Dynamic Epistemic Logic Modeling in Clue

Travis Litke
COSI112a
Pustejovsky

Abstract

This report describes the process, application, and findings from an attempt to simulate the progressive updates of Dynamic Epistemic Logic Kripke structures in a Clue board game environment. The simulation tracks a Kripke model whose possible worlds are all of the possiblities of a Clue mystery: a suspect, with a weapon, in a room. As announcements are made, accessiblity relations between worlds are eliminated, resulting in a solution based on an agent's ability to eventually distinguish between the actual world (solution) and all of the possible worlds (every combination of Clue cards).

1 Introduction

Dynamic Epistemic Logic allows for the introduction of new information and asymmetrical knowledge updates if more than one agent is involved in the model. The number of possible worlds grows exponentially as well as the number of possible propositions increases. Clue is an appropriate framework for modeling changes in a DEL Kripke Model as it starts with each agent having access to the same possible worlds, but as the game progresses, learning facts about the other players and reducing their uncertainty about the actual world. The code accompanying this report was designed to track the evolution of a DEL Kripke model by simulating a game of Clue between 2-6 agent scripts.

2 Methodology

2.1 Overview

The script this report is based on was written completely in Python and is available at `https://github.com/travislitke/dynamic_epistemic_logic_Cluedo.git`. The main script takes arguments for different parameters of the simulation and runs the game.

2.2 The Game Loop

The game is initialized with n players. The Game objects init creates the players, a new deck, and the Kripke Model for the game. The game tracks the number of current active players, and while the number of active players is greater than one, the game loops through a ”turn” for each active player in succession.

At the beginning of the game, one of each type of card is removed from the deck: a Suspect, a Weapon, and a Character. This is saved to the game state as the Solution. These cards have been removed from the deck and will never be seen by the agents. The remaining cards are dealt to each agent and as each receives a card, that card is ”seen” by the agent, and the agent's accessibility relation in the Kripke Model is updated to reflect the event. At the end of the deal, if there are leftover cards (modulo of the number of cards by the number of players) these cards are shown to all agents. This is a public announcement that updates common knowledge: every agent knows that those cards are not in play, and won't appear in the solution.

As long as there is more than one agent remaining in the game, each agent takes a turn. A turn consists of reviewing the Kripke Model to check what the agent ”Knows”: if that agent has more than one accessibility relation in the model, the agent is unsure of what the state of the actual world is. The physical board game of Clue involves rolling dice and moving the corresponding number of spaces, trying to reach a room. In this simulation, the agent simply moves to a room that it hasn't ruled out yet, or a room that the agent ”believes” could be part of the solution to the crime. The agent makes a suggestion involving one suspect and weapon that it hasn't ruled out yet, as well as the room it has moved to. A function checks each other agent's hand (of card objects) to see if that agent is holding a card corresponding to the suggestion. The suggesting agent receives a private update from the first opponent agent to have a matching card in its hand.

This puts a limitation on DEL updates in the model. The agent is only allowed to be shown one matching card from one opponent agent. The agent can eliminate possible worlds based on that one card, but it hasn't ”seen” every card possible, so it is still unable to distinguish an actual world. The fact that the agent saw one but not the other two cards in the suggestion does not rule out the possibility of the other two cards still being in play.

Conversely, in the case that none of the agents are holding a matching card, the agent learns from no reply that none of the opponent agents are holding any card from the suggestion. This allows the agent to safely deduce that none of the cards in the suggestion are in play, and therefore must be the solution to the game. In the current state of the Clue engine, every game ends this way.

2.3 Command Line Arguments

There are three implemented command line arguments:

- Number of Players - This is the number of agents initialized in the Kripke structure. They play Clue against each other.
- Number of Simulations - This argument affords the ability to run as few or as many simulations of Clue as the user wants!
- Accessiblity Relation Logging- There are *many* updates every turn, and this argument allows the user to return each updated status in the log. This creates dense logs, so by default is disabled.

2.4 Classes

The **Game** class holds all of the game state information in its attributes, such as the state of each agent, the deck of cards in play, and the Kripke Model that is being updated every turn.

The **Agent** class has an identifier and a hand of cards. Every time it "sees" a card (through the function agent.see()), the agent's accessibility relations in the game's Kripke structure are updated to eliminate the possibility of any worlds where the card it has seen holds true.

The **Deck** class holds the Deck and Card objects used to update player knownledge each turn. The Deck holds a list of card objects to facilitate shuffling and dealing cards. As each card comes into play by being dealt, the card is removed from the deck and the recieving player's accessibility relation is updated.

The **Knowledge Graph** class is the focal point of the entire project. The Model is made up of a set of worlds, a set of propositions, a set of agents, and a set of accessibility relations between agents and worlds. This is the portion of the project that relates directly to the course material from this past semester.

2.5 Logging

The python logging library output a record of every move to a log file in order to trace the knowledge updates as they occur.

```
[INFO] Agent a3 has entered the Dining Room.
[INFO] Agent a3 suggests Mustard with the Rope
       in the Dining Room.
[INFO] Agent a2 shows Agent a3 Character: Mustard.
[INFO] Agent a3 has seen card Character: Mustard.
[INFO] Agent a3 updated accessibility relations:
[('w166', 'w3'), ('w166', 'w4'), ('w166', 'w6'), ('w166', 'w12'),
('w166', 'w13'), ('w166', 'w15'), ('w166', 'w21'), ('w166', 'w30'),
('w166', 'w40'), ('w166', 'w49'), ('w166', 'w57'), ('w166', 'w58'),
 ('w166', 'w60'), ('w166', 'w66'), ('w166', 'w67'), ('w166', 'w69'),
('w166', 'w75'), ('w166', 'w84'), ('w166', 'w94'), ('w166', 'w103'),
('w166', 'w112'), ('w166', 'w120'), ('w166', 'w123'), ('w166', 'w138'),
('w166', 'w157'), ('w166', 'w165'), ('w166', 'w166'), ('w166', 'w168'),
('w166', 'w174'), ('w166', 'w175'), ('w166', 'w177'), ('w166', 'w183'),
('w166', 'w192'), ('w166', 'w202'), ('w166', 'w211'),
('w166', 'w220'), ('w166', 'w228'), ('w166', 'w231'), ('w166', 'w246'),
('w166', 'w265'), ('w166', 'w273'), ('w166', 'w282'), ('w166', 'w293')]
```

Figure 1: Sample output from a Clue run logging a knowledge update.

```
[INFO] Agent a1 has entered the Garage.
[INFO] Agent a1 suggests Green with the Rope in the Garage.
[INFO] All other players remain silent.
       Agent a1 has deduced the solution.
[INFO] Agent a1 has proposed ('Green', 'Rope', 'Garage').
[INFO] Agent a1 has solved the mystery.
[INFO] {'Solution': ('Green', 'Rope', 'Garage'),
       'Num Players': 3,
       'Winner': 'a1',
       'Eliminations': 0,
       'Num Rounds': 8}
2025-12-18 20:24:04 [INFO] End of simulation 1.
```

Figure 2: Sample output log from an agent deducing the solution.

3 Results

The results were less fun than the real game of Clue because the scripts that play against each other are perfect reasoners and record-keepers. At the time of writing, the "bluffing" mechanic in a physical Clue game has not been implemented. Agents are unable to mislead each other by making suggestions about cards that they already have in their hand. A propensity to bluff and cause false knowledge updates would lead to many more games ending by elimination: when an agent makes a false proposal (falsely believes that they have ruled out all worlds that don't satisfy the actual world propositions) they are removed from the game. This would allow an agent to win by lying. Additionally, this would allow the implementation of an agent learning more from follow-up suggestions. By seeing a suggestion with a proposition that has already been made, the agent could reason that another one of the agents must be lying and could update its knowledge graph accordingly. Currently, with no bluffing mechanic, each agent makes perfect guesses and is able to deduce the solution without any eliminations from bad guesses.

```
[INFO] Agent a2 has proposed ('Scarlet', 'Rope', 'Courtyard').
[INFO] Agent a2 has solved the mystery.
[INFO] {'Solution': ('Scarlet', 'Rope', 'Courtyard'),
        'Num Players': 3,
        'Winner': 'a2',
        'Eliminations': 0,
        'Num Rounds': 12}
[INFO] Final Kripke Model:
{'a1': [('w108', 'w79'), ('w108', 'w108'), ('w108', 'w151'),
('w108', 'w169'), ('w108', 'w288')], 'a2': [('w108', 'w108'),
('w108', 'w197')], 'a3': [('w108', 'w26'), ('w108', 'w62'),
('w108', 'w89'), ('w108', 'w108'), ('w108', 'w131'),
('w108', 'w135'), ('w108', 'w178'), ('w108', 'w189'),
('w108', 'w230'), ('w108', 'w260'), ('w108', 'w275'),
('w108', 'w296')]}
```

Figure 3: Resulting final Kripke Model from a complete run. Notice an interesting bug where the winning agent still technically has two possible worlds. Removing that final world results in an endless loop, a problem that I would love to get to the bottom of.

4 Discussion

I wanted to go a lot further with this project than I had the time or capacity to go. The logic was quickly complicated by the addition of deduction, where agents win the game by asking about cards that are not in play. Implementing a bluffing mechanic would be the most interesting addition to this engine, but it would create meta-layers of knowledge tracking. New domains would be added such that each agent would need to track whether or not they know another agent has bluffed, which would involve writing logic to reason with a history of card suggestions to recall what had already been announced and reasoning the truth, resulting in a private knowledge update. This could also allow for the implementation of confidence: based on an agent's previous actions, how likely is another agent to "believe" or accept their suggestion as a concrete knowledge update. Even further, an agent could decide whether or not to bluff based on its confidence in its knowledge of an opponent's hand as opposed to implementing a random coin toss for bluffing or not, since the drawback of a bluff is that the agent misses out on a potential update by seeing another players card. A bluff involves the agent suggesting a card it already holds, meaning that it has already made that knowledge update and is not gaining new information but misleading other agents.

5 Conclusion

In its current state, the scripts play Clue perfectly because there is very little decision making. As each update is made, the game is essentially undergoing a trivial ”process-of-elimination”. In the future, after implementing bluffing, the next logical step will be to incorporate agentic language models and prompt each to make decisions every turn based on the current state of the game, i.e. the Kripke Model.