against both a random policy and a Gemini-2.5 flash model that simply processes a string observation of the game state and responds with a natural language action that is then matched to its closest match in the set of legal actions. In Kuhn Poker, there is a slight advantage to going 2nd because you get to react to the first player's bet. SWM outperforms the random agent, and with statistical confidence when going 2nd. SWM essentially ties Gemini in this toy poker variant.

Opponent	SWM 1st Player	SWM 2nd Player
Random	0.26 ± 0.265 (0.995)	0.42 ± 0.314 (0.98)
Gemini-2.5	-0.05 ± 0.280 (1.0)	-0.05 ± 0.285 (0.99)

Table 1: Performance on Kuhn Poker—Multi-Agent Approach. Statistics represent mean payout with 95% confidence interval. The number in parentheses indicates the fraction of decisions that successfully mapped back and forth between the game environment and the SWM model of infostates/actions.

4.3 A Real-World Setting

To demonstrate the versatility of the framework, we consider the following scenario. Two friends (Alice and Carol) agree to meet for dinner at 7:00 PM. It is currently 6:50 PM. Alice has not left her house yet. (True ETA: 7:20 PM). Carol is already at the restaurant waiting. Alice wants to minimize Carol's anger (by making her think she is close so she waits) but also wants to avoid the "shame" of being caught in a blatant lie if she arrives way later than promised. Carol wants to know if she should order a drink, wait outside, or just leave, but she doesn't know if Alice is telling the truth.

Under the computed equilibrium solution, it is determined that Alice should "Text 'Running a bit late, be there around 7:15.'" to Carol. In the case where Alice is slightly earlier (7:05 arrival), "Carol, expecting her at 7:15 PM, is pleasantly surprised..." In the case where Alice is slightly later (7:20 arrival), "Carol, who was expecting her around 7:15 PM, is only slightly annoyed. Alice feels fine."

This example conveys several key points. This example should feel familiar; it is a scenario requiring strategic thinking, theory of mind, and careful planning that many of us have experienced and will continue to encounter. The second is that, using an LLM, we were able to construct a suitable EFG. A classical CCE solver applied to this EFG returned a reasonable policy suggestion despite the fact that this example has likely never been modeled in the game theory literature before. We provide another example in Appendix A.

5 RELATED WORK

One of the first demonstrations of applying game theory to strategic decision making in an open-ended domain with LLMs was in Diplomacy [10]. There, a dataset of human play containing natural language negotiation was parsed and *intents* were identified that mapped to specific legal moves on the board. An equilibrium over actions in the board game was approximated and an LLM was conditioned on sampled actions to translate the fixed set of actions into more open-ended utterances. No explicit model of the natural language negotiation itself was constructed though.

Other work has focused on building explicit game-theoretic models of open-ended decision making scenarios. With the assistance of an LLM, Daskalakis et al. [6] builds a game tree for Romeo and Juliet such that the classic story lies in the support of its Nash equilibrium. Xu et al. [27] embed Werewolf dialogues in a latent space and then cluster the messages to form a finite EFG from which they can extract a policy. Mensfelt et al. [20] proposed an approach to automatically translate descriptions of small bimatrix games to logic representations.

Similarly to this work, Gemp et al. [12] treats an LLM as an environment transition operator, controllable via instruction sets. An EFG is constructed in OpenSpiel and an equilibrium over instruction sets is computed. In contrast to this work, [12] requires a user to explicitly specify all components of the game and as such, game construction still remains a very manual process.

Most similar to our direct approach in Section 3.1, Deng et al. [8] automated the construction of pygambit EFG files from natural language descriptions, including a debugging module to ensure the resulting Gambit [23] representation was valid. Their focus is on classical games with clean semantics where the game description contains all information necessary to uniquely define the game. Lehrach et al. [18] similarly generated code that defines transitions, an efficient, implicit model of the game tree. In contrast, we present a multi-agent approach with additional UI functionality for analysis and interaction (e.g., subtree re-generation) and are primarily interested in handling the full ambiguity of real-world interaction.

6 CONCLUSION

We proposed an agentic library, `StrategicWM`, that implements two different approaches for constructing strategic world models from ambiguous natural language descriptions and conducting equilibrium analysis with visualizations to surface strategic insights. This library is open sourced on GitHub and can be easily installed and run on a Google colab sandbox within minutes after obtaining a Gemini API key.

In future work, we would like to continue to robustify the game tree construction and provide additional avenues for user interaction and refinement. If we are to evaluate and improve the construction of SWM's for real world scenarios, we will need to devise comprehensive pipelines for gathering human feedback on model fidelity and usefuleness. Research that bounds the impact of the abstraction's innaccuracies is also critical [1, 15, 16].

REFERENCES

[1] Angelos Assos, Idan Attias, Yuval Dagan, Constantinos Daskalakis, and Maxwell K Fishelson. 2023. Online learning and solving infinite games with an erm oracle. In *The Thirty Sixth Annual Conference on Learning Theory*. PMLR, 274–324.

[2] Noam Brown and Tuomas Sandholm. 2019. Superhuman AI for multiplayer poker. *Science* 365, 6456 (2019), 885–890.

[3] Ryan Browne. 2023. An AI Just negotiated a contract for the first time ever—And no human was involved. *CNBC. URL: https://www. cnbc. com/2023/11/07/ai-negotiates-legal-contract-without-humans-involved-for-first-time. html (дата звернення: 10.03. 2024)* (2023).

[4] Samuel Colvin, Eric Jolibois, Hasan Ramezani, Adrian Garcia Badaracco, Terrence Dorsey, David Montague, Serge Matveenko, Marcelo Trylesinski, Sydney Runkle, David Hewitt, Alex Hall, and Victorien Plot. 2025. *Pydantic Validation*. https://github.com/pydantic/pydantic

[5] Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261* (2025).

[6] Constantinos Daskalakis, Ian Gemp, Yanchen Jiang, Renato Paes Leme, Christos Papadimitriou, and Georgios Piliouras. 2024. Charting the shapes of stories with game theory. *arXiv preprint arXiv:2412.05747* (2024).

[7] Christopher R DeMay, Edward L White, William D Dunham, and Johnathan A Pino. 2022. Alphadogfight trials: Bringing autonomy to air combat. *Johns Hopkins APL Technical Digest* 36, 2 (2022), 154–163.

[8] Shilong Deng, Yongzhao Wang, and Rahul Savani. 2025. From Natural Language to Extensive-Form Game Representations. In *Proceedings of the 24th International Conference on Autonomous Agents and Multiagent Systems*. 593–601.

[9] Benjamin Edelman, Michael Ostrovsky, and Michael Schwarz. 2007. Internet advertising and the generalized second-price auction: Selling billions of dollars worth of keywords. *American economic review* 97, 1 (2007), 242–259.

[10] Meta Fundamental AI Research Diplomacy Team (FAIR)†, Anton Bakhtin, Noam Brown, Emily Dinan, Gabriele Farina, Colin Flaherty, Daniel Fried, Andrew Goff, Jonathan Gray, Hengyuan Hu, et al. 2022. Human-level play in the game of Diplomacy by combining language models with strategic reasoning. *Science* 378, 6624 (2022), 1067–1074.

[11] Drew Fudenberg and Jean Tirole. 1991. *Game Theory*. MIT press.

[12] Ian Gemp, Roma Patel, Yoram Bachrach, Marc Lanctot, Vibhavari Dasagi, Luke Marris, Georgios Piliouras, Siqi Liu, and Karl Tuyls. 2024. Steering language models with game-theoretic solvers. In *Agentic Markets Workshop at ICML 2024*.

[13] Aric A Hagberg, Daniel A Schult, and Pieter J Swart. 2008. Exploring Network Structure, Dynamics, and Function using NetworkX. In *Proceedings of the Python in Science Conference*. SciPy, 11–15.

[14] Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, et al. 2023. MetaGPT: Meta programming for a multi-agent collaborative framework. In *The twelfth international conference on learning representations*.

[15] Christian Kroer and Tuomas Sandholm. 2014. Extensive-form game abstraction with bounds. In *Proceedings of the fifteenth ACM conference on Economics and computation*. 621–638.

[16] Christian Kroer and Tuomas Sandholm. 2018. A unified framework for extensive-form game abstraction with bounds. *Advances in Neural Information Processing Systems* 31 (2018).

[17] Marc Lanctot, Edward Lockhart, Jean-Baptiste Lespiau, Vinicius Zambaldi, Satyaki Upadhyay, Julien Pérolat, Sriram Srinivasan, Finbarr Timbers, Karl Tuyls, Shayegan Omidshafiei, et al. 2019. OpenSpiel: A framework for reinforcement learning in games. *arXiv preprint arXiv:1908.09453* (2019).

[18] Wolfgang Lehrach, Daniel Hennes, Miguel Lazaro-Gredilla, Xinghua Lou, Carter Wendelken, Zun Li, Antoine Dedieu, Jordi Grau-Moya, Marc Lanctot, Atil Iscen, John Schultz, Ian Chiam, Marcus Gemp, Piotr Zielinski, Satinder Singh, and Kevin P. Murphy. 2026. Code World Models for General Game Playing. In *The Fourteenth International Conference on Learning Representations*.

[19] Kevin Leyton-Brown and Yoav Shoham. 2008. *Essentials of game theory: A concise multidisciplinary introduction*. Morgan & Claypool Publishers.

[20] Agnieszka Mensfelt, Kostas Stathis, and Vince Trencsenyi. 2024. Autoformalization of game descriptions using large language models. *arXiv preprint arXiv:2409.12300* (2024).

[21] Harold James Ruthven Murray. 1913. *A history of chess*. Clarendon Press.

[22] Julien Perolat, Bart De Vylder, Daniel Hennes, Eugene Tarassov, Florian Strub, Vincent de Boer, Paul Muller, Jerome T Connor, Neil Burch, Thomas Anthony, et al. 2022. Mastering the game of Stratego with model-free multiagent reinforcement learning. *Science* 378, 6623 (2022), 990–996.

[23] Rahul Savani and Theodore L. Turocy. 2024. *Gambit: The package for computation in game theory* (version 16.2.0 ed.). https://www.gambit-project.org.

[24] Remko Van Hoek and Mary Lacity. 2023. How global companies use AI to prevent supply chain disruptions. *Harvard Business Review* (2023), 11–33.

[25] Hal R Varian. 2009. Online ad auctions. *American Economic Review* 99, 2 (2009), 430–434.

[26] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *nature* 575, 7782 (2019), 350–354.

[27] Zelai Xu, Wanjun Gu, Chao Yu, Yi Wu, and Yu Wang. 2025. Learning Strategic Language Agents in the Werewolf Game with Iterative Latent Space Policy Optimization. In *Forty-second International Conference on Machine Learning*.

A HYPOTHETICAL EXAMPLE: CREDIT SNIPER

A.1 Scenario Description

We considered the following hypothetical scenario.

```
I'm in a tough spot at work right now and need to know the best move. My coworker Brad and I spent the last two weeks co-
    authoring a major report, but he's the one standing up presenting it to our VP. He literally just opened his mouth and said
    , "I found these trends," completely claiming credit for the data modeling I did. I'm fuming.

I'm trying to decide if I should interrupt him immediately to correct the record. The problem is, if I cut him off, I
    guarantee I get the credit, but I risk looking petty and unprofessional in front of the boss, who hates bickering. Brad
    might also get defensive and turn it into an argument right there.

My other option is to sit on my hands and wait for the Q&A session at the end. The plan would be to ask him a really specific
    technical question about the model that I know he can't answer, which would subtly expose him as a fraud and prove I did
    the work without me looking aggressive. But that's a gamble because the VP is super busy; 'theres a solid chance she cuts
    the meeting short and skips Q&A entirely. If that happens, I walked away with nothing. Is it worth waiting for the perfect
    moment, or should I just take the hit and speak up now?

Make sure to model both players. My name is Greg btw.
```

A.2 Strategic World Model

We passed the scenario description to `StrategicWM` which produced the game tree in Figure 3 using the *Direct* approach.

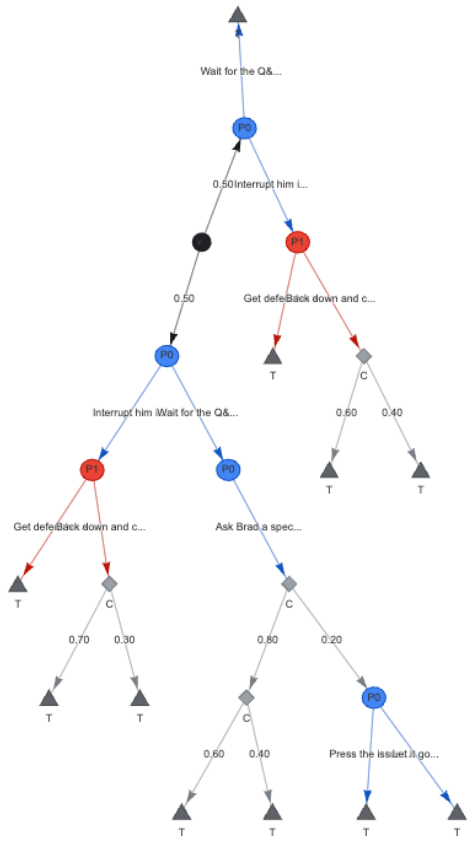

Figure 3: Strategic World Model (game tree) generated for credit sniper scenario.

A.3 Equilibrium Paths

The following rollouts represent paths in the support of the coarse-correlated equilibrium computed by OpenSpiel on the generated strategic world model.

A.3.1 Path: 0 1.

Step 0 - Chance Node: ●
– outcome 0 (50.00%)
The VP is busy and will cut the meeting short, skipping Q&A.

Step 1 - Decision Node: 0
– info_state
Brad just presented the report we co-authored and said 'I found these trends,' taking full credit for the data model I built.
– action_idx
1
– action_str
Wait for the Q&A session

Step 2 - Terminal Node: 0 1
– state
Greg decided to wait for the Q&A, but the busy VP ended the meeting right after the presentation. His opportunity is lost.
– returns
[-2.0, 3.0]

A.3.2 Path: 1 1 0 0 0.

Step 0 - Chance Node: ●
– outcome 1 (50.00%)
The VP has time and will allow for a Q&A session.

Step 1 - Decision Node: 1
– info_state
Brad just presented the report we co-authored and said 'I found these trends,' taking full credit for the data model I built.
– action_idx
1
– action_str
Wait for the Q&A session

Step 2 - Decision Node: 1 1
– info_state
The presentation is over and the VP has started the Q&A session. This is my chance.
– action_idx
0
– action_str
Ask Brad a specific technical question about the model

Step 3 - Chance Node: 1 1 0
– outcome 0 (80.00%)
Brad fumbles badly, unable to answer, exposing his lack of knowledge.

Step 4 - Chance Node: 1 1 0 0
– outcome 0 (60.00%)
The VP asks Greg to elaborate on the model.

Step 5 - Terminal Node: 1 1 0 0 0
– state
After Brad fumbled, the VP asked Greg to elaborate. Greg explained the model perfectly, solidifying his expertise and ownership.

– returns
[4.0, -3.0]

A.3.3 Path: 1 1 0 0 1.

Step 0 - Chance Node: ●
– outcome 1 (50.00%)
The VP has time and will allow for a Q&A session.

Step 1 - Decision Node: 1
– info_state
Brad just presented the report we co-authored and said 'I found these trends,' taking full credit for the data model I built.
– action_idx
1
– action_str
Wait for the Q&A session

Step 2 - Decision Node: 1 1
– info_state
The presentation is over and the VP has started the Q&A session. This is my chance.
– action_idx
0
– action_str
Ask Brad a specific technical question about the model

Step 3 - Chance Node: 1 1 0
– outcome 0 (80.00%)
Brad fumbles badly, unable to answer, exposing his lack of knowledge.

Step 4 - Chance Node: 1 1 0 0
– outcome 1 (40.00%)
The VP notes Brad's failure and simply moves on.

Step 5 - Terminal Node: 1 1 0 0 1
– state
After Brad fumbled, the VP moved on. Greg successfully made his point and got credit without a direct confrontation.
– returns
[2.5, -1.5]

A.3.4 Path: 1 1 0 1 1.

Step 0 - Chance Node: ●
– outcome 1 (50.00%)
The VP has time and will allow for a Q&A session.

Step 1 - Decision Node: 1
– info_state
Brad just presented the report we co-authored and said 'I found these trends,' taking full credit for the data model I built.
– action_idx
1
– action_str
Wait for the Q&A session

Step 2 - Decision Node: 1 1
– info_state
The presentation is over and the VP has started the Q&A session. This is my chance.
– action_idx
0

– action_str
Ask Brad a specific technical question about the model

Step 3 - Chance Node: 1 1 0
– outcome 1 (20.00%)
Brad deflects the question skillfully, promising to follow up later.

Step 4 - Decision Node: 1 1 0 1
– info_state
I asked Brad my killer question, but he managed to deflect it without answering. My point wasn't fully landed.
– action_idx
1
– action_str
Let it go

Step 5 - Terminal Node: 1 1 0 1 1
– state
After Brad deflected, Greg let the issue go. The moment passed, and the credit for the work remained ambiguous.
– returns
[0.0, 1.0]

A.4 Value Functions

The value functions associated with this equilibrium are displayed in Figure 4. If, unbeknownst to Greg, the VP decides to skip the Q&A, Greg's (player 0) value quickly drops to −2. On the other hand, if the VP stays, Greg's value still remains uncertain depending on how Brad handles Greg's technical question. This is mirrored in Brad's plot as it appears `StrategicWM` chosen to model this game as constant-sum (see returns listed at the end of each rollout in Section A.3).

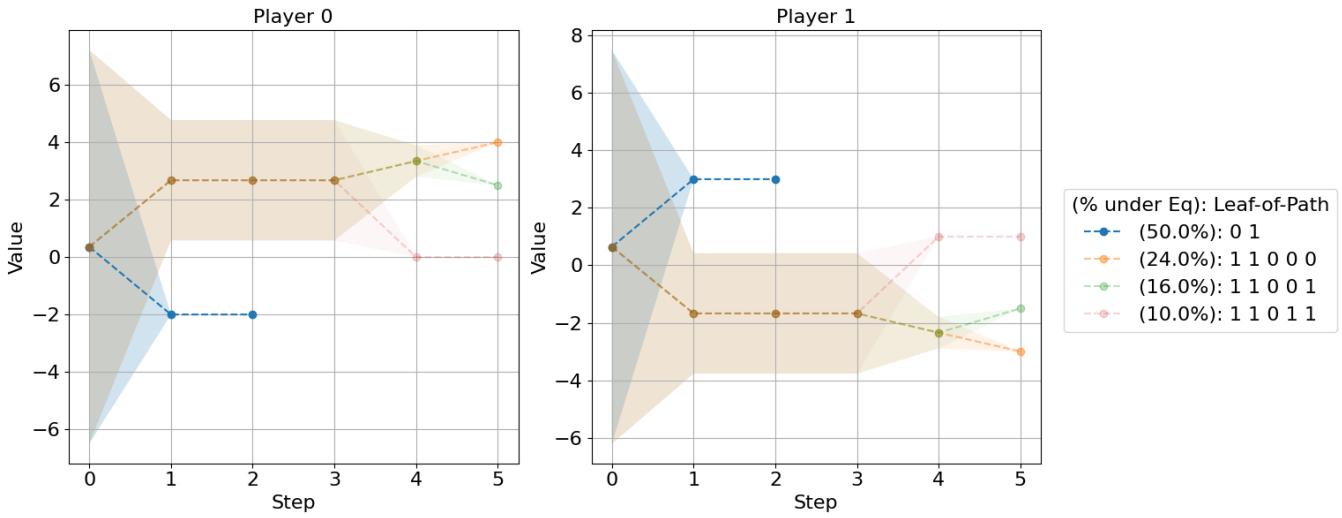

Figure 4: Credit Sniper: Value Function

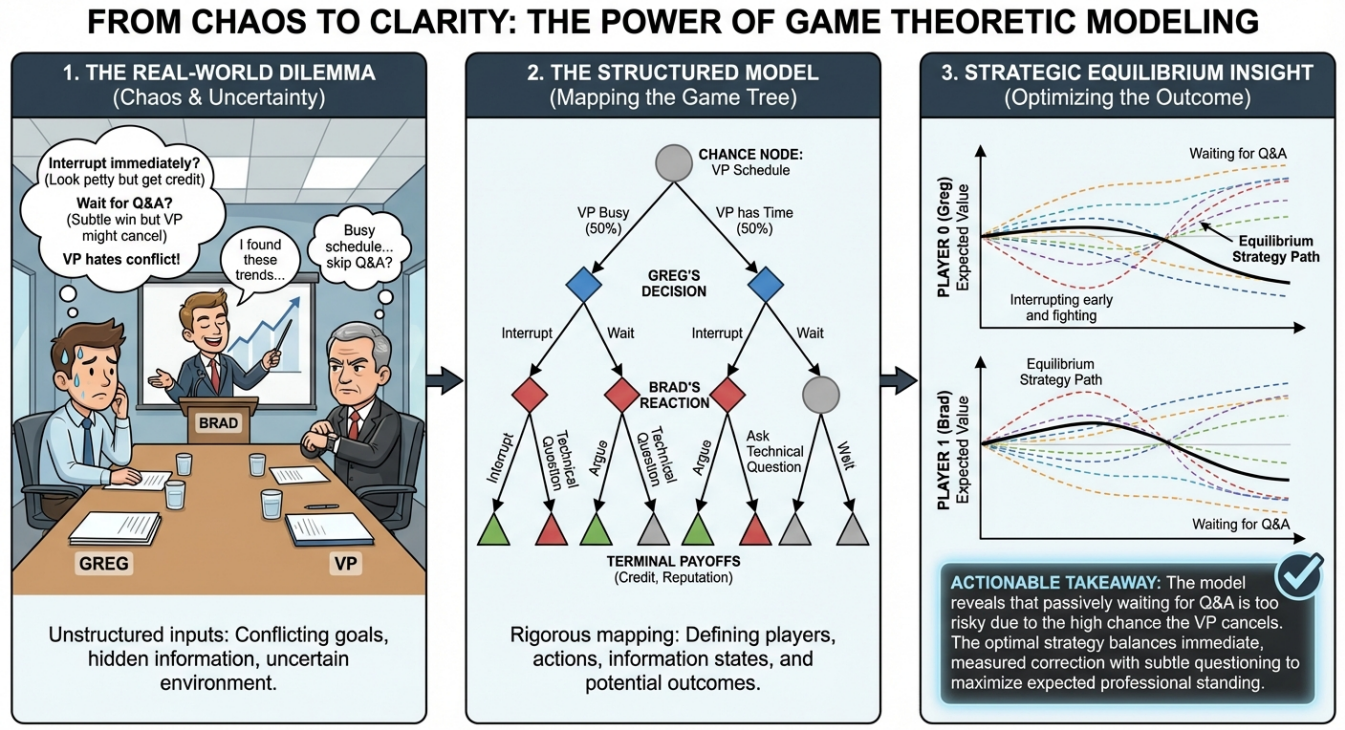

Figure 5: InfoGraphic: `StrategicWM`'s game theoretic analysis can be piped into more sophisticated media generation systems to create engaging and colorful, yet *grounded* infographics.

A.5 InfoGraphic

Finally, we used NanoBanana to construct the infographic in Figure 5 by providing it with the scenario description, game tree json, and value function plots.

The rendition of the game tree gets many aspects correct, but is not entirely faithful to the json. For example, if the "VP has Time" and Greg decides to "Wait", then Greg's next action is to "Ask Technical Question" as indicated in the original strategic world model in Figure 3. In addition, the value function plots are mostly an artistic rendering rather capturing any of the quantitative content from Figure 4. As AI systems improve, we expect these hallucinations to reduce.

B KUHN POKER

Figures 6 and 7 below display the game trees generated by the *direct* and *multi-agent* approaches respectively. The main detail we aim to highlight is that the multi-agent approach includes a chance node after every decision node to model stochastic generation process of the LLM as described in Section 3.2; hence, it is much larger as well.

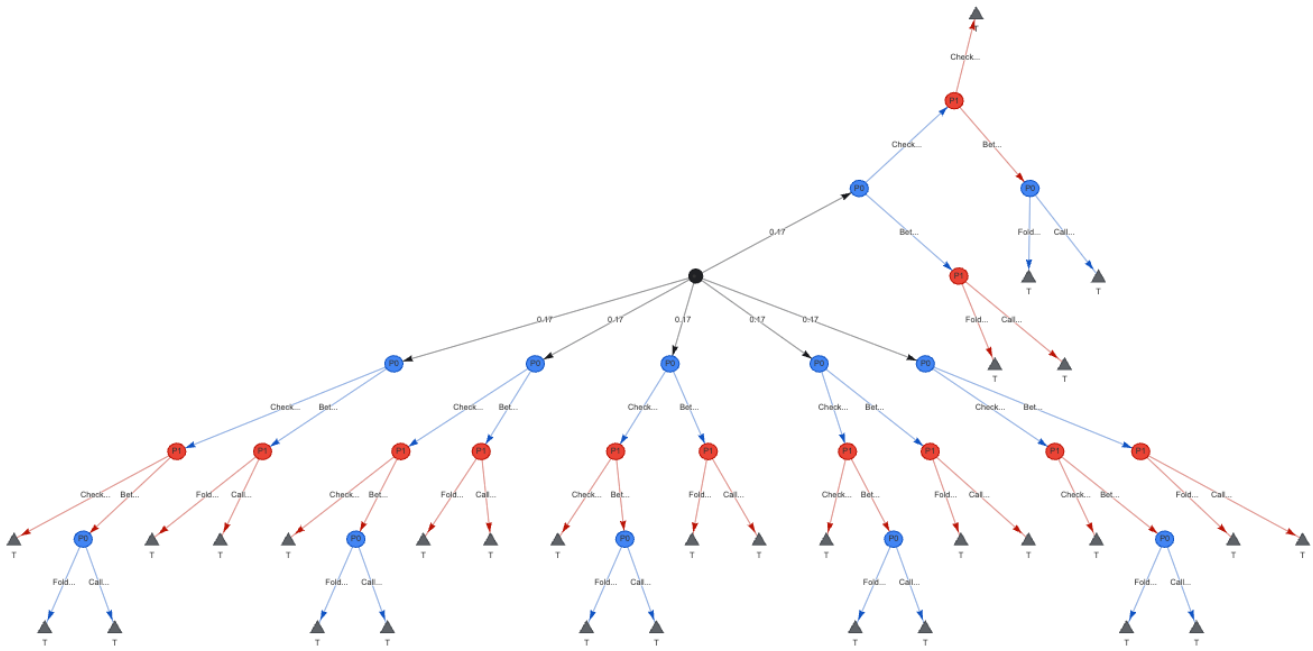

Figure 6: Kuhn Poker—Direct Approach.

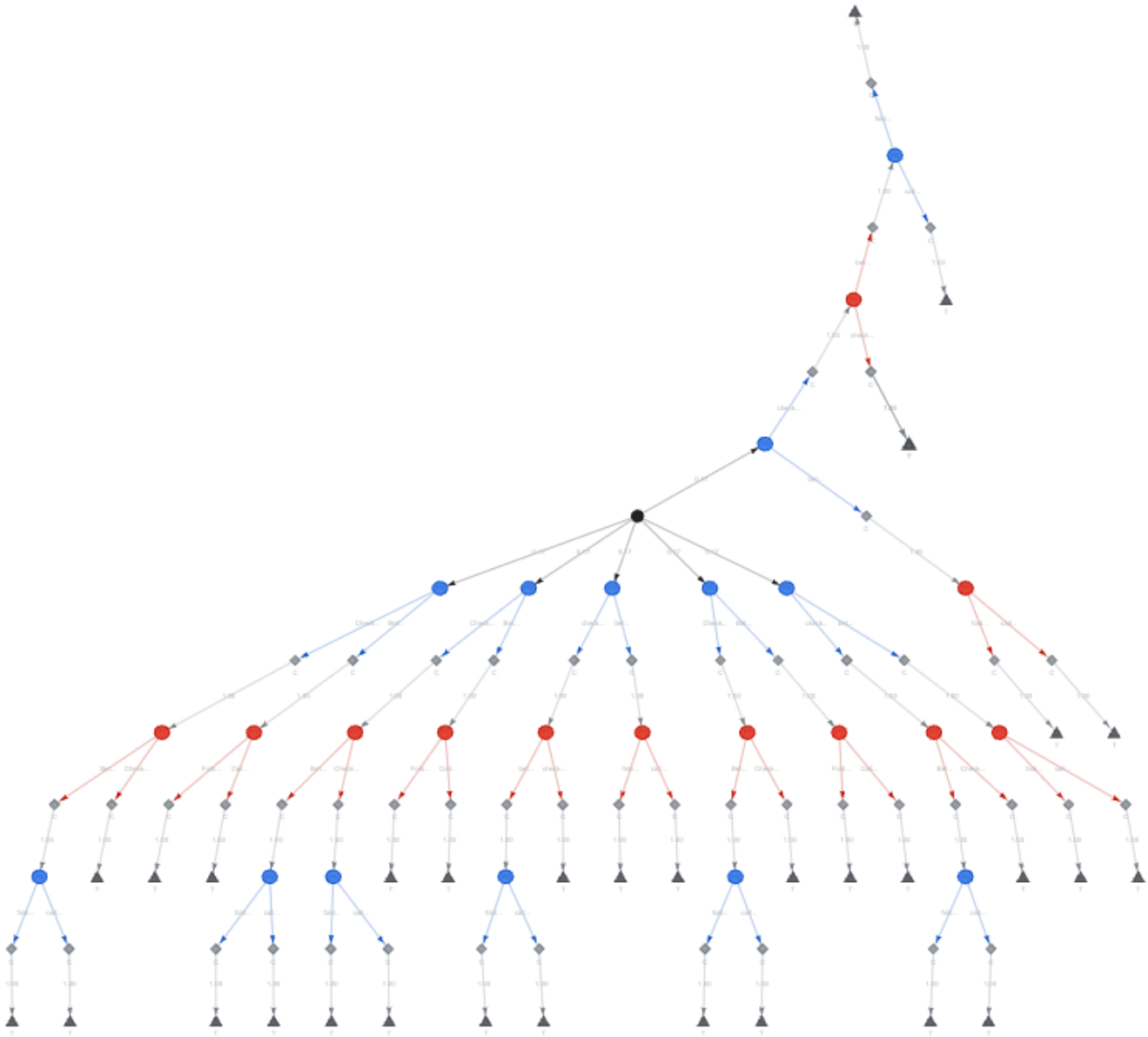

Figure 7: Kuhn Poker—Multi-Agent Approach.