NARS in Social Deduction Games

Edward Sharick

Temple University

CIS 5603 - Artificial Intelligence

May 2, 2022

Abstract

Implementing and testing artificial intelligent systems in games has become commonplace as most games, such as Chess or Go, have a clearly defined environment, the state of the game is easy to represent in a computer program, and the system can 'learn' winning strategies in a stable world where all the knowledge and information is known. However, in many real world applications, not all the knowledge or information is known, and often there is also unclear or possibly deceptive information. In social deduction games, players attempt to uncover each other's hidden role or team allegiance, using logic and deductive reasoning, or lying and bluffing to deceive other players. This project attempts to use NARS, a non-axiomatic reasoning system, to explore how well an AI system can learn to play a social deduction reasoning game called "The Resistance".

Introduction

A major goal of artificial intelligence is to be able to create a system which is adaptable and capable of handling many real-world situations, where things are often unpredictable or unknown; where other agents don't always act logically; where we must make approximations or best guesses so far. A good training ground for an AI system in tackling problems in these unknown or unpredictable environments is in social deduction games. In the game by Proavalon[3], "The Resistance", there are rules for the players to follow and some things are predictable, but there is also hidden information, bluffing, lying, and deception. This project aims to use NARS[4] to observe the game play and attempt to determine the hidden information in the game.

Rules of "The Resistance"

"The Resistance is a game played between 5 and 10 players. There are two teams, the spies, and the resistance, and each player is given a hidden role they keep secret from the other players which determines their team. About one-third of the players are spies, for example, in a 5 and 6 player game, there are 2 spies, and for a 7, 8 and 9 player game, there are 3 spies. For a 10 player game, there are 4 spies. After each player is assigned their role, all players close their eyes, and then the spies open their eyes, revealing their identity to the other spies. Thus, at the beginning of the game, the spies know who the other spies are, but the resistance members only know their own identity.

The game is played over the course of 5 missions, and the goal of the game for the resistance is to pass 3 missions, and the goal of the game for the spies is to fail 3 missions. Each mission team is a subset of the players, where the size of the mission team is determined by the round and the number of players. The leader for the round proposes a team and then everyone simultaneously and publicly votes on either approving or disapproving the team. If the majority approve of the team, the game proceeds to the mission team. If the majority (or tie) disapprove of the mission, the leader passes to the next player. The spies also win if they get 5 disapproving votes in a row.

Once a mission team is approved, each member is given a pass or fail card, and secretly selects whether they want to pass or fail the mission. The resistance members must always pass the mission, but spies have the option to fail or pass the mission. If there is a failing vote on the mission, the mission fails. The players do not see who played which card, they only know that of the subset of players on the mission team. So, for a failing mission, they know that one of the players put in a fail card, but they don't know exactly who that player was. At any point in time in the game, players may say anything to any other player, but that information must be said publicly. This often looks like people accusing or vouching for other players, with the resistance members trying to find holes in other player's logic, and the spies trying to concoct a story that pins the blame on someone else.

Previous Research

Researchers in the Artificial Intelligence field have looked into the area of social deduction games before, although no research has been done specifically on "The Resistance" or using NARS. One example of AI in social deduction games was done by Eger and Martins[1] (2018) when they compared AI commitment strategies for a particular social deduction game, One Night Ultimate Werewolf. Their AI comes up with a plan to "convince" the other agents of their own role and can decide to change plans or stay committed to their current story. One aspect of their research that was used in this project was how they implemented a variation of the game One Night Ultimate Werewolf, making it turn-based and limiting the possible questions and responses the agents may use. Similarly, this project used a turn-based variation of "The Resistance" in order to simplify what NARS needed to learn abou the game.

Another group, Kopparapu et al. (2022) used Deep Learning to show that reinforcement strategies could be used to give an agent the ability to learn strategies and play a game called "Hidden Agenda" effectively. This research was not used in this project, as NARS is quite different than how one uses deep neural networks to solve problems.

Methods

In this section, the methodology of the project is discussed, including the AI system used, NARS, and how the game was conceptualized and the procedure for getting NARS to observe the game.

Artificial Intelligence System - NARS

“NARS (Non-Axiomatic Reasoning System) is a project aimed at the building of a general-purpose intelligent system that follows the same principles as the human mind, and can solve problems in various domains.”[4] NARS is fundamentally different from traditional reasoning systems, mainly because of its assumption of insufficient knowledge and resources. The logic of NARS is named ”Non-Axiomatic Logic” (NAL), because none of the knowledge it processes (as premise or conclusion) can be considered as ”axiom”, as with a fixed truth-value. Instead, the system’s beliefs are summaries of the system’s experience, and are always revisable.

NARS processes tasks imposed by its environment, in this case, these are Narsese input statements; Narsese is the language of NARS. The task in this case are statements about the game and questions, specifically which players are spies. NARS repeatedly carries out the working cycle of the system[5]:

1. Select tasks in the buffer to insert into the corresponding concepts, which may include the creation of new concepts and beliefs, as well as direct processing on the tasks.
2. Select a concept from the memory, then select a task and a belief from the concept.
3. Feed the task and the belief to the inference engine to produce derived tasks.
4. Add the derived tasks into the task buffer, and send report to the environment if a task provides a best-so-far answer to an input question, or indicates the realization of an input goal.
5. Return the processed belief, task, and concept back to memory with feedback.

This project uses OpenNars[5], an open-source Java implementation and the source code can be found here: `https://github.com/opennars/opennars`. The reason for choosing NARS is that in social deduction games, very little information is known, and as new evidence comes to light, your beliefs may change. In future implementations of this project, when attempting to play the game (and not just observe), the starting beliefs of the system can be different for NARS as a spy or as a member of the resistance. Each action in the game can be processed as a task and these actions will be processed and change NARS’ beliefs about other players. NARS will have some concepts such as game, player, spy, the rounds of the game, the selected team (subset of players), how agents voted, whether the mission passed/failed, etc. Then NARS can use the rules of the game to try to make decisions about which players it believes to be spies.

Procedure

The project workflow can be broken into the following distinct steps (see Figure 1 below). The individual steps are discussed in more detail in the subsections below. As mentioned in the previous section, the OpenNars system processes input as Tasks in a language called Narsese. Thus, the first step of this procedure was to be able to break a game of resistance into English statements. This English summary could then be translated into Narsese. Steps 3 and 4 involved writing the game rules and questions to answer in Narsese. Finally, all this information was fed into NARS and the answers to the questions were outputted in Narsese. The output data was parsed and the results were collected. The results are analyzed in the section below.

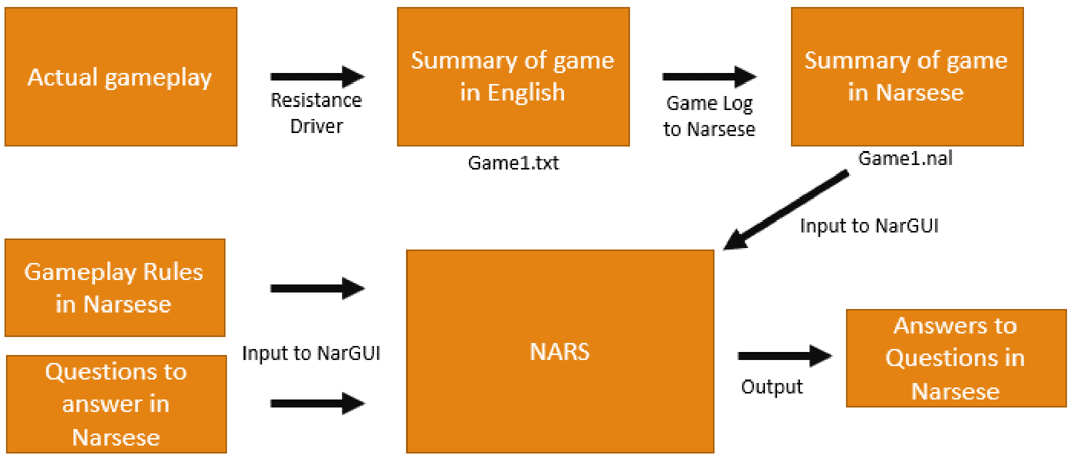
Actual gameplay
Resistance Driver
Summary of game in English
Game1.txt
Game Log to Narsese
Summary of game in Narsese
Game1.nal
Input to NarGUI
Gameplay Rules in Narsese
Questions to answer in Narsese
Input to NarGUI
NARS
Output
Answers to Questions in Narsese

Figure 1: Project workflow.

Real-time game to English summary

The first step in the procedure was to take actual game play and create a summary of the game in English. A difficulty in this step was in deciding what information was important to keep and what could be discarded. A sample of a game in English can be found in Appendix A of the report and a flowchart showing the general steps of the game play can be found in figure 2 below. The first thing we

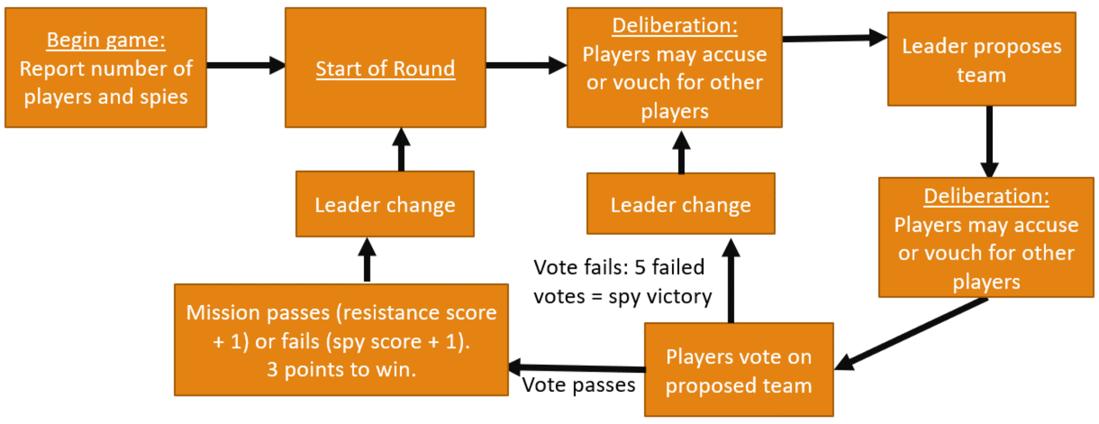
Begin game: Report number of players and spies
Start of Round
Deliberation: Players may accuse or vouch for other players
Leader proposes team
Leader change
Leader change
Deliberation: Players may accuse or vouch for other players
Vote fails: 5 failed votes = spy victory
Mission passes (resistance score + 1) or fails (spy score + 1). 3 points to win.
Vote passes
Players vote on proposed team

Figure 2: Diagram of game play.

have is the number of players and the number of spies. In training games, we list who the spies are, but in testing games, we don't include this information. This allows NARS to be able to answer prompts about how sure it is that each player is a spy. The game goes through the rounds, states who the leader is and what the score is. There is a deliberation phase where players can vouch for other players or make accusations. Then a team is proposed by the leader, another deliberation phase happens, and the players vote on the proposed team. A simplification I made to the original game was that instead of allowing the players to say anything, I kept the deliberation to two simple statements: A accuses B and A vouches for B. In the real time game, players don't deliberate in order, but instead may speak up at any time. Furthermore, players may read each other's facial expression, body language or other tells, which can give more information to the player. As my game is a text-based implementation, I cannot process these subtleties, which make it more difficult to determine who the spies are. As can be seen in the results section below, sometimes NARS is able to figure out who the spies are, and sometimes it is not. I created sample games using the "Resistance Driver.java", which also relies on the code in "Player.java", "Resistance.java", and "FileLogger.java".

English game summary to Narsese

The next step in the procedure involved taking the game summary in English and converting it into Narsese. As OpenNARS does not yet have functioning natural language processing, I manually parsed the various English statements from the game. A sample game can be found in Appendix B. Here, I will disucss some of the decisions that I made on how to parse the English sentences into Narsese. To start, each game was given a number so that NARS could link concepts such as games, rounds, mission teams, etc. together. On training games, NARS was told directly which players were spies. Here are a few of the Narsese statements from sample game 1 and their explanations.

```
<{game1} --> game>. %1.00;0.99%
```

In this Narsese statement we have that game1 is an instance of the term game. This will allow NARS to build a concept for the term 'game' with several different instances. We add the two percentages, the first is the frequency, the second is the confidence. Statements which should be considered factual should be 1.00 and 0.99.

```
<{A1} --> player>. %1.00;0.99%
<{A1} --> [leader]>. :|: %1.00;0.99%
```

In these two statements, we are declaring that there is a player called A1, and that player A1 has the property leader. We add the temporal marking : | : to indicate that it is currently the leader at this time.

```
<{team1g1} --> team>. %1.00;0.99%
<{team1g1} --> [<size --> [2]>]>. %1.00;0.99%
```

In these two statements, we are declaring a team of size 2 and giving it a name, it is the first proposed team in game 1 (team1g1).

```
<(*,{A1},{team1g1}) --> proposes>. %1.00;0.99%
<(*,{team1g1},{round1g1}) --> proposedOn>. %1.00;0.99%
<(*,{B1},{team1g1}) --> supports>. :|: %1.00;0.99%
```

In these statements, we start to combine terms and define relationships between these terms. The first statement tells NARS that A1 is the one who proposed team1g1; the second statement is that team1g1 was proposed on round1g1; and the third statement is that player B1 supports team1g1.

```
<{team1g1} --> [accepted]>. %1.00;0.99%
<{team1g1} --> [success]>. %1.00;0.99%
```

When a team is accepted (majority vote approval), we see that the team is given the accepted property. When a team is successful (passes the mission), it is given the success property.

```
<*, {A1}, (--, <{C1} --> spy>)) --> believes)>. :|: %1.00;0.99%
```

Finally, statements like these can be found during deliberation. In this statement, A1 is vouching for C1, that is, A1 believe that C1 is not a spy. These statements could be modified in future versions to try to relay confidence, such as, "I'm 75 percent sure that they are a spy". There are other statements about the game that I did not go over here, but they can be found in Appendix B.

Game play rules to Narsese

This part of the procedure required teaching NARS some of the rules of the game. Some of these rules could be modified with the frequency and confidence values, but they are general rules of thumb (or strategies) for NARS to try to follow. Further testing should be done into whether or not these rules actually help NARS to make better guesses to who the spies are. Here are a few examples:

```
<(&&,<$1 --> player>,<#2 --> team>,<#2 --> [success]>,<(*, $1, #2)
--> memberOf>) ==> (--,<$1 --> spy>)>. %0.8000002;0.9%
```

This rules gives NARS multiple things to consider, but the rule basically says that if a there is a player who is a member of a successful team, they are not likely to be a spy, 80 percent of the time. This 80 percent might be too high and more testing should be done by modifying this percentage.

```
<(&&,<$3 --> player>,<#4 --> player>,<$3 --> spy>,<(*, $3, <#4 --> spy> -->
believes>) ==> (--,<#4 --> spy>)>. %0.70;0.90%
```

This rule tells NARS that if there are two players, and one player is a spy, and that player accuses a second player of being a spy, then that other player is probably not a spy (the first player is lying). This strategy doesn't always hold true, as sometimes a spy will accuse their other spy teammate to confuse the opponents. There are some other rules that can be found in Appendix C.

Questions to answer in Narsese

The final part of the Narses input is questions. We want to be able to get NARS to tell us what it currently believes about the players in the game. Does NARS think that player A is a spy? How confident in this answer is it? It is important to be able to ask NARS these questions at various times as well, since as the game progresses, new information reveals itself and it can modify its beliefs. There is basically only one question, and another statement which helps to parse the output:

```
<{A1} --> spy>?
''outputMustContain('<{A1} --> spy>?' )
```

The first line asks NARS to consider the statement, is A1 a spy? The second statement simply tells NARS to only output an answer to that question.

Feeding the data to NARS

Once all the game data was created and translated to Narsese, and the rules and the questions were also translated, we start the NAR SGui from the OpenNARS code[5] and then feed in the information. First, we show NARS the rules and then give it some games where it knows from the beginning who the spies are, so that it can watch the behavior of those players and begin to make connections and learn the game. Then, we give it data for test games. The only difference is that NARS does not know who the spies are in this game. Then, after each round, we ask NARS to output who it thinks the spies are. It should be noted that the rules are fed to NARS before each training game, since NARS will forget things naturally, and we want to remind it of the things that are important. It still is unclear if it is most effective to give NARS all the game data at once, or feed it piece by piece and allow it to do some working cycles in between in order to further connections. More experimentation must be done.

Parsing the output data

The NARS Gui outputs a lot of information, so we constrict the output to only answering the question of whether each player is a spy or not. However, NARS does not always answer the question directly, and its answers will vary depending on what concepts and terms it is considering. This makes it challenging to get usable data, so further work needs to be done in parsing the output and reporting only the relevant information, especially to make NARS be able to play the game.

Results and Analysis

After training NARS on eight training games, its ability to determine the spies in two 5-player games was tested. NARS was asked after each game how likely it was that any given player was a spy. The results are summarized below.

```
Test Game 1 (Spies: A, B)
Is Spy?           A       B       C       D       E
Frequency         0.74    0.6     0.67    0.7     0.58
Confidence        0.82    0.86    0.82    0.79    0.86

Test Game 2 (Spies: A, C)
Is Spy?           A       B       C       D       E
Frequency         0.95    0.98    1       0.72    0.74
Confidence        0.51    0.41    0.49    0.62    0.78
```

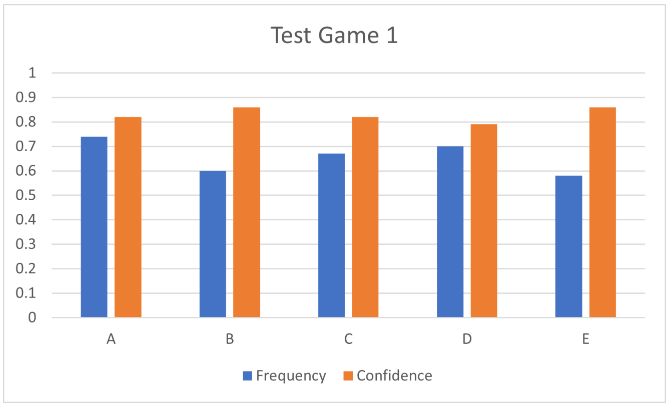

Figure 3: Test Game 1 Results.

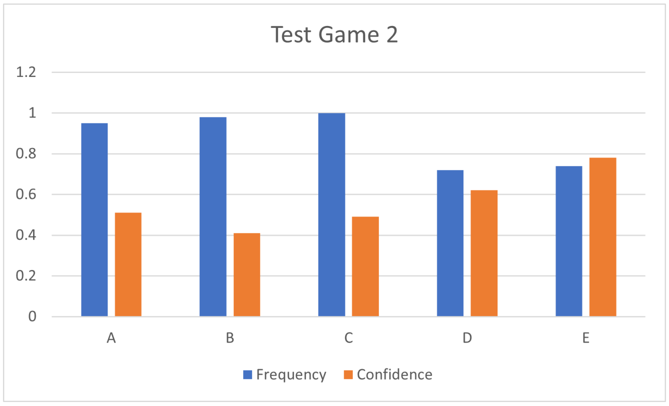

Figure 4: Test Game 2 Results.

In Game 1, the spies were player A and B, but NARS did not think that B was a spy. This was because the two spies played a strategy where they turned on each other, pretending to be enemies. This fooled NARS into believing player B and blaming player D instead. In Game 2, the spies were A and C. NARS figured out that D and E were less likely to be spies, but could not differentiate between player A, B, and C. Overall, I think if NARS had more data to process (more experience), and more understanding of the game (more strategy), then it would be able to do better. More experimentation must be done to determine which data is most effective in helping NARS to be successful in social deduction games.

Conclusions

In many real world situations, agents are faced with changing environments and insufficient knowledge, yet they still must make reasonable decisions at that point in time, based on the current state of the environment and the knowledge that they have been provided with, as well as relying on past experiences. These types of situations also occur in social deduction games, where players attempt to uncover each other's hidden role or team allegiance, using logic and deductive reasoning, or lying and bluffing to deceive other players. In this project, a non-axiomatic reasoning system, NARS, was taught the rules to a social deduction reasoning game called "The Resistance", and then observed games to see how well it could determine the players' hidden roles. This experiment saw some success and points to NARS (with more work) being able to navigate these type of environments and situations, but there is still work to be done in making the results more consistent.

Future Work

More research can still be done in the area of AI in social deduction games, and NARS has shown promise in being able to understand and potentially play these social deduction games. For the current version of this research project, more training and testing data could be collected, and more experiments could be done to determine the best way to teach the concepts, terms, and rules of the game. Some questions to consider are how much time should NARS be given to process new information, how often should it be reminded of the rules and of past information, what strategies work best, and is there a better task/concept framework that would allow NARS to better understand the game.

Furthermore, this project could be extended in a variety of ways. One obvious extension to this research is to enable NARS to play the game in real time, making decision about what to do from both the perspective of a spy or a resistance member. Furthermore, as the research in NLP continues, perhaps NARS could process more complicated arguments that players are making about each other in the game. There could me more ways to classify a player's deliberation than just vouching for or accusing another player. NARS could also read into the sentiment of player's based on their word choices or even choosing to remain silent, which can often be a strategy to deflect suspicion from a player. Lastly, more complicated versions of the game exist, which allow players to have more options in the game, and will reveal certain information to certain players. This further complicates the strategy in the game. Overall, this project shows the potential that NARS has in social deduction games and other situations where there are agents with conflicting information, goals, and allegiances; and one must make decisions on who is being honest and who is being deceitful.

References

[1] Eger, M., & Martens, C. (2018). *View of Keeping the Story Straight: A Comparison of Commitment Strategies for a Social Deduction Game.* AAAI Publications. Retrieved February 28, 2022, from https://ojs.aaai.org/index.php/AIIDE/article/view/13015/12863

[2] Kopparapu et al. (2022, January 5). *Hidden Agenda: a Social Deduction Game with Diverse Learned Equilibria.* Arxiv.Org. Retrieved February 28, 2022, from https://arxiv.org/pdf/2201.01816.pdf

[3] *Play The Resistance Avalon Online!*(n.d.). Proavalon. Retrieved February 28, 2022, from https://www.proavalon.com/rules

[4] Wang, P. (n.d.). *A Logical Model of Intelligence — An introduction to NARS.* Cis.Temple.Edu. Retrieved March 19, 2022, from https://cis.temple.edu/ pwang/NARS-Intro.html

[5] *GitHub - opennars/opennars: OpenNARS for Research 3.0+.* GitHub. Retrieved April 16, 2022, from https://github.com/opennars/opennars

Appendix

Appendix A: Sample Game in English

```
The spies are: C, E, (only included in training games)
The resistance are: A, B, D, (only included in training games)
Number of players: 5
Number of spies: 2

Round: 1
Failed Team Votes: 0
Leader: A
Mission Score: Succeeded = 0, Failed = 0

Deliberation phase begins.
A is done deliberating.
B is done deliberating.
C is done deliberating.
D is done deliberating.
E is done deliberating.
Deliberation phase ends.
A proposes the team: [A, C]
Deliberation phase begins.
A is done deliberating.
B is done deliberating.
C is done deliberating.
D is done deliberating.
E is done deliberating.
Deliberation phase ends.
A votes Y on proposed team [A, C]
B votes N on proposed team [A, C]
C votes Y on proposed team [A, C]
D votes Y on proposed team [A, C]
E votes N on proposed team [A, C]
Vote passes.
The mission passes with 0 failure votes.
Leader passes to B

Round: 2
Failed Team Votes: 0
Leader: B
Mission Score: Succeeded = 1, Failed = 0

Deliberation phase begins.
A is done deliberating.
B is done deliberating.
C is done deliberating.
D is done deliberating.
E is done deliberating.
Deliberation phase ends.
B proposes the team: [A, B, C]
Deliberation phase begins.
A is done deliberating.
B is done deliberating.
C is done deliberating.
D is done deliberating.
E is done deliberating.
Deliberation phase ends.
```

```
A votes Y on proposed team [A, B, C]
B votes Y on proposed team [A, B, C]
C votes Y on proposed team [A, B, C]
D votes N on proposed team [A, B, C]
E votes N on proposed team [A, B, C]
Vote passes.
The mission fails with 1 failure votes.
Leader passes to C

Round: 3
Failed Team Votes: 0
Leader: C
Mission Score: Succeeded = 1, Failed = 1

Deliberation phase begins.
A vouches for C.
B accuses A of being a spy.
C vouches for A.
D vouches for D.
E vouches for E.
A accuses B of being a spy.
B accuses C of being a spy.
C accuses B of being a spy.
D is done deliberating.
E is done deliberating.
A is done deliberating.
B is done deliberating.
C is done deliberating.
Deliberation phase ends.
C proposes the team: [A, C]
Deliberation phase begins.
A is done deliberating.
B is done deliberating.
C is done deliberating.
D is done deliberating.
E is done deliberating.
Deliberation phase ends.
A votes Y on proposed team [A, C]
B votes N on proposed team [A, C]
C votes Y on proposed team [A, C]
D votes Y on proposed team [A, C]
E votes Y on proposed team [A, C]
Vote passes.
The mission fails with 1 failure votes.
Leader passes to D

Round: 4
Failed Team Votes: 0
Leader: D
Mission Score: Succeeded = 1, Failed = 2

Deliberation phase begins.
A accuses C of being a spy.
B is done deliberating.
C accuses A of being a spy.
D is done deliberating.
E is done deliberating.
A is done deliberating.
```

```
C is done deliberating.
Deliberation phase ends.
D proposes the team: [D, B, C]
Deliberation phase begins.
A accuses C of being a spy.
B is done deliberating.
C accuses A of being a spy.
D is done deliberating.
E vouches for D.
A is done deliberating.
C is done deliberating.
E vouches for E.
E is done deliberating.
Deliberation phase ends.
A votes N on proposed team [D, B, C]
B votes N on proposed team [D, B, C]
C votes Y on proposed team [D, B, C]
D votes Y on proposed team [D, B, C]
E votes Y on proposed team [D, B, C]
Vote passes.
The mission fails with 1 failure votes.
Leader passes to E
3 failed missions, the spies win.
```

Appendix B: Sample Game in Narsese

```
<{game1} --> game>. %1.00;0.99%
<{C1} --> spy>. %1.00;0.99%
<{E1} --> spy>. %1.00;0.99%
<{A1} --> spy>. %0.00;0.99%
<{B1} --> spy>. %0.00;0.99%
<{D1} --> spy>. %0.00;0.99%
<{A1} --> player>. %1.00;0.99%
<{B1} --> player>. %1.00;0.99%
<{C1} --> player>. %1.00;0.99%
<{D1} --> player>. %1.00;0.99%
<{E1} --> player>. %1.00;0.99%
<{round1g1} --> round>. %1.00;0.9%
<(*,{game1},{round1g1}) --> in>. :|: %1.00;0.9%
<{round1g1} --> [<failedVotes --> [0]>]>. :|: %1.00;0.9%
<{A1} --> leader>. :|: %1.00;0.9%
<{game1} --> [<resistanceScore --> [0]>]>. :|: %1.00;0.9%
<{game1} --> [<spyScore --> [0]>]>. :|: %1.00;0.9%
<{game1} --> deliberating>. :|: %1.00;0.9%
<{game1} --> deliberating>. :|: %0.00;0.9%
<{team1g1} --> team>. %1.00;0.99%
<{team1g1} --> [<size --> [2]>]>. %1.00;0.9%
<(*,{A1},{team1g1}) --> proposes>. %1.00;0.9%
<(*,{team1g1},{round1g1}) --> proposedOn>. %1.00;0.9%
<(*,{A1},{team1g1}) --> memberOf>. %1.00;0.99%
<(*,{C1},{team1g1}) --> memberOf>. %1.00;0.99%
<{game1} --> deliberating>. :|: %1.00;0.9%
<{game1} --> deliberating>. :|: %0.00;0.9%
<(*,{A1},{team2g1}) --> supports>. :|: %0.9;0.8%
(--,<(*,{B1},{team2g1}) --> supports>). :|: %0.9;0.8%
<(*,{C1},{team2g1}) --> supports>. :|: %0.9;0.8%
<(*,{D1},{team2g1}) --> supports>. :|: %0.9;0.8%
(--,<(*,{E1},{team2g1}) --> supports>). :|: %0.9;0.8%
<{team2g1} --> [accepted]>. %1.00;0.99%
<{team2g1} --> [success]>. %1.00;0.99%
```

```
<{team2g1} --> [noFails]>. %1.00;0.99%
(--,<{A1} --> leader>). :|: %1.00;0.9%
<{B1} --> leader>. :|: %1.00;0.9%
<{round2g1} --> round>. %1.00;0.9%
<(*,{game1},{round2g1}) --> in>. :|: %1.00;0.9%
(--,<(*,{game1},{round1g1}) --> in>). :|: %1.00;0.9%
<{round2g1} --> [<failedVotes --> [0]>]>. :|: %1.00;0.9%
<{B1} --> leader>. :|: %1.00;0.9%
<{game1} --> [<resistanceScore --> [1]>]>. :|: %1.00;0.9%
<{game1} --> [<spyScore --> [0]>]>. :|: %1.00;0.9%
<{game1} --> deliberating>. :|: %1.00;0.9%
<{game1} --> deliberating>. :|: %0.00;0.9%
<{team2g1} --> team>. %1.00;0.99%
<{team2g1} --> [<size --> [3]>]>. %1.00;0.9%
<(*,{B1},{team2g1}) --> proposes>. %1.00;0.9%
<(*,{team2g1},{round2g1}) --> proposedOn>. %1.00;0.9%
<(*,{A1},{team2g1}) --> memberOf>. %1.00;0.99%
<(*,{B1},{team2g1}) --> memberOf>. %1.00;0.99%
<(*,{C1},{team2g1}) --> memberOf>. %1.00;0.99%
<{game1} --> deliberating>. :|: %1.00;0.9%
<{game1} --> deliberating>. :|: %0.00;0.9%
<(*,{A1},{team3g1}) --> supports>. :|: %0.9;0.8%
<(*,{B1},{team3g1}) --> supports>. :|: %0.9;0.8%
<(*,{C1},{team3g1}) --> supports>. :|: %0.9;0.8%
(--,<(*,{D1},{team3g1}) --> supports>). :|: %0.9;0.8%
(--,<(*,{E1},{team3g1}) --> supports>). :|: %0.9;0.8%
<{team3g1} --> [accepted]>. %1.00;0.99%
(--,<{team3g1} --> [success]>). %1.00;0.99%
<{team3g1} --> [oneFail]>. %1.00;0.99%
(--,<{B1} --> leader>). :|: %1.00;0.9%
<{C1} --> leader>. :|: %1.00;0.9%
<{round3g1} --> round>. %1.00;0.9%
<(*,{game1},{round3g1}) --> in>. :|: %1.00;0.9%
(--,<(*,{game1},{round2g1}) --> in>). :|: %1.00;0.9%
<{round3g1} --> [<failedVotes --> [0]>]>. :|: %1.00;0.9%
<{C1} --> leader>. :|: %1.00;0.9%
<{game1} --> [<resistanceScore --> [1]>]>. :|: %1.00;0.9%
<{game1} --> [<spyScore --> [1]>]>. :|: %1.00;0.9%
<{game1} --> deliberating>. :|: %1.00;0.9%
<(*, {A1}, (--, <{C1} --> spy>)) --> believes>. :|: %0.9;0.9%
<(*, {B1}, <{A1} --> spy>) --> believes>. :|: %0.9;0.9%
<(*, {C1}, (--, <{A1} --> spy>)) --> believes>. :|: %0.9;0.9%
<(*, {D1}, (--, <{D1} --> spy>)) --> believes>. :|: %0.9;0.9%
<(*, {E1}, (--, <{E1} --> spy>)) --> believes>. :|: %0.9;0.9%
<(*, {A1}, <{B1} --> spy>) --> believes>. :|: %0.9;0.9%
<(*, {B1}, <{C1} --> spy>) --> believes>. :|: %0.9;0.9%
<(*, {C1}, <{B1} --> spy>) --> believes>. :|: %0.9;0.9%
<{game1} --> deliberating>. :|: %0.00;0.9%
<{team3g1} --> team>. %1.00;0.99%
<{team3g1} --> [<size --> [2]>]>. %1.00;0.9%
<(*,{C1},{team3g1}) --> proposes>. %1.00;0.9%
<(*,{team3g1},{round3g1}) --> proposedOn>. %1.00;0.9%
<(*,{A1},{team3g1}) --> memberOf>. %1.00;0.99%
<(*,{C1},{team3g1}) --> memberOf>. %1.00;0.99%
<{game1} --> deliberating>. :|: %1.00;0.9%
<{game1} --> deliberating>. :|: %0.00;0.9%
<(*,{A1},{team4g1}) --> supports>. :|: %0.9;0.8%
(--,<(*,{B1},{team4g1}) --> supports>). :|: %0.9;0.8%
<(*,{C1},{team4g1}) --> supports>. :|: %0.9;0.8%
<(*,{D1},{team4g1}) --> supports>. :|: %0.9;0.8%
<(*,{E1},{team4g1}) --> supports>. :|: %0.9;0.8%
<{team4g1} --> [accepted]>. %1.00;0.99%
(--,<{team4g1} --> [success]>). %1.00;0.99%
```

```
<{team4g1} --> [oneFail]>. %1.00;0.99%
(--,<{C1} --> leader>). :|: %1.00;0.9%
<{D1} --> leader>. :|: %1.00;0.9%
<{round4g1} --> round>. %1.00;0.9%
<(*,{game1},{round4g1}) --> in>. :|: %1.00;0.9%
(--,<(*,{game1},{round3g1}) --> in>). :|: %1.00;0.9%
<{round4g1} --> [<failedVotes --> [0]>]>. :|: %1.00;0.9%
<{D1} --> leader>. :|: %1.00;0.9%
<{game1} --> [<resistanceScore --> [1]>]>. :|: %1.00;0.9%
<{game1} --> [<spyScore --> [2]>]>. :|: %1.00;0.9%
<{game1} --> deliberating>. :|: %1.00;0.9%
<(*, {A1}, <{C1} --> spy>) --> believes>. :|: %0.9;0.9%
<(*, {C1}, <{A1} --> spy>) --> believes>. :|: %0.9;0.9%
<{game1} --> deliberating>. :|: %0.00;0.9%
<{team4g1} --> team>. %1.00;0.99%
<{team4g1} --> [<size --> [3]>]>. %1.00;0.9%
<(*,{D1},{team4g1}) --> proposes>. %1.00;0.9%
<(*,{team4g1},{round4g1}) --> proposedOn>. %1.00;0.9%
<(*,{D1},{team4g1}) --> memberOf>. %1.00;0.99%
<(*,{B1},{team4g1}) --> memberOf>. %1.00;0.99%
<(*,{C1},{team4g1}) --> memberOf>. %1.00;0.99%
<{game1} --> deliberating>. :|: %1.00;0.9%
<(*, {A1}, <{C1} --> spy>) --> believes>. :|: %0.9;0.9%
<(*, {C1}, <{A1} --> spy>) --> believes>. :|: %0.9;0.9%
<(*, {E1}, (--, <{D1} --> spy>)) --> believes>. :|: %0.9;0.9%
<(*, {E1}, (--, <{E1} --> spy>)) --> believes>. :|: %0.9;0.9%
<{game1} --> deliberating>. :|: %0.00;0.9%
(--,<(*,{A1},{team5g1}) --> supports>). :|: %0.9;0.8%
(--,<(*,{B1},{team5g1}) --> supports>). :|: %0.9;0.8%
<(*,{C1},{team5g1}) --> supports>. :|: %0.9;0.8%
<(*,{D1},{team5g1}) --> supports>. :|: %0.9;0.8%
<(*,{E1},{team5g1}) --> supports>. :|: %0.9;0.8%
<{team5g1} --> [accepted]>. %1.00;0.99%
(--,<{team5g1} --> [success]>). %1.00;0.99%
<{team5g1} --> [oneFail]>. %1.00;0.99%
(--,<{D1} --> leader>). :|: %1.00;0.9%
<{E1} --> leader>. :|: %1.00;0.9%
<{game1} --> [spyVictory]>. :|: %1.00;0.99%
```

Appendix C: Rules and Questions in Narsese

```
<(&&,<$1 --> player>,<#2 --> team>,<#2 --> [success]>,<(*, $1, #2) --> memberOf>)
==> (--,<$1 --> spy>)>. %0.8000002;0.9%

<(&&,<$1 --> player>,<#2 --> team>,(--,<#2 --> [success]>),<(*, $1, #2) --> memberOf>)
==> <$1 --> spy>>. %0.8000002;0.9%

<(&&,<$1 --> player>,<#2 --> team>,<#2 --> [success]>,<(*,$1, #2) --> supports>)
==> (--,<$1 --> spy>)>. %0.60;0.70%

<(&&,<$1 --> player>,<#2 --> team>,<#2 --> [success]>,(--,<(*,$1, #2) --> supports>))
==> <$1 --> spy>>. %0.60;0.70%

<(&&,<$1 --> player>,<#2 --> team>,(--,<#2 --> [success]>),<(*,$1, #2) --> supports>)
==> <$1 --> spy>>. %0.70;0.70%

<(&&,<$1 --> player>,<#2 --> team>,(--,<#2 --> [success]>),(--,<(*,$1, #2) --> supports>))
==> (--,<$1 --> spy>)>. %0.70;0.70%

<(&&,<$1 --> player>,<#2 --> team>,<#2 --> [success]>,<(*,$1, #2) --> proposes>)
==> (--,<$1 --> spy>)>. %0.60;0.70%

<(&&,<$1 --> player>,<#2 --> team>,(--,<#2 --> [success]>),<(*,$1, #2) --> proposes>)
```

```
==> (--,<$1 --> spy>)>. %0.60;0.70%

<(&&,<$3 --> player>,<#4 --> player>,<$3 --> spy>,<(*, $3, <#4 --> spy>) --> believes>)
==> (--,<#4 --> spy>)>. %0.60;0.60%

<(&&,<$3 --> player>,<#4 --> player>,<$3 --> spy>,<(*, $3, (--, <#4 --> spy>)) --> believes>)
==> <#4 --> spy>>. %0.60;0.6%

<{A9} --> spy>?
''outputMustContain('<{A9} --> spy>?' )
<{B9} --> spy>?
''outputMustContain('<{B9} --> spy>?' )
<{C9} --> spy>?
''outputMustContain('<{C9} --> spy>?' )
<{D9} --> spy>?
''outputMustContain('<{D9} --> spy>?' )
<{E9} --> spy>?
''outputMustContain('<{E9} --> spy>?' )
```