Universiteit Utrecht

Artificial Intelligence
Graduate School of Natural Sciences
Intelligent Systems

A stability-based approach to inquiry dialogues in agent argumentation

L.W. Leijten

First examiner
Prof. Dr. F.J. Bex
Department of Information and Computing Sciences
Utrecht University

Second examiner
Dr. G.A.W. Vreeswijk
Department of Information and Computing Sciences
Utrecht University

August 30, 2019

Abstract

In inquiry dialogues, two or more agents work together to prove or disprove a proposition. The de facto standard for this type of dialogues are exhaustive systems, in which agents make every move that could impact the outcome of the dialogue before terminating the dialogue. This behaviour results in positive properties such as soundness and completeness, but as a downside generates long dialogues including redundant moves. New research proposes a stability-based querying system, in which agents stop making queries when it is certain that the outcome of the process will not change anymore. In this thesis, this querying system is extended into a set of three, first of their kind, stability-based inquiry dialogue systems with increasing levels of expressiveness. First it is explored how performance of inquiry systems can be defined and compared. Next, the performance of these stability-based systems is compared to each other and to exhaustive systems. Experiments are performed on multiple rule sets in which the agents try to minimize either the dialogue length or the amount of observations shared. Training and test sets for these experiments are initiated using three different instantiation functions to model different cases that could occur in real life situations. It is shown that stability-based systems in general outperform the exhaustive systems and that, depending on the performance measure and instantiation function used, the additional expressiveness results in increased performance as well. The downside of the additional expressiveness is an increase in complexity, resulting in the more expressive systems not being able to perform well on large problems.

Contents

Chapter 1

Introduction

This thesis aims to make a contribution to the field of agent argumentation dialogues. In argumentation dialogues, as explained by Amgoud et al. [2], two or more agents argue with each other and try to reach their own goals, or cooperate towards a shared goal. What is important in these dialogues are the underlying arguments. The arguments brought forth by the agents can be used in real life situations such as supporting medical or juridical decisions. It is thus important to make the arguments of the agents transparent so that the system can explain itself and be checked by human experts as well.

Argumentation dialogues follow a protocol, which determines what moves can be made and when they can be made. The literature generally considers six different types of dialogue: persuasion, inquiry, negotiation, information-seeking, deliberation and eristic dialogues. These types were defined for the first time by Walton and Krabbe [24]. The three types that will be most relevant to this thesis are those of persuasion dialogues, inquiry dialogues and information-seeking dialogues.

This research will be performed in support of a collaboration done between the University Utrecht and the Dutch national police. The goal of this collaboration is to have software agents handle online fraud reports through an online chat interface with complainants, increasing the amount and quality of complaints that are handled. As stated by Schraagen et al. [20], online fraud is a high volume crime with a relatively simple legal foundation. This makes it the ideal type of crime to develop such systems for.

A lot of the research done so far on the topic of argumentation dialogues is on the field of persuasion dialogues, where each agent tries to convince the other agents of their own beliefs. Examples of research done on strategies, which dictate what actions agents should perform, for these persuasion dialogues are: Hadjinikolis et al. [12], Hadoux et al. [13], Rienstra et al. [19], Alahmari et al. [1] and Black et al. [4]. For the national police however, inquiry dialogues are more interesting since it resembles their use case more closely, as argued for by Kumeling [14] and Testerink, Odekerken and Bex [23]. In an inquiry dialogue two or more agents work together to find grounds for a shared opinion on a

certain piece of information, called the dialogue topic. In the case of the police project this is to find a legal base for fraud having taken place. Related to this are two variants of the inquiry dialogue, namely: argument inquiry dialogues and warrant inquiry dialogues, which are defined by Black and Hunter [5]. The goal of argument inquiry is to find out if an argument for a given proposition exists. The goal of warrant inquiry is to find an acceptable argument for a given proposition. An acceptable argument is an argument that can still hold when taking into account possible counter-arguments.

In an argumentation dialogue, the participating agents follow a strategy. Outcome and performance of the dialogue is dependent on the strategies used. There exist, for example, exhaustive strategies that play every argument that can be made, leading to favourable properties that will be explained later on in this thesis. It is important to define strategies that allow agents to debate effectively, but that also do not ask too many questions in real life situations. If the system asks too many redundant questions the end user conversing with it might lose focus and interest. The argumentation, however, also has to be precise and thorough because often it is used in domains that are critical in regards to safety. Depending on the domain the focus could lie more on one of the aspects than on the other, but in general it is important to come up with methods that respect both aspects of this dilemma. Kumeling [14], for example, tries to decrease the amount of moves made by exhaustive strategies whilst trying to still adhere the properties of soundness and completeness. These considerations can be quantified using performance measures. These measures can then be used to determine how well a system performs in the dialogues it generates. Examples of such performance measures are: average dialogue length, computational complexity of dialogues, explainability of the generated dialogues and the soundness and completeness of the dialogue system. In this context, explainability means the capacity of the system to explain the choices that it made and the outcomes it generated.

An interesting idea used throughout agent argumentation research, mostly on persuasion dialogues, is that of opponent models. The dialogue agent creates an internal model of the opponent he is debating and uses this model to determine what moves would be the most effective. This model can consist of different beliefs about the opponent such as rules about how facts known by the opponent relate to facts that he then might know as well. Research on this topic has been done by Rienstra et al. [19] and Hadjinikolis et al. [12] amongst others. A different approach, as researched by Hadoux et al. [13] is to have the opponent behave stochastically and then model them using Markov Decision Processes.

Recently, research on the topic of inquiry dialogues has been done by Testerink, Odekerken and Bex [23]. They propose a querying system for inquiry in which an agent uses the concept of stable states, states after which the outcome of the process cannot change, to minimize the amount of queries the agent has to perform before reaching their verdict. Their system is not yet a fully fledged dialogue system, but it has the potential to lead to shorter and more explainable dialogues for the end user if it were to be extended to one. When converting this

system into a dialogue game, it must be determined what kind of knowledge and actions the participating agents should have at their disposal. This is defined in so called opponent and speaker models. The speaker in this scenario is the agent asking the questions and the opponent is the agent answering these questions. An example of a simple speaker and opponent model would be that the speaker could only perform simple queries and the opponent can only reply with yes or no. More elaborate models could have the opponent have internal knowledge of rules and arguments and be able to additionally reply with arguments to queries performed by the speaker.

Inquiry dialogues seem like the most fitting dialogue type to model police intake conversations as. Current research done on inquiry dialogues is not as elaborate as that done on other types of dialogues. Chapter 2 will give more insight into what research has been done on different aspects of agent argumentation and primarily on inquiry dialogues. With this background, lacking elements of the current standard can be identified and solutions and new additions can be proposed. Section 1.1 shows what research questions will be answered in this thesis.

1.1 Research questions

- **Main research question:** How does the performance of inquiry dialogue systems change when using different opponent and speaker models?

  Sub questions:

  - What are the best performance measures for inquiry type dialogues and more specifically dialogues that model a police intake conversation?
  - How does the set of available locutions for both agents affect the performance of inquiry dialogues?
  - How does the rule set used in inquiry dialogues affect the performance of inquiry dialogues?
  - How do stability-based inquiry systems perform compared to exhaustive systems?

1.2 Method

In this thesis we will implement a dialogue game based on the paper by Testerink, Odekerken and Bex [23] that will help with answering the above research questions. Their definitions will lay the foundations for a dialogue system, which will form the foundation of our research and on which further improvements can be researched and implemented. This dialogue system will have to be formalised so that it can later on be turned into an implementation. When the dialogue system is in place, multiple extensions upon it will be formalised and implemented. This implementation will be programmed using Python. Using the

found performance measures we are then able to compare the performance of the different extensions of the system when using different rule sets.

For these experiments, the strategy will be a stable factor. The agent will always make use of a stability based-strategy, which stops when a stable state has been reached. Depending on the models of both the speaker and the opponent, the exact implementation of the strategy might differ. The core, however, always remains the same. Using the above defined research questions, multiple versions of the system will have to be implemented and compared. First the most suitable measures of performance will have to be found. This will be done by consulting existing literature on the topic of inquiry dialogues. Next, different variations of opponent and speaker models will have to be defined, implemented and then tested in dialogue simulations. These opponent and speaker models encapsulate what knowledge the agents have at their disposal and the locutions they are allowed to use. In the base system the inquiring agent can only query non-defeasible literals and the replying agent can then only confirm or deny that query. An extension of this system would allow the inquiring agent to also query defeasible literals and the responding agent to reply to such queries. A final extension would allow the inquiring agent to ask for arguments that support earlier given responses of the responding agent. If needed, an accept locution could be implemented as well, which allows the inquiring agent to accept the different responses being given by the responding agent. This was not deemed necessary within the scope of this thesis. Another change that could be implemented and experimented with is the knowledge bases of both agents. In our system the inquiring agent will have knowledge of all the rules and the responding agent will have knowledge of all observable literals as well as the set of rules, similar to how the knowledge would be divided in a real fraud intake conversation. In possible extensions, which lie outside the scope of this thesis, both agents should be able to have a mix of rules and literals in their knowledge base.

The remainder of this thesis is structured as follows: chapter 2 will give an overview of literature on the topic of agent argumentation dialogues. In chapter 3 we will look at what kind of performance measures are used throughout literature and which ones are relevant for our research. In chapter 4 we will formally define the dialogue system and its extensions that are used to answer our research questions. Chapter 5 will describe the experiments that are performed to answer our research questions. Chapter 6 will discuss the findings that we have done in this thesis. Chapter 7 will conclude this thesis.

Chapter 2

Literature

To get an idea of what the current standings of agent argumentation are, a literature study is performed. First we will take a short look at the AI for policing project of the Utrecht University, which this thesis will support. Next we will look at dialogue systems and primarily dialogue systems for inquiry dialogues and information-seeking dialogues. These types of dialogues are interesting because they are the most fitting to model police intake conversations with. We will look at what these systems are and how they compare to each other. By doing this we can identify where further research could be applied in the field of inquiry dialogues. Finally we will look at other approaches of research that have elements that might be useful for inquiry and information seeking dialogues, for example, strategies used in persuasion type dialogues.

2.1 AI for policing

AI for policing is a collaboration between the Dutch national police and researchers of Utrecht University. This collaboration has resulted in research being performed on agent argumentation systems that could aid the police in their daily work. This thesis will also be done in support of this project.

Bex et al. [3] describe a system being developed by Utrecht University in combination with the national police. The system uses argumentation to build cases for online fraud complaints. This system is also elaborated on by Schraagen et al. [20]. The system uses techniques from natural language processing, is able to produce arguments and can efficiently gain new knowledge by executing dialogues with complainants. Testerink and Bex [21] describe a formal framework that can be used for communication protocols in open multi-agent systems. Key here is that the protocol should work in peer-to-peer situations. Peer-to-peer protocols are important for real life applications. In these applications the dialogues are parallel, do not have complete information and are peer-to-peer. In such situations, it is undesirable to have agents representing the same organization contradicting each other. Such protocols are especially

useful for the argumentation project done by the police and Utrecht University. This protocol makes use of templates. Templates are structures that determine when agents can send and receive locutions and how these locutions update the dialogue graph. An example of a template is how you can use a why question as a response to a claim made by the other agent. Templates are the main building blocks of this protocol. Agents can only send messages if there is a template that allows the sending of that message. Moves that the agents can make in this system are claim, why and support. Testerink and Bex [22] also give a framework for programming argumentation dialogues. Main contributions of this paper include the ability to develop protocols, the agent modules that interpret the protocol, an example multi-agent system and a visualization of the view of an agent on the dialogue, which can be inspected on the internet.

2.2 Dialogue systems

Dialogue systems are systems that define how agents should perform coherent dialogues. In a dialogue two or more agents argue over a topic. These dialogues can be of different types such as inquiry and persuasion. Dialogue systems are used in support of medical decision making, juridical decision making, customer support and more. These systems are becoming more advanced over time, with more complex agents taking part in them. This thesis will propose a dialogue system that performs inquiry dialogues. To get an idea of what the current state of dialogue systems for inquiry is, a comparison is performed. This comparison will be made on common elements of dialogue systems as defined throughout literature. Common elements of dialogue systems are defined by McBurney and Parsons [15] and Prakken [17]. The following list of common elements is based on their definitions:

- Logic: a logic that describes a topic language. This topic language is used by the agents to reason and argue about the dialogue topic.
- Topic: an element of the topic language. The topic is the piece of information around which a dialogue revolves.
- Argumentation system: describes what arguments look like and how they relate to one another.
- Knowledge bases: describes what types of knowledge the agents have at their disposal and how it is divided between the agents.
- Locutions: a communication language containing allowed locutions allowed and rules how they can interact with each other.
- Moves: definitions for moves that the players can make using locutions.
- Commitments: rules on when agents make commitments and alter their commitment stores and how to deal with conflicting commitments.

- Dialogues: definitions for what a dialogue looks like and when a dialogue is valid.
- Players: the amount of players supported in the dialogue and the roles they can take.
- Protocol: determines what moves the agents are allowed to make during the dialogue. The protocol can also determine when the dialogue is over and whose turn it is at what point in the dialogue.
- Strategy: dictates, given the legal moves returned by the protocol, what the exact move is that the agent should make.

2.2.1 Systems for inquiry

A modest amount of research has been done on the topic of inquiry dialogues. In inquiry dialogues agents cooperate to prove a statement. We will give an overview and descriptions of six different approaches of inquiry dialogue systems. These approaches have a lot of common elements, but also differ enough from each other to be interesting for a comparison. By comparing these approaches we can identify the current state of research and elements that are currently missing or underrepresented.

2.2.1.1 Black and Hunter

Black and Hunter [5] give a framework for so called warrant inquiry and argument inquiry dialogues, both of which are sub-types of inquiry dialogues. The goal of argument inquiry is to find out if an argument for a given proposition exists. The goal of warrant inquiry is to find an acceptable argument for a given proposition. This means that the argument should still hold when considering possible counter-arguments. The system contains protocols for both warrant inquiry and argument inquiry, which tell what the legal moves are during the dialogues. Additionally, they propose an exhaustive strategy to solve these dialogues, which is both sound and complete. This means that all the utterances made in the dialogue follow from the agents' knowledge bases, and everything that can follow from the agents' knowledge bases is uttered in the dialogue. An exhaustive strategy according to Black and Hunter is a strategy that makes all moves that might have an impact on the outcome of the dialogue. Exhaustive strategies such as the one described by Black and Hunter can be seen as the current de facto standard used in inquiry dialogues.

Black and Hunter use an argumentation system, which is based on DeLP [11]. They make use of a combination of defeasible rules and defeasible facts. The rules and facts being defeasible means that they have preference levels and can be attacked by other rules and arguments. The topic of an argument inquiry dialogue is a defeasible rule with as its consequent the literal that the inquiry tries to find an argument for. The topic of a warrant inquiry dialogue is a defeasible fact for which the agents try to find an acceptable argument. The

system itself does not assume a specific split of the rules and facts between the agents. The agents' belief bases can possibly be inconsistent, but this poses no problem because of preference levels. Arguments are tuples of the form $\langle \Phi, \phi \rangle$ where Φ is the support of the argument and ϕ is the conclusion of the argument.

Example 2.2.1 *An example of an argument in this system is as follows:* $A =$ *(*$\{$*(a, 1), (b, 1), (*$a \wedge b \Rightarrow c$*, 1)* $\}$*, c), where 1 is the value of the preference of each premise.*

Arguments can attack each other on their subarguments when their conclusions contradict each other. The winner of such a conflict is decided by the preference levels of the arguments. When the preference level of the attacking argument is more preferable than that of the attacked argument, the attacking argument is known as a proper defeater for the attacked argument. When the preference levels are equal, both arguments are blocking defeaters for each other.

Example 2.2.2 *Consider the following two arguments:* $a1 = \langle \{(a, 1)\}, a \rangle$ *and* $a2 = \langle \{(\neg a, 1)\}, \neg a \rangle$*. Argument a1 and a2 attack each other and argument a1 is a blocking defeater for argument a2, and argument a2 is a blocking defeater for argument a1.*

Example 2.2.3 *Consider the following two arguments:* $a1 = \langle \{(a, 2)\}, a \rangle$ *and* $a2 = \langle \{(\neg a, 1)\}, \neg a \rangle$*. Argument a1 and a2 attack each other and argument a2 is a proper defeater for argument a1.*

The status of an arguments indicates the current standing of the argument in the argumentation system in relation to other arguments. This status is based on attacks and defeat and can be used to determine the outcome of a dialogue. Generally the status of an argument can be one of the following three: winning, losing or undecided but variations of these are used throughout literature. As the status of arguments, Black and Hunter use defeated and undefeated. They use this status to determine the outcome of warrant inquiry dialogues. This will be elaborated on in detail later on in this thesis.

The agents share a so called query store for each argument inquiry dialogue. This query store contains the set of literals that could help the agents construct an argument supporting the topic of this argument inquiry dialogue. At the start of an argument inquiry dialogue, this query store will be initiated with as its contents the premises and the consequent of the topic.

Example 2.2.4 *Consider an argument inquiry being opened for the topic* $a \wedge b \rightarrow c$*. This would result in a query store being created for this dialogue, which would look as follows:* $QS = \{a, b, c\}$

Each agent also has a personal commitment store for each top-level dialogue. This commitment store tracks the assertions made by its owner. When agents infer arguments they can not only use their own beliefs, but also the content of the other agents' commitment stores. These commitment stores can thus be seen as public knowledge.

Agents in Black and Hunter's system can make open moves, close moves and assert moves. The open move comes in two forms: an agent can open a new warrant inquiry dialogue or a new argument inquiry dialogue. These new dialogues will be nested as sub dialogues in the dialogue in which the move has been made. A close move indicates that the speaking agent wants to close the current dialogue. Assert moves allow the agent to assert new arguments and propositions. These asserted arguments and propositions will be added to the commitment store of the agent making the move. The format of a move is {agent, act, content} where agent is the identifier of the agent making the move, act is the type of move being made and content contains the details of the move.

Example 2.2.5 *Consider an assert move being made by agent A that asserts b based on the premise a. This move would look as follows:* $\langle A, assert, \langle \{(a, 1), (a \rightarrow b, 1)\}, b \rangle \rangle$.

Example 2.2.6 *Consider an open move being made by agent A for a warrant inquiry dialogue with as topic the defeasible fact b. This move would look as follows:* $\langle A, open, dialogue(wi, b) \rangle$.

Black and Hunter give protocols for both argument inquiry and warrant inquiry. The protocols assume two agents taking part in the dialogue, who can make one move each turn and the turns are alternating. The generated dialogue is a sequence of moves with a certain topic. The two protocols are defined below:

- Argument inquiry (AI) protocol:
    1. Close moves for the current dialogue are always added to the set of legal moves.
    2. Assert moves for information from the agent's knowledge base that have not yet been asserted and have its claim in the query store are added to the set of legal moves.
    3. Open moves for new argument inquiry dialogues with a topic that has not been opened before and with a member of the query store as its consequent are added to the set of legal moves.
- Warrant inquiry (WI) protocol:
    1. Close moves for the current dialogue are always added to the set of legal moves.
    2. Assert moves that assert an argument which has not yet been asserted, and either are the first assert for the dialogue topic or change the dialectical tree, are added to the set of legal moves.
    3. Open moves for embedded argument inquiry dialogues with a defeasible rule as the topic which has not been opened before and for which one of the following is true are added to the set of legal moves:

(a) No argument for that topic has been asserted yet and the consequent of the defeasible rule is the topic of the dialogue.

(b) It is possible to defeasibly derive the negation of the consequent from the union of both agents their commitment stores.

Dialogues using these protocols are terminated when both agents make a close move in succession of each other.

The outcome of an argument inquiry dialogue is a set containing all arguments that can be constructed from the union of the commitment stores and whose claims are in the query store.

Example 2.2.7 *The outcome of an argument inquiry dialogue with topic* $b \rightarrow c$ *could be as follows:* $\{\langle\{(a,1),(a \rightarrow b,1),(b \rightarrow c,1)\},c\rangle,\langle\{(b,1),(b \rightarrow c,1)\},c\rangle\}$.

The outcome of a warrant inquiry dialogue is based on the dialectical tree created for the topic argument. Recall that the status of an argument in this system can be either defeated or undefeated. A dialectical tree is a tree consisting of argumentation lines, sequences of arguments such that each argument in the sequence is a proper defeater or a blocking defeater of its predecessor. All leaves of the tree have the status undefeated and other nodes have the status undefeated if all of their children are marked defeated and the status defeated if one of their children is undefeated. The tree is constructed from the union of both agents their commitment stores. The outcome is decided by the status of the root of this dialectical tree. If the status of the root of the generated dialectical tree is undefeated, then the outcome is the argument in the root. If the status of the root of the generated dialectical tree is defeated, then the outcome is null.

Example 2.2.8 *The outcome of a succesfull warrant inquiry dialogue with topic* b *could be as follows:*$\langle\{(a,1),(a \rightarrow b,1)\},b\rangle$

The strategy used by the agents is an exhaustive strategy. Exhaustiveness implies that both agents use all their knowledge they have at their disposal to make moves before they reach a conclusion. This means that the strategy plays all legal moves given by the protocol, plus some additional restrictions. It first defines a subset of the legal assert moves and legal open moves given by the protocol. Agents are not allowed to make up arguments or deceive the other agent. The agents also are not allowed to open an argument inquiry with a topic of which they do not have a belief in their knowledge base. Given these restrictions on legal moves, the exhaustive strategy is as follows:

1. If the subset of legal assert moves is not empty, a move from that set will be played.

2. If the subset of legal assert moves is empty but the subset of legal open moves is not empty, an open move will be played.

3. If both the subset of legal assert moves and the subset of legal open moves are empty a close move will be played.

This strategy is both sound and complete. Soundness and completeness for argument and warrant inquiry dialogues are defined as follows:

- An argument inquiry dialogue is sound if and only if "when the outcome of the dialogue includes an argument, then that same argument can be constructed from the union of the two participating agents' beliefs" (Black & Hunter, 2009, p. 198).
- An argument inquiry dialogue is complete if and only if "if the dialogue terminates at t and it is possible to construct an argument for a literal in the query store from the union of the two participating agents' beliefs, then that argument will be in the outcome of the dialogue at t" (Black & Hunter, 2009, p. 199).
- A warrant inquiry dialogue is sound if and only if, if the outcome of the dialogue is an argument and that argument is the root of a dialectical tree T, which is constructed from the unified belief bases of the participating agents then the status of the root of that tree is undefeated.
- A warrant inquiry dialogue is complete if and only if, if the root argument of the dialogue is at the root of a dialectical tree T, which is constructed from the unified belief bases of the participating agents, then the status of the root node of that tree is undefeated.

An example of a dialogue generated by this system is shown further on in table 2.1.

2.2.1.2 Kumeling

Kumeling [14] proposes improvements to the earlier described exhaustive strategy of Black and Hunter [5]. The original exhaustive strategy contains redundant information and duplicity of arguments and this research tries to find solutions to prevent that from happening.

The research uses the same structured argumentation framework as Black and Hunter [5]. Kumeling, for some of their experiments however, does assume a split of knowledge where one agent has all the rules, and the other agent only has facts. This split is inspired by police intake conversations. Kumeling uses the same set of locutions and moves as Black and Hunter [5], but uses a slightly different notation for moves. The format Kumeling uses is {x, θ, p(c)} where x is the identifier of the agent making the move, θ is the type of the current dialogue and p(c) is a locution.

Example 2.2.9 *Consider an assert move being made by agent A, in an argument inquiry dialogue that asserts b based on the premise a. This move would look as follows:* $\langle A, ai, assert(\langle\{(a,1),(a \Rightarrow b,1)\},b\rangle)\rangle$.

Example 2.2.10 *Consider an open move being made by agent A for a warrant inquiry dialogue with as topic the defeasible fact b. This move would look as follows:* $\langle A, wi, open(wi,b)\rangle$.

t	CS_A	m_t	CS_B	QS_t
1		$\langle A, open, dialogue(wi, b)\rangle$		
2		$\langle B, close, dialogue(wi, b)\rangle$		
3	(a, 4) (a $\rightarrow$ b, 4)	$\langle A, assert, \langle\{(a, 4), (a \rightarrow b, 4)\}, b\rangle\rangle$		
4		$\langle B, assert, \langle\{(d, 3), (d \rightarrow \neg a, 3)\}, \neg a\rangle\rangle$	(d, 3) (d $\rightarrow \neg$ a, 3)	
5	(c, 3) (c $\rightarrow \neg$ b, 3)	$\langle A, assert, \langle\{(c, 3), (c \rightarrow \neg b, 3)\}, \neg b\rangle\rangle$		
6		$\langle B, assert, \langle\{(\neg d, 1)\}, \neg d\rangle\rangle$	($\neg$ d, 1)	
7		$\langle A, open, dialogue(ai, a \rightarrow b)\rangle$		$QS_7 = \{a,b\}$
8		$\langle B, close, dialogue(ai, a \rightarrow b)\rangle$		
9		$\langle A, close, dialogue(ai, a \rightarrow b)\rangle$		
10		$\langle B, open, dialogue(ai, d \rightarrow \neg a)\rangle$		$QS_{10} = \{d, \neg a\}$
11		$\langle A, close, dialogue(ai, d \rightarrow \neg a)\rangle$		
12		$\langle B, close, dialogue(ai, d \rightarrow \neg a)\rangle$		
13		$\langle A, open, dialogue(ai, c \rightarrow \neg b)\rangle$		$QS_{13} = \{c, \neg b\}$
14		$\langle B, close, dialogue(ai, c \rightarrow \neg b)\rangle$		
15		$\langle A, close, dialogue(ai, c \rightarrow \neg b)\rangle$		
16		$\langle B, open, dialogue(ai, e \rightarrow \neg d)\rangle$		$QS_{16} = \{e, \neg d\}$
17	(e, 2)	$\langle A, assert, \langle\{(e, 2)\}, e\rangle\rangle$		
18		$\langle B, assert, \langle\{(e, 2), (e \rightarrow \neg d, 2)\}, \neg d\rangle\rangle$	(e, 2) (e $\rightarrow \neg$ d, 2)	
19		$\langle A, close, dialogue(ai, e \rightarrow \neg d)\rangle$		
20		$\langle B, close, dialogue(ai, e \rightarrow \neg d)\rangle$		
21		$\langle A, close, dialogue(wi, b)\rangle$		
22		$\langle B, assert, \langle\{(\neg e, 1)\}, \neg e\rangle\rangle$	($\neg$ e, 1)	
23		$\langle A, close, dialogue(wi, b)\rangle$		
24		$\langle B, close, dialogue(wi, b)\rangle$		

Table 2.1: A warrant inquiry dialogue with dialogue topic b. Embedded argument inquiry dialogues are opened at lines 7, 10, 13 and 16. The commitment stores increase gradually over time, they start out being empty and the additions can be read in the CS coloms. Whenever an argument inquiry dialogue is opened, an associated query store is initiated, which can be seen in the QS colom.

Kumeling uses the same protocols as Black and Hunter [5], but proposes two improvements on their exhaustive strategy: the limited exhaustive strategy and the smart original strategy. The limited exhaustive strategy has two variations, the limited commitment strategy and the limited dialogue strategy. The changes are implemented by creating different picking functions for assert moves:

- The limited commitment strategy picks the first legal assert move with a conclusion for which no argument is constructable from the union of the commitment stores.
- The limited dialogue strategy picks the first assert move that has a conclusion for which no argument was asserted earlier in the dialogue.
- The smart original strategy, puts extra constraints on assert moves: if the argument made would change the dialectical tree, then they also have to change the status of the root argument in order to be legal.

The possible improvements are measured by the size of the resulting dialogues, size of the dialectical tree and whether or not they still adhere the properties of soundness and completeness. The smart strategy still is sound and complete. The limited strategies do not have both properties. In the case of argument inquiry the limited strategies are sound, but they are not complete. In warrant inquiry it is the other way around: the strategies are not sound, but they are complete. During experimentation some of the strategies resulted in reduced dialogue sizes. There however was no strategy between the three that reduced the size of dialogues, reduced the size of the dialectical trees and still had the properties of soundness and completeness. They concluded that it is difficult to increase the performance of inquiry dialogues, without giving the agent access to more information to base a strategy on.

A dialogue would look the same as the one earlier shown for Black and Hunter [5] in table 2.1, except for minor differences in the notation of moves.

2.2.1.3 Yan et al.

Yan et al. [25] propose a different improvement on the framework of Black and Hunter [5]. Yan et al. take inspiration from Black and Hunter but want to improve upon it since Black and Hunter's approach has stayed away from practical applications. Yan et al. claim that the extension given by their paper results in more simplified implementations with clearer and faster inquiry dialogues.

Yan et al. use their own argumentation system that is based on possibilistic logic rather than defeasible logic like the systems of Black and Hunter [5] and Kumeling [14]. The agents make use of possibilistic beliefs, which can either be state beliefs or domain beliefs. Possibilistic beliefs are tuples of the form (ϕ, b) where ϕ is a rule and p is a lower bound of the belief of that rule ϕ. Rules are of the form of $\alpha_1 \wedge ... \wedge \alpha_n \rightarrow \beta$, where α_1 through α_n are called the premises of the rule and β the conclusion. The premises and the conclusion are all literals. If the rule has zero premises, it is called a fact and is denoted by $\rightarrow \beta$. If ϕ is a

fact the belief is a state belief, otherwise it is a domain belief. Arguments are tuples of the form $\langle \Phi, (l, p) \rangle$ where Φ is a set of beliefs and (l, p) is a possibilistic literal, which is known as the claim of the argument.

Example 2.2.11 *An example of an argument in this system is as follows: A* $= \langle ((\rightarrow a, 1), (\rightarrow c, 1), (a \wedge c \rightarrow f, p2)), (f, p2) \rangle$. *p2 is a possibilistic value that stands for probable.*

Arguments can conflict on their conclusion and their premises. The knowledge could be split in any way between the two agents, but in their example implementation they choose to have one agent only know about facts and the other agent know about both facts and rules.

In addition to the query store and commitment stores used by Black and Hunter [5], Yan et al. introduce two additional data structures which are needed for their approach. They also introduce a small difference in the way the commitment store gets updated. Yan et al. update the agents's commitment stores when *assert wi* moves are being made and when *ai* dialogues close. The possible beliefs queue is a queue used in warrant inquiry dialogues that stores an agent's related beliefs of the current dialogue topic. The related beliefs are state beliefs and domain beliefs from the agent's knowledge base that have the current dialogue topic, or the negation of this topic, as its conclusion. The beliefs added for a topic x thus are those believes that have a conclusion x or $\neg x$. Each agent has their own possible beliefs queue. These beliefs are used when the agents have to make a move.

Example 2.2.12 *Consider a warrant inquiry dialogue being opened for the literal f. The possible beliefs queue of an agent P could then look as follows:* $PBQ_P(f) = \{(a \wedge b \wedge c \rightarrow f, p1), (d \rightarrow \neg f, p1)\}$

The result store is a set of tuples that stores the intermediate results of warrant inquiry dialogues. The purpose of the result store is to prevent having to do repetitive and duplicate work during dialogues.

Example 2.2.13 *A result store storing the results of two warrant inquiry dialogues could look as follows:* $\{\langle a, \langle T, 1 \rangle \rangle, \langle b, \langle U, null \rangle \rangle\}$. *The first element stores the result of a warrant inquiry dialogue with topic a and as a result the value true with a probabilistic value of 1. The second element is the result of a warrant inquiry dialogue with topic b that is not resolved yet.*

Yan et al. use the same set of locutions and moves as Black and Hunter [5], but use a slightly different notation for moves. The format Yan et al. use {agent, move type, dialogue type, topic} where agent is the identifier of the agent making the move, move type is the type of move being made, dialogue type is the type of the current dialogue and topic is the content of the move.

Example 2.2.14 *Consider an assert move being made by agent A, in an argument inquiry dialogue that asserts b based on the result of an earlier completed warrant inquiry dialogue with topic b. This move would look as follows:* $\langle A, assert, ai, (T \rightarrow b, 1) \rangle$.

The tuple $(T \rightarrow b, 1)$ is the outcome of an earlier warrant inquiry dialogue with topic b, stored in the result store, where T stands for true and 1 is the possibilistic value of belief.

Example 2.2.15 *Consider an open move being made by agent A for a warrant inquiry dialogue with as topic the literal b. This move would look as follows:* $\langle A, open, wi, b \rangle$

Yan et al. propose improvements upon the protocols defined by Black and Hunter [5], which results in similar but more optimized protocols. Their protocols, here referred to as adjusted protocols, are defined as follows:

- Adjusted argument inquiry (AI) protocol:
    1. If the query store is empty, make a close move for the current dialogue.
    2. Else if one of the topic's premises is not yet in the query store, make an open warrant inquiry dialogue move with that premise as the topic.
    3. Else if the premise for the argument of the topic is in the result store, an assert move is made for that premise.
    4. Else if the premise for the argument of the topic is not in the result store, an assert move will be made for the negation of that premise. This also results in two following close moves since the agents can not prove the current premise, and thus also the current argument.
- Adjusted warrant inquiry (WI) protocol:
    1. If the possible beliefs queue is empty, make a close move for the current dialogue.
    2. Else if the first element of the possible beliefs queue is a state belief, make an assert move for this belief.
    3. Else if the first element of the possible beliefs queue is a domain belief, make an open argument inquiry dialogue move for that belief.

Note that the original protocols of Black and Hunter [5] generate and return a set of legal moves. The adjusted protocols stop whenever they find their first legal move, resulting in always returning only one move. The original and adjusted protocols also differ in they way they handle sub-dialogues. The original protocols allow for warrant inquiry dialogues to have multiple embedded argument inquiry dialogues. The adjusted protocols also allow these kinds of sub-dialogues, but additionally they allow argument inquiry dialogues to have embedded warrant inquiry dialogues.

The adjusted dialogues terminate at the same moment when the original dialogues terminate: when both agents make a close move in succession. The outcome of an adjusted argument inquiry dialogue is a belief created from the consequent of the rule that is the topic and a belief value if all the premises of the topic are true. If not all the premises are true, the outcome is null.

Example 2.2.16 *The outcome of an adjusted argument inquiry dialogue with topic $x \rightarrow a$ could be as follows: $(a, p2)$, where p2 is a possibilistic value that stands for probable.*

The outcome of an adjusted warrant inquiry dialogue can be calculated using two different algorithms. Both these algorithms split the made commitments into two sets, one being in support of the topic and one being against the topic. The first algorithm then calculates the greatest lower bound of belief for each of the sets. The set with the highest value wins and decides the outcome. If the values are the same, the set with the highest amount of elements with that value wins. If these are equal again, all elements with that value are removed from the sets and the algorithm restarts. The second algorithm they propose only compares the elements from both sets with the highest value once and does not loop. If the set in support of the topic wins, the outcome will be a tuple of T and the value of the lower bound of belief. If the set against the topic wins the outcome will be a tuple of F and the value of the lower bound of belief. In the case of a draw the outcome is a tuple of U and null.

Example 2.2.17 *The outcome of an adjusted warrant inquiry dialogue with topic b could be as follows:$\langle T, 1 \rangle$.*

Yan et al. their improvements are primarily made on the protocols. Since the adjusted protocols only result in at most one legal move, the strategy used in the system is simple: pick the one legal move given by the protocol. The protocol only gets overruled when in an argument inquiry dialogue one of the premises of the topic is the topic of an unclosed warrant inquiry dialogue. In that case both agents will make a close move and the query store of that dialogue will be reset to the empty set. This is called the fast end strategy and has the purpose of preventing endless loops from happening in the dialogues. The changes made in the protocol result in more clearly structured and more efficient dialogues according to the authors. They claim that, because their system allows the reasoning to be performed in well-defined steps and because their system solves problems one-by-one, their system avoids unnecessary complexity and uncertainty, which gives a more clear and efficient system. They, however, do not give any formal proofs or an empirical comparison to other systems to support these claims.

Yan et al. implemented the system in a web application used for diagnosing dementia. Domain experts model the knowledge of the subject and insert it in the domain repository, which is used by the domain agent, who resembles a domain expert, to reason. Other knowledge known by regular professionals can be inserted as well, but this knowledge is stored in the actor repository. The domain knowledge is stored as state beliefs and their possibilistic values are determined using scales that are linked to the symptoms. This repository and a database with knowledge about a specific patient is used by the professional agent, who resembles a novice physician, to reason. The two agents will perform an inquiry dialogue together and based on their beliefs a conclusion is deducted. An example of a warrant inquiry dialogue using this system from their paper is shown below in table 2.2.

t	m_t	PBQ_{PA}	PBQ_{DA}	QS
1	PA, open, wi, f	PBQ(f)={$(\Phi 2, p2)$}		
2	DA, open, ai, $(\Phi 1, p1)$		PBQ(f)={$(\Phi 1, p1)$, $(\Phi 3, p1)$}	$QS_{\Phi 1}$={a,b,c}
3	PA, open, wi, a	PBQ(a)={$(a, 1)$}	PBQ(f)={$(\Phi 3, p1)$}	$QS_{\Phi 1}$={b,c}
4	DA, close, wi, a		PBQ(a)={}	
5	PA, assert, wi, $(T \rightarrow a, 1)$	PBQ(a)={}		
6	DA, close, wi, a			
7	PA, close, wi, a			
8	DA, open, wi, b		PBQ(b)={}	$QS_{\Phi 1}$={c}
9	PA, close, wi, b	PBQ(b)={}		
10	DA, close, wi, b			$QS_{\Phi 1}$={}
11	PA, close, ai, $(\Phi 1, p1)$			
12	DA, close, ai $(\Phi 1, p1)$			
13	PA, open, ai, $(\Phi 2, p2)$	PBQ(f)={}		$QS_{\Phi 2}$={a,c}
14	DA, assert, ai, $(T \rightarrow a, 1)$			$QS_{\Phi 2}$={c}
15	PA, open, wi, c	PBQ(c)={}		$QS_{\Phi 2}$={}
16	DA, assert, wi, $(T \rightarrow c, 1)$		PBQ(c)={(c,1)}	
17	PA, close, wi, c		PBQ(c)={}	
18	DA, close, wi, c			
19	PA, close, ai, $(\Phi 2, p2)$			
20	DA, close, ai, $(\Phi 2, p2)$			
21	PA, close, wi, f			
22	DA, open, ai $(\Phi 3, p1)$		PBQ(f)={}	$QS_{\Phi 3}$={d}
23	PA, open, wi, d	PBQ(d)={(d,1)}		$QS_{\Phi 3}$={}
24	DA, close, wi, d		PBQ(d)={}	
25	PA, assert, wi, $(T \rightarrow d, 1)$	PBQ(d)={}		
26	DA, close, wi, d			
27	PA, close, wi, d			
28	DA, close, ai, $(\Phi 3, p1)$			
29	PA, close, ai, $(\Phi 3, p1)$			
30	DA, close, wi, f			
31	PA, close, wi, f			

Table 2.2: A warrant inquiry dialogue with dialogue topic f taken from Yan et al. . Embedded argument inquiry dialogues are opened at lines 2, 13 and 22. Further embedded warrant inquiry dialogues are opnened at lines 3, 8, 15 and 23. Whenever an argument inquiry dialogue is opened, an associated query store is initiated, which can be seen in the QS colom. QS and PBQ coloms only show the new or updated information. For brevity sake, $\Phi 1 = (a \wedge b \wedge c \rightarrow f), \Phi 2 = (a \wedge c \rightarrow f)$ and $\Phi 3 = (d \rightarrow \neg f)$. Commitment store and result store are not shown in this example. Commitment stores get updated by assert wi moves and when ai dialogues terminate. Result store is updated when wi dialogues are terminated.

2.2.1.4 Parsons et al.

Parsons et al. [16] describe a system for information-seeking, inquiry and persuasion dialogues. They try to give detailed characteristics of outcomes of argumentation dialogues. The agents can have different attitudes regarding assertion and acceptation. These attitudes determine how strict the agent is when asserting and accepting propositions. In their research one combination of these attitudes is tested, namely thoughtful and skeptical. Agents that are thoughtful and skeptical may assert propositions for which they can construct an acceptable argument and accept propositions if there is an acceptable argument for that proposition. These agents are not able to mislead each other, but there is a possibility that they end up concluding something together of which they are both sure that it should not hold.

Example 2.2.18 *Consider a dialogue between agents A and B. Let agent A have a knowledge base* $\{\neg c, a, a \rightarrow b\}$ *and agent B have a knowledge base* $\{\neg c \rightarrow \neg b\}$ *where* $\neg c$ *and* $\neg c \rightarrow \neg b$ *have a higher preference than all other formulas. If agent A asserts b, agent B will have to accept that assertion even though according to preferences the only acceptable outcome is* $\neg b$*.*

This example shows that the system does not adhere the properties of soundness and completeness, which can be seen as positive for some domains. In other domains such as mechanism design however you do want the outcome to be consistent and predetermined. They finally show that by changing the tactics used by agents you can ensure whether soundness and completeness are present or not. This allows for choosing the right type of dialogue for the context you are using it in. Further on in this section, when looking at the protocol used, we will elaborate on how you can exactly change the protocols to allow for soundness and completeness to be present or not.

Parsons et al. use a system that is inspired by Dung [6], but with an added preference ordering for facts. They split the beliefs, which are propositions, amongst the two agents. Each agent has a public commitment store CS tracking their made commitments and a private knowledge base Σ. Arguments are tuples, the first element being a set of formulae from which the conclusion can be inferred and the second element being the concluding proposition.

Example 2.2.19 *An example of an argument using the above mentioned system is as follows:* $(\{a, a \rightarrow b, b \rightarrow c\}, c)$*.*

Arguments can be attacked on their support by conflicting facts. Conflicts are resolved using a preference ordering.

The move set used by Parsons et al. is the largest out of the considered approaches and consists of four moves: assert, accept, challenge and question. Assert allows the agent to assert a propositional formula or a set of formulas that form the support of an argument. Accept is the counterpart of assert and allows an agent to accept a propositional formula or a set of propositional formulas brought forward by another agent. Challenge allows an agent to force the other participant to give his explicit argument supporting the challenged

propositional formula. Question is used to query the other participant about a propositional formula. Moves are of the form of movetype(p) or movetype(S) depending on whether the subject of the move is a propositional formula or a set of propositional formulas that form the support of an argument. The agents' commitment stores get updated when making assert moves or accept moves.

Example 2.2.20 *Consider an assert move being made that gives the support for the argument* $(\{a, a \rightarrow b, b \rightarrow c\}, c)$*, that would look as follows: assert(*$\{a, a \rightarrow b, b \rightarrow c\}$*).*

Example 2.2.21 *Consider a question move being made that asks the other agents opinion on proposition p. This move would look as follows: question(p).*

Example 2.2.22 *Consider an accept move being made after the other agent has asserted the formula* $\{a \rightarrow b\}$*. This move would look as follows: accept(*$a \rightarrow b$*).*

Example 2.2.23 *Consider a challenge move being made after the other agent has asserted the formula* $\{a \rightarrow b\}$*. This move would look as follows: challenge(*$a \rightarrow b$*).*

Parsons et al. give three protocols, for information-seeking, inquiry and persuasion. Since we are looking at inquiry approaches, only their inquiry protocol is defined below:

- Inquiry protocol:
    1. For the inquiry dialogue it is predetermined what proposition p is going to be the topic of the dialogue. The dialogue starts out with an agent A making an argument for p with antecedent q.
    2. Agent B then either accepts or challenges this assertion based on his attitude.
    3. If B challenges, A will reply with an assert for the support of an argument for the last challenged proposition.
    4. B then goes back to step 2 and goes over all the propositions asserted by A and either accepts or challenges them.
    5. B then asserts q, an argument for q or is undecided and makes an assert(U) move.
    6. If the unified commitment sets of A and B now include an acceptable argument for p, both agents accept p and the dialogue ends.
    7. If not the dialogue goes to step 5 but agent A will have to assert an argument t for r and they try to accept it again.

The purpose of this inquiry protocol is for two agents to answer a question together of which the question is not known to either at the start. The inquiry dialogues of Parsons et al. terminate when the unified commitment sets of the agents include an acceptable argument for the topic or when they can not reach

that conclusion and as a result of that one agent has to make an assert(U) move. The outcome depends on what commitments have been made by the agents when the dialogue finishes. If both agents have the topic in their commitment store, the outcome is the topic. If it is not possible for the agents to agree upon the topic, the dialogue will end with one agent making an U move and with an empty result.

Earlier, in example 2.2.18, we showed that agents using this protocol can end up concluding something of which they are both sure that it should not hold. This property holds for all their protocols, namely: information-seeking, inquiry and persuasion. They claim that it can be seen as beneficial to have the outcome of the dialogue not depend on only the agents' knowledge, but also on the order in which the agents make their moves. This non-predeterminism, as a result of the system lacking the properties of soundness and completeness, allows for rhetorical ability and smart tactics to be built in to the agents. They also argue a different view, which says that outcomes should be predetermined just like economic mechanisms where the order of actions performed does not influence the outcome of, for example, an auction. The key to making the protocols deterministic lies within the acceptable joint knowledge outcomes, the set of acceptable propositions that can be proven by creating arguments using the union of both the agents' knowledge bases. To be deterministic, the protocol has to ensure all propositions needed to establish these outcomes is asserted. The simplest, but also most computationally intensive, method to achieve this is simply asserting all the propositions in both the agents' knowledge bases. A more efficient alternative is to assert those propositions that have a bearing on the acceptability of the dialogue topic. This, however, is difficult to determine since the connections between arguments attacking the topic, and attacking these arguments, is revealed only over time during the dialogue. The only viable solution they finally give is modifying the protocol in three ways. Firstly, any response to a challenge must assert the support of *all* arguments in support of the challenged proposition. Secondly, at every step in the proof, every possible implication that might be the next step of the proof has to be asserted. Finally, agents should not be restricted to have turns anymore.

Parsons et al. do not define strategies for their system, the agents simply have to follow protocol similarly to the approach of Yan et al. [25]. Parsons et al. do not focus on optimizing their dialogues in the amount of moves and the computational complexity. The main purpose of their research is to give a characterisation of the outcome of their dialogue system. They then showed that you can change whether or not the system has the properties of soundness and completeness by changing the protocol used by the agents.

An example of an inquiry dialogue with topic p in this system is given below:

t	m_t	Σ_P	Σ_C	CS(C)	CS(P)
0		$q \to p$, $r \to p$	r		
1	P: assert($r \to p$)			$r \to p$	
2	G: accept($r \to p$)				$r \to p$
3	G: assert(r)				$r \to p$, r

Table 2.3: An inquiry dialogue with topic p. Content of the knowledge bases is given at t = 0 and the columns of the commitment stores have entries when they get updated. The outcome of this dialogue is p since there is an acceptable argument for both agents in the union of their commitment stores.

2.2.1.5 Testerink, Odekerken and Bex

Testerink, Odekerken and Bex [23] propose a method for agent inquiry policies. The aim for this method is for it to be more efficient than exhaustive strategies in inquiry dialogues. The paper proposes an approximation algorithm for stability using labelling. A state is stable when new knowledge will not change the outcome of the argumentation anymore. This stability is used to determine optimal policies, which are used for strategies. The optimal policy is the policy that achieves a stable state with the least amount of actions taken. Their proposed system is not yet a dialogue system, but it is a querying system for an agent.

Testerink, Odekerken and Bex define and use a structured argumentation framework similar to ASPIC+ [18] combined with Dung's grounded semantics for abstract argumentation [6]. The argumentation setup consists of a topic language $\mathcal{L}$, knowledge base $\mathcal{K}$, defeasible rules $\mathcal{R}$, queryable literals $\mathcal{Q}$ and a topic $\tau \in \mathcal{L}$.

Example 2.2.24 *An argumentation setup for a simple fraud scenario would look as follows:*

- $\mathcal{L} = \{f, \neg f, cp, \neg cp, c, p, \neg p, s, \neg s, w, \neg w\}$
- $\mathcal{R} = \{p \Rightarrow c, s \Rightarrow c, (\neg cp, c) \Rightarrow f, w \Rightarrow cp, w \Rightarrow \neg f\}$
- $\mathcal{Q} = \{p, s, w, cp\}$
- $\mathcal{K} = \{p, \neg cp, w\}$
- $\tau = f$

Testerink, Odekerken and Bex assume a hard split of the knowledge between the participants. One agent has access to all the defeasible rules. The other participant has knowledge of how to answer to queries asked by the agent. This results in observations, which are non-defeasible, being added to the knowledge base of the agent after performing queries. These observations, being non-defeasible, cannot be attacked or challenged by other knowledge. The defeasible rules and observations can be used for further inferencing. The agent answering the questions could have a knowledge base containing facts, or it could come

to these facts by using rules but this depends on how one would implement the system.

Arguments in this system are defined recursively. The base case of an argument has the form of $\emptyset => c$ where c is the conclusion derived from an empty set of premises. More complex arguments rely on other arguments and are of the form of $A_1....A_m \Rightarrow c$ where A_1 through A_m is a set of arguments known as the premises and c is is the conclusion of the argument. Base arguments can be inferred from an argumentation setup when the conclusion c exists in the knowledge base $\mathcal{K}$ of the agent. A complex argument can be inferred from an argumentation setup when A_1 through A_m can be inferred from that setup and there is a rule $c_1...c_m \Rightarrow c$ in the set of rules $\mathcal{R}$, such that $c_1...c_m$ are the conclusions of A_1 through A_m. Additionally, c is not allowed to occur in any of the arguments A_1 through A_m.

Example 2.2.25 *An example of an argument in this system is as follows:* $A2 : A1 => c$*, where* $A1 : \emptyset => p$*.*

Arguments can attack each other on their premises and conclusions, as long as the premise or conclusion is not an observation. Arguments can attack other arguments on their sub-arguments as well. An argument defends another argument when it attacks all its attackers. Arguments cannot be cyclic.

Example 2.2.26 *Consider the following three arguments, a subset of the arguments that can be inferred from the argumentation setup given in example 2.2.24:*
A_1*:*$\emptyset => \sim cp$
A_2*:*$\emptyset => w$
A_3*:*$A_2 => cp$
Argument A_1 *attacks argument* A_3*, since the conclusion of* A_1 *negates the conclusion of* A_3 *and the conclusion of* A_3 *is not an observation. The conclusion of* A_3 *also negates the conclusion of* A_1*, but* A_3 *does not attack* A_1*, since the conclusion of* A_1 *is an observation from the agent's knowledge base* $\mathcal{K}$*.*

In the system of Testerink, Odekerken and Bex, the agent has a simple move set consisting of only one possible move, the querying of observable literals. This move is of the form of "x?" where x is the proposition being queried.

Example 2.2.27 *Consider an agent querying for the queryable literal* s*. This move would look as follows:* s*?.*

This move has a similar purpose as the question move used in Parsons et al. [16].

The agent uses the concept of stability to determine what queries it should perform. An argumentation setup is stable when, given the available queries, no new arguments can be added that would change the acceptability of the dialogue topic. Stability of an argumentation setup is determined using the grounded extension of its inferred arguments. The grounded extension is the smallest set of arguments such that there are no conflicts within the set and so that it contains all acceptable arguments.

Example 2.2.28 *Consider the argumentation setup described in example 2.2.24. The inferred arguments from this setup are as follows:*
$A_1{:}\emptyset \Rightarrow p$ $\quad$ $A_5{:}\emptyset \Rightarrow w$
$A_2{:}\emptyset \Rightarrow \sim cp$ $\quad$ $A_6{:}A_5 \Rightarrow cp$
$A_3{:}A_1 \Rightarrow c$ $\quad$ $A_7{:}A_5 \Rightarrow \sim f$
$A_4{:}A_2A_3 \Rightarrow f$
Argument A_2 attacks argument A_6 and arguments A_4 and A_7 attack each other. The grounded extension of this setup consists of the following arguments: A_1, A_2, A_3 and A_5.

Stability of an argumentation setup is based on all its future setups. The future setups of an argumentation setup are all the setups where the still remaining queryable literals are added to the agent's knowledge base, as either positive or negative. An argumentation setup is stable when all of its future setups, including the setup itself, follow one and the same of the following cases:

- Unsatisfiable: the case unsatisfiable holds when there is no argument for the topic τ in the set of inferred arguments of the setup.
- Defended: the case defended holds when there is an argument for the topic τ in the grounded extension.
- Out: the case out holds when there is an argument for the topic τ in the set of inferred arguments of the setup, but all arguments for τ are attacked by an argument in the grounded extension.
- Blocked: the case blocked holds when there is an argument for the topic τ in the set of inferred arguments of that setup but not in the grounded extension of that setup and at least one argument for τ is not attacked by an argument from the grounded extension.

Example 2.2.29 *Consider the argumentation setup defined in example 2.2.24 and the arguments, attack relations and grounded extension from example 2.2.28. This argumentation setup and all its future setups are of the case blocked, since there exist arguments for the topic f and its negation that are both outside of the grounded extension. The only queryable that is left is s, but s and $\neg s$ cannot influence the situation. This argumentation setup thus is a stable setup.*

Stability can be calculated using a brute-force method. This method considers all possible future setups where all queries have been executed, calculates their grounded extensions and determines if their case is different from the current setup. If it is the same for all setups, the setup is stable. When a stable state is reached, the outcome regarding the acceptability of the topic is determined based on what case applies. This outcome is thus based on whether or not an argument for the topic exists in the grounded extension. If there exists one, the outcome is that the topic is defended and thus acceptable. The other three cases, unsatisfiable, out and blocked, all result in a topic that is not acceptable.

The protocol that the querying agent has to follow in the approach of Testerink, Odekerken and Bex is straightforward. The agent has one move that he is allowed to make, the querying of observations. The agent then gets a response from the other agent. The responding agent either confirms the questioned literal by replying x, or denies the questioned literal by replying $\neg x$. This commitment is added to the knowledge base of the querying agent. The aim of this inquiry protocol is for the inquiring agent to form an opinion on the topic of the dialogue. The agent stops querying when either a stable state has been reached or when the set of queryable literals has been exhausted.

Testerink, Odekerken and Bex do not provide explicit strategies for their system. They do give an optimal query policy, which can be used to determine a strategy for inquiry dialogues. They provide multiple optimizations in their system that could lead to faster and more optimized dialogues. The notion of stability allows agents to stop asking questions when it is certain that the acceptability of the topic will not change anymore. Because the agent can stop querying sooner, this will result in shorter dialogues. Since calculating stability is quite costly computationally, they also provide an approximation for stability that is sound and results in a lower complexity algorithm. Assuming that the agent being questioned can vary in responses to the given queries they also define a Markov decision process that allows an agent, given these probabilities of responses, to strategize towards a stable state as fast as possible using an optimal querying policy. The MDP is defined as follows: the actions are the available queries, the states are tuples of argumentation setup and a set of queries, the transition function is the probability of getting a useful answer to a query and the reward function is a negative value for each action. The solution for the problem is thus maximizing the reward on this decision process by minimizing the amount of queries done before reaching stability.

Since Testerink, Odekerken and Bex do not describe a dialogue system, an example of a dialogue can not be given. An execution of the system would look like a series of queries being performed in the form of $x?$ and the other agent replying with either x or $\neg\ x$ until the execution ends.

2.2.1.6 Fan and Toni

Fan and Toni [10] propose a single dialogue model that is capable of generating multiple types of dialogues. The model they propose in their research is, for now, limited to generating inquiry and information-seeking dialogues. Their approach can model multiple dialogue types in a single dialogue model by using game theoretical notations instead of defining separate protocols for each dialogue type.

The approach of Fan and Toni is based on assumption-based argumentation, or ABA for short. ABA is an argumentation framework defined by Dung et al. [7]. Fan and Toni adapt an existing dialogue model for assumption-based argumentation, which is defined by themselves as well in Fan and Toni [8, 9], in order to obtain the general model. ABA serves as the underlying argumentation formalism of their model. The model itself is neutral with respect to the cur-

rent dialogue type and changes behaviour based on its initialization. Dialogues are modelled as games, in which utterances correspond to actions. The different dialogue types are realized by instantiating different utility profiles for the agents, resulting in them performing different actions and strategies. Of importance are the legal-move functions that determine what moves are legal and the strategy-move functions that determine what moves the agents should make. Legal-move functions enforce that there are no repeated utterances towards the same target in a dialogue. Fan and Toni face a challenge that is twofold, firstly they must map dialogue notions to game-theorectic notions in a generic way, so that dialogue models are mapped to game models. Secondly, they must define different utility profiles that are able to model agents' behaviour in the different types of dialogues.

Agents in this system have internal private beliefs and they share information with each other through a shared language. This language is that of ABA, and is used to exchange rules, assumptions and contraries using an underlying logical language $\mathcal{L}$. Assumptions are literals and contraries are negations of such literals. Rules are of the form $\beta_0 \leftarrow \beta_1, ..., \beta_m$. In a rule β_0 is known as the head of the rule and $\beta_1, ..., \beta_m$ is known as the, possibly empty, body of the rule. The head and body of the rule are all elements of $\mathcal{L}$ and the body of the rule cannot contain duplicates.

Example 2.2.30 *An example of a rule in this system is as follows:* $boy\_innocent \leftarrow boy\_not\_proven\_guilty$.

Agents are equipped with an ABA framework that represents their knowledge and are often denoted by this framework.

Example 2.2.31 *An agent α in this system can be denoted as its ABA framework as follows: $\alpha = \langle \mathcal{L}, \mathcal{R}_1, \mathcal{A}_1, \mathcal{C}_1 \rangle$, where $\mathcal{L}$ is the underlying logical language, $\mathcal{R}_1$ are the rules known to the agent, $\mathcal{A}_1$ are the assumptions known to the agent and $\mathcal{C}_1$ are the contraries known to the agent.*

The joint framework of two agents, representing their shared knowledge, is denoted as the union of the agents' individual ABA frameworks. Both the individual frameworks and the joint framework are flat, meaning that the frameworks have no assumptions in the heads of rules and that all assumptions have contraries.

Example 2.2.32 *The joint framework $\mathcal{F}_J$ of two agents is denoted as follows: $\mathcal{F}_J = \langle \mathcal{L}, \mathcal{R}_1 \cup \mathcal{R}_2, \mathcal{A}_1 \cup \mathcal{A}_2, \mathcal{C}_J \rangle$, where $\mathcal{C}_J(\alpha) = \mathcal{C}_1(\alpha) \cup \mathcal{C}_2(\alpha)$, for all α in $\mathcal{A}_1 \cup \mathcal{A}_2$.*

Utterances in this system are defined as tuples $\langle a_i, a_j, T, C, ID \rangle$ where a_i is the speaker agent, a_j is the target agent, T is the target utterance of the move, C is the content of the move and ID is the identifier of the move. The content C of a move can be one of the following: a claim, a rule, an assumption or a contrary.

Example 2.2.33 *An example of a move in this system that utters a rule is as follows: $\langle \alpha, \beta, 1, rl(boy\_innocent \leftarrow boy\_not\_proven\_guilty), 2 \rangle$. In this move, rl*

indicates that a rule is being uttered and the specific rule is notated within the following parentheses. The head and body of the rule are both elements of $\mathcal{L}$.

Dialogues are defined according to the earlier defined dialogue system of Fan and Toni [8, 9]. Dialogues are a sequence of utterances that target earlier made utterances, starting with a claim utterance. A strategy-move function determines what moves the agents should make. A strategy-move function is a mapping from the legal moves of the current dialogue, generated by some legal-move function, to the set of utterances. The system described in Fan and Toni [10] does not assume a concrete legal-move function, but rather considers the set of all legal-move functions. They give three different strategy-move functions, which are used to define the possible behaviours of the agents.

- The *thorough* strategy-move function constructs dialogues that contain all information that is relevant to the topic from both agents. Agents following this strategy utter all their rules, assumptions and contraries. These dialogues have the desirable property that admissible arguments obtained in the dialogue are also admissible in the joint ABA framework of the two agents.
- The *non-attack thorough* strategy-move forces the agents using it to utter all rules and assumptions, but not contraries that are related to some utterance in the dialogue.
- Agents using the *pass* strategy-move function will only make the initial claim move of the dialogue, setting the dialogue topic. Besides this the agent does not utter any rule, assumption or contrary in the dialogue.

Next, they define a mapping from dialogue notions to game theory notions, in order to study the behaviour of the agents in a game based framework. The types for agents α and β, indicating what behaviour they should follow, are defined as θ_α and θ_β respectively. The action space of an agent, the actions that the agents can make in a game, is its possible utterances. The dialogue strategy of an agent, the strategy followed by the agent in this game, is the set of utterances made by that agent in a dialogue. Because this set of utterances is determined by the strategy-move function that the agent uses, we can equate a dialogue strategy with the strategy-move function used by the agent in this dialogue. The game-theoretic outcome of the dialogue is equal to the ABA framework drawn from the dialogue, since this framework represents all information disclosed by the agents. These notions are all generic and do not depend on the dialogue type.

Fan and Toni define two types of inquiry dialogues, which they call I-Type I dialogues and I-Type II dialogues. The goal of I-Type I dialogues is to test the admissibility of the topic in $\mathcal{F}_J$. This is comparable to warrant inquiry dialogues as defined by Black and Hunter [5]. The goal of I-Type II dialogues is to test whether an argument for the topic exists in $\mathcal{F}_J$. This is comparable to argument inquiry dialogues as defined by Black and Hunter. Arguments in ABA are deductions of claims using rules. An argument for the claim β is a finite trees where the root is labelled with the claim of the argument β. The leaves

of this tree are labelled with either assumptions from the premises of the rule or τ representing an empty body. Non leaves are marked β' with, as children, the premises of a rule with as its conclusion β'. An argument A_1 attacks an argument A_2, when the claim of A_1 is a contrary of one of the assumptions in A_2.

The utility functions of agents in I-Type I dialogues punishes the agents for not having disclosed rules, assumptions and contraries that would be related to the dialogue outcome and for every element of the outcome that was not given by the agent. Rules, assumptions and contraries used to build a dispute for an argument are related to the claim of the argument. This utility function accurately reflects the agents' need to find out the acceptability of the topic with respect to their joint knowledge. The authors give a theorem indicating that the *thorough* strategy-move function is the dominant strategy for agents using this utility function. The common good of two agents in a dialogue is defined as the outcome of a social choice function. This social choice function has as its outcome an ABA framework $\langle \mathcal{L}, \mathcal{R}_{i1}, \mathcal{A}_{i1}, \mathcal{C}_{i1} \rangle$ containing the rules and assumptions from $\mathcal{F}_J$ related to the topic and the contraries of all assumptions in the social choice outcome. The common good thus is that any information related to the claim of the dialogue topic in the agents' individual knowledge bases must be disclosed. Dialogues constructed using the *thorough* strategy-move function meet this common good. An example of an I-Type I dialogue is given in table 2.4.

$\langle \alpha, \beta, 0, claim(boy\_innocent), 1 \rangle$
$\langle \beta, \alpha, 1, rl(boy\_innocent \leftarrow boy\_not\_proven\_guilty), 2 \rangle$
$\langle \alpha, \beta, 2, asm(boy\_not\_proven\_guilty), 3 \rangle$
$\langle \beta, \alpha, 3, ctr(boy\_not\_proven\_guilty, guilty), 4 \rangle$
$\langle \alpha, \beta, 4, rl(guilty \leftarrow W1), 5 \rangle$
$\langle \beta, \alpha, 5, asm(W1), 6 \rangle$
$\langle \alpha, \beta, 6, ctr(W1, not\_W1), 7 \rangle$
$\langle \alpha, \beta, 7, rl(not\_W1 \leftarrow contradicted), 8 \rangle$
$\langle \alpha, \beta, 8, rl(contradicted \leftarrow), 9 \rangle$
$\langle \beta, \alpha, 4, rl(guilty \leftarrow W2), 10 \rangle$
$\langle \alpha, \beta, 10, asm(W2), 11 \rangle$
$\langle \beta, \alpha, 11, ctr(W2, not\_W2), 12 \rangle$
$\langle \alpha, \beta, 12, rl(not\_W2 \leftarrow W2\_has\_poor\_eyesight), 13 \rangle$
$\langle \beta, \alpha, 13, rl(W2\_has\_poor\_eyesight \leftarrow), 14 \rangle$

Table 2.4: An I-Type inquiry dialogue. The game-theoretic outcome is the same as the joint framework F_J of the two agents.

Behaviour for I-Type II dialogues is created by altering the utility functions. Agents in this type of dialogue are only concerned with finding all arguments for the topic, so it is not in their interest to utter contraries creating arguments that could attack the topic. The utility function punishes the agents for not having disclosed rules and assumptions that are rule-related to the topic and for each

element of the outcome that was not given by the agent. Rules and assumptions used to construct an argument are rule-related to the claim of the argument. The authors give a theorem indicating that the *non-attack thorough* strategy-move function is the dominant strategy for this utility function. The outcome of the social choice function is an ABA framework $\langle \mathcal{L}, \mathcal{R}_{i1}, \mathcal{A}_{i1}, \mathcal{C}_{i1} \rangle$ that contains all rules and arguments from the joint outcome $\mathcal{F}_J$ that are rule-related to the topic, but contains no contraries. The common good thus is finding all rules and assumptions that form arguments for the dialogue topic. Dialogues generated using the *non-attack thorough* strategy-move function meet this common good for both agents.

2.2.1.7 Summarizing

We have looked at six different systems for inquiry dialogues. These systems all have their own characteristics, strengths and weaknesses.

System	**Knowledge bases**	**Locutions**
Black and Hunter 2009	Defeasible rules and defeasible facts	Open, close, assert
Kumeling 2018	Defeasible rules and defeasible facts	Open, close, assert
Yan et al. 2018	Possibilistic beliefs	Open, close, assert
Testerink, Odekerken and Bex 2019	Defeasible rules and observations	Querying literals
Parsons et al. 2003	Propositions	Assert, Accept, Challenge, Question
Fan and Toni 2015	Rules, assumptions and contraries	Claim, rule, assumption, contrary

Table 2.5: Comparing knowledge bases and locutions

Table 2.5 gives an overview of the types of knowledge and available locutions in each of the approaches. Black and Hunter [5] and Kumeling [14] are based on the same system. They are both based on defeasible logic and agents can make open, close and assert moves. Yan et al. [25] use a similar system, but instead of defeasible logic they make use of possibilistic logic. The available moves are the same. Testerink, Odekerken and Bex [23] make use of defeasible logic. A difference with Black and Hunter [5] is that Testerink, Odekerken and Bex [23] make use of observations, instead of the defeasible facts Black and Hunter use [5]. The agent of Testerink, Odekerken and Bex [23] has the simplest moveset out of the considered approaches. The agent can only query for observations. The approach of Parsons et al. [16] makes use of propositional formulas. Their approach has the largest moveset out of the approaches, together with the system of Fan and Toni [10]. Agents can assert, accept, challenge and question propositional formulas. Agents in the system of Fan and Toni [10] have knowledge of rules, assumptions and contraries. In their moves they can utter a claim for an

assumption from their knowledge, utter a rule, utter an assumption or utter a contrary to some assumption.

System	**Protocol**
Black and Hunter 2009	AI protocol and WI protocol
Kumeling 2018	AI protocol and WI protocol
Yan et al. 2018	Adjusted AI protocol and adjusted WI protocol
Testerink, Odekerken and Bex 2019	Can only query observations. Stops at stable state
Parsons et al. 2003	Varies between dialogue types
Fan and Toni 2015	Moves have to come from some legal-move function

Table 2.6: Comparing protocols

Table 2.6 gives an overview of the different protocols used in each of the approaches. Black and Hunter [5] defined protocols for argument inquiry dialogues and warrant inquiry dialogues. These exact protocols were also used by Kumeling [14]. Yan et al. [25] have proposed improvements for these two protocols. Their improvements return only one legal move, in stead of the set of legal moves that the protocols of Black and Hunter [5] return. They also allow a deeper nesting of sub-dialogues. In the protocols of Yan et al. [25], warrant inquiry dialogues can have nested argument inquiry dialogues and argument inquiry dialogues are allowed to have nested warrant inquiry dialogues. In the protocols of Black and Hunter [5], only warrant inquiry dialogues are allowed to have nested argument inquiry dialogues. The simplest protocol is the one used by Testerink, Odekerken and Bex [23]. This protocol allows the agent to make one move, questioning literals, to which he then either gets a positive or a negative response. Parsons et al. [16] give multiple protocols, of which one for inquiry dialogues. This protocol is rather simple and does not allow sub-dialogues. Fan and Toni [10] have agents that have to follow legal-move functions. These are functions that enforce that there are no repeated utterances towards the same target in a dialogue. In their paper they do not assume a specific legal-move function, but instead consider the set of all legal-move functions.

System	**Strategy**
Black and Hunter 2009	Exhaustive strategy.
Kumeling 2018	Limited exhaustive strategy and smart original strategy
Yan et al. 2018	Pick the one legal move + Fast end strategy
Testerink, Odekerken and Bex 2019	Use the optimal policy as basis for strategies
Parsons et al. 2003	-
Fan and Toni 2015	Strategy-move functions

Table 2.7: Comparing strategies

Table 2.7 gives an overview of the different strategies used in each of the approaches. The strategy used by Black and Hunter [5] is exhaustive and thus asks each question that could have an impact on the dialogue. This can result in

many redundant moves being made, but it also results in favourable properties such as soundness and completeness of the strategy. Kumeling [14] tries to improve the exhaustive strategy so that less moves will be made whilst still adhering the properties of soundness and completeness. This is realized by creating more strict picking functions for assert moves in the strategy. Yan et al. [25] do not use any variation of the exhaustive strategies. Instead their proposed strategy is rather simple and comes down to picking the only legal move that is returned by their adjusted protocols. They do not give any statements or proofs that indicate their approach is still sound and complete. Testerink, Odekerken and Bex [23] do not provide an explicit strategy to use in their system. Instead they provide a policy that can be used to base a strategy on. The strategy used by Parsons et al. [16] is the same as the one used by Yan et al. [25], the strategy strictly follows the protocol and makes the one move it is told to make. Fan and Toni [10] their agents decide what move to make based on their strategy-move functions. In their inquiry dialogues, the agents make use of two different strategy-move functions. In I-Type I dialogues the agents will utter all information that is relevant to the topic. In I-Type II dialogues the agents will utter all rules and assumptions relevant to the topic, but no contraries. These strategy-move functions maximize the utility gained according to the related utility functions of these types of dialogues.

System	**Optimizations**
Black and Hunter 2009	None
Kumeling 2018	Limited strategy and Smart strategy
Yan et al. 2018	Protocol only returns 1 legal move
Testerink, Odekerken and Bex 2019	Concept of stability and approximation of stability
Parsons et al. 2003.	None
Fan and Toni 2015	None

Table 2.8: Comparing optimizations

Table 2.8 gives an overview of the different optimizations in each of the approaches. Since Black and Hunter [5] use a relatively inefficient exhaustive strategy, we will use their approach as a base case against which we can compare the optimizations used by other approaches. The optimizations of Kumeling [14] are realised in the used strategies of the system. In the end none of the proposed strategies was able to both keep the properties of soundness and completeness and decrease the size of either the dialogue and the generated dialectical tree. Yan et al. [25] have optimized their approach on the protocol side. Their protocol only returns one legal move. This results in a more efficient system than that of Black and Hunter [5]. They also claim that their dialogues have a clearer structure because of the additional nesting they allow for sub-dialogues. Testerink, Odekerken and Bex [23] have proposed multiple optimizations for inquiry dialogues. Firstly, the notion of stability allows an agent to stop querying when the outcome is already certain. Secondly, because stability can be costly to compute, they also give an approximation algorithm to calculate stability. This

approximation algorithm is sound, but not complete. This means that it can mislabel some stable states as not yet stable.

All the approaches have their own measures and priorities of performance. Black and Hunter [5] stress that their approach is for a safety critical domain and that it is important to them that the approach gives guarantees for soundness and completeness. Kumeling [14] also values these properties, but in additions tries to increase the performance in dialogue length and size of dialectical trees as well. Yan et al. [25] also want more efficient dialogues than that of Black and Hunter [5]. They want their approach to be feasible in real life instead of just as a piece of research. Additionally they want their dialogues to be more clear and structured than those of Black and Hunter [5]. Testerink, Odekerken and Bex [23] their main focus lies on dialogue length, computational complexity of dialogues and explainability of dialogues. Parsons et al. [16] measure whether or not their protocols are deterministic or not. They argue that for certain applications soundness and completeness are a benefit whilst for other applications it is not wanted. Fan and Toni [10] show a interesting approach in which they model multiple types of dialogues into one model using game-theory. They establish a generic correspondence between dialectical concepts and game notions. Altering the utility profiles used by the agents is enough to model the behaviour of a different type of dialogue.

2.2.2 Systems for information-seeking

A dialogue type that is similar to inquiry is that of information-seeking. Not a lot of research has been done yet on this topic. In information-seeking dialogues one agents seeks to answer some question made by the other agent. Information-seeking dialogues could be used to model an intake scenario by a police officer, just like inquiry dialogues can. This makes these system interesting for us to discuss and reflect on in this literature review.

2.2.2.1 Parsons et al.

Besides their inquiry protocol, Parsons et al. [16] also define a protocol for information-seeking dialogues. This protocol is made for the same system that is used by Parsons et al. [16] for their inquiry dialogues. The difference solely being the used protocol:

- Information-seeking:
    1. The information-seeking protocol starts with an agent A making a question move towards an agent B.
    2. B will then reply positively, negatively or indecisively with an assert move.
    3. If B replies either positively or negatively then agent A will either accept or challenge that assertion based on his attitudes. If A accepts then the dialogue is over.

4. If A challenges then B will have to reply with assertions that make up the support of their earlier asserted argument.
5. The dialogue then returns to step 3 where A will have to choose to either accept or challenge the made assertions.

This protocol terminates when agent B makes an U move because there is a lack of knowledge, when an agent repeats a locution or when agent B has accepted either p or ¬p. The outcome of the dialogue then depends on the opinions of both agents of topic p. When both agents have p in their set of acceptable knowledge outcomes the outcome of the dialogue is p. When both agents have ¬p in their set of acceptable knowledge outcomes the outcome of the dialogue is ¬p. An acceptable knowledge outcome is a proposition x, which is the conclusion of an argument that resides in the union of the agent's knowledge base and the other agent's commitment set.

2.2.2.2 Fan and Toni

Besides the two types of inquiry dialogues Fan and Toni [10] show in section 2.2.1.6, they also show how agents in their system can perform information-seeking dialogues. Information-seeking dialogues are performed by two agents, the questioner agent and the answerer agent. In these dialogues, the questioner agents utters a claim which he desires to be confirmed. This is the only contribution in the dialogue given by the questioner. The answerer responds by uttering all information information in his knowledge base that is in support of the topic.

The utility functions that allow the agents to perform a dialogue like the one described above, are as follows: the questioner agent gets utility for uttering the initial claim, and nothing else. The answerer agents is punished for not uttering rules and assumptions that are related to the topic or to other rules in the outcome of the dialogue. The answerer agent also is punished for all the elements of the outcome that did not come from his knowledge. These utilities result in the *pass* strategy-move function being the dominant strategy for the questioner and the *non-attack thorough* strategy-move function being the dominant strategy for the answerer, as given by the authors in a theorem. The outcome of the social choice function, denoting the common good of the two agents, is an ABA framework $\langle \mathcal{L}, \mathcal{R}_{i1}, \mathcal{A}_{i1}, \mathcal{C}_{i1} \rangle$ containing all rules from the answerer that are rule-related to the topic, all assumptions from the answerer that are rule-related to the topic and no contraries. This outcome means that the common good for both agents is that the answerer agent gives all his information in support of the claim, but nothing else. The agents meet the common good when they follow the strategy-move functions that are also their individual dominant strategies. An example of an information-seeking dialogue in this system is shown in table 2.9.

$\langle \alpha, \beta, 0, claim(w1\_not\_believable), 1 \rangle$
$\langle \beta, \alpha, 1, rl(w1\_not\_belieable \leftarrow w1\_contradicted\_by\_w2), 2 \rangle$
$\langle \beta, \alpha, 2, rl(w1\_contradicted\_by\_w2 \leftarrow), 3 \rangle$

Table 2.9: An information-seeking dialogue where α is the questioner and β is the answerer.

2.3 Other strategies

Besides gathering inspiration from just information-seeking dialogues and inquiry dialogues, we can also look at systems that perform other types of dialogues. The most research has been done on the topic of persuasion dialogues. Interesting findings in this field could be used in inquiry and information-seeking dialogue systems as well.

2.3.1 Opponent models

A method used to increase performance in particularly persuasion dialogues is that of opponent models. An agent will create a model of the opponent he is facing. This model can consist of different types of information such as assumed beliefs and assumed rules that the opponent has. Using this model the agent can determine what moves would be effective against this specific opponent.

Hadjinikolis et al. [12] give a general methodology for updating and augmenting opponent models in argumentation games. They do this by modelling the likelihood of beliefs and arguments that the agent knows the opponent knows and what the opponent then probably knows. The models are maintained and updated through dialogues. Because their method is computationally intensive, they make use of sampling.

Hadoux et al. [13] try to find the optimal policy of an agent when playing against a stochastic opponent. Their model is state based, uses Dung's grounded semantics [6] and forces agents to always make a move if they have a legal move available. They apply optimizations to make computation more feasible. These optimizations remove irrelevant arguments, infer attacks made from arguments, prune certain moves with respect to the initial state of the problem and prune arguments that are dominated. Depending on the size of the problem however, the computations could still be costly even after the optimizations.

Another approach to opponent modelling is that by Rienstra et al. [19]. They implemented three different opponent models, increasingly complex, and tested them. The first model is a simple and certain model of the opponent. The second model is a set of opponent models each with a probability of being true. The final model is the same as the previous, but adds in so called virtual arguments to the model. These virtual arguments are arguments that you believe the opponent has, but you do not know the exact form of said argument. They show that as they increased the complexity of the opponent model the performance of the argumentation would increase as well.

2.3.2 Q-Learning

Alahmari et al. [1] try to make agents more adaptable to new situations in persuasion dialogues, whereas a lot of work on this topic so far has been on negotiation dialogues. The framework used for this research is the abstract argumentation framework of Dung with the grounded extension [6]. They teach the agent how to effectively play the game using Q-learning with as reward function the number of acceptable arguments in the grounded extension. The agent receives its reward only at the end of the learning session. The results in general were encouraging but the way they represented states had some weaknesses in it.

2.3.3 Planning

Black et al. [4] translate the persuasion dialogue problem to a planning problem. Doing this they can use regular solvers for planning problems to solve persuasion dialogues. The proponent of the persuasion dialogue has a uncertain opponent model. The goal is to find strategies that are successful independent of the fact if the opponent model is correct or not. The possible dialogues of all the possible opponent models are tracked. The success rate for a plan is determined by calculating the results of the plan against all possible dialogues of all possible opponent models. Summing the results of each possible model results in the probability of success for that plan. The probabilities of guaranteed success against the different opponent models are adjusted after every action the opponent makes. The result is that this approach gives a certain probability of the guaranteed chance of success. Their approach usually was faster than the comparison, especially on more complicated problems. A limitation of this research is probably the scalability and that it only considers simple strategies, but this did not greatly effect the results.

2.4 Summary

In this chapter we have looked at literature that could support this thesis. First we looked at different research papers of the AI for policing project. These are done by Utrecht University in collaboration with the Dutch national police. This thesis will also be in support of this project. Next we looked at different inquiry dialogue systems. By looking at these systems we could identify the current standings of research and determine where new possibilities for research were present. A promising new approach is that of Testerink, Odekerken and Bex [23]. They proposed a querying system that minimizes the amount of moves needed before terminating by using the concept of stability. The downside of their approach however is that it is not yet a dialogue system, so it would have to be converted and implemented into one. We shortly looked at dialogue systems of other types than inquiry. Doing so we could identify strategies and techniques that might be of interest for inquiry dialogues as well.

There clearly are multiple measures of performance for these kind of systems. The systems need to have certain guarantees for the outcome of the dialogues it generates, such as soundness and completeness. But on the other hand, dialogues also need to be limited in the amount of moves made. If the amount of moves needed for a conclusion is too big, participants might lose interest. Related to that, it also is important that the dialogue system contains some form of explainability. If the system is going to be used in real life, the system should be able to explain how it came to it's conclusion and why it made the moves that it has made. It would be useful to quantify a performance measure that takes into account these aspects.

Currently, none of the implemented inquiry dialogue systems excel at both guaranteeing results and having as short as possible dialogues. There are systems like that of Black and Hunter [5] that are sound and complete. This gives good guarantees for the quality of the outcome of the dialogues their system generates. Their approach however makes too many redundant moves and is not very feasible in real life situations. It is clear that their is a need for a system that can keep the length of the dialogue minimal. Approaches such as Kumeling [14] and Yan et al. [25] try to make the approach of Black and Hunter more viable for real life situations by reducing the dialogue length and trying to provide more clear dialogues. Some improvements are made, but there is still room for improvement. Techniques from other dialogue types such as opponent models and machine learning could be implemented to improve the inquiry approaches, but there still is enough research to be done on the topic of inquiry dialogues itself. Testerink, Odekerken and Bex have proposed interesting and promising optimizations for inquiry processes using the concept of stable states. Their system has the potential to lead to shorter and more explainable inquiry dialogues. Their system however is not yet a true dialogue system, but a querying system. It would be interesting to see their system worked out into a full dialogue system. This way, we can introduce stability-based inquiry systems as a new class of inquiry systems in agent argumentation. We can realize this translation using the knowledge and ideas that we have acquired from other research done on inquiry dialogues. Translating their system into a dialogue system allows for us to perform experimentation. In the current system the agent is limited in the actions he can make and the queries he can perform. The agent can only query for observables and not for arguments. The relatively small scale of the protocol, speaker model and opponent model allow for different extensions and adjustments to be defined. This can be used for experiments that test the impact of different set ups regarding knowledge distribution, rule sets and available locutions. The opponent model and speaker model describe what knowledge both agents have at their disposal and what types of locutions they can make.

Chapter 3

Performance measures

To be able to measure the difference in performance between different versions of our dialogue system, we first have to define the relevant performance measures that we will use to do so. We will go over the performance measures found throughout the related literature and decide which ones are most fitting and relevant for our inquiry system.

It is important that the amount of questions needed in order to form a verdict is minimal. If the person submitting a complaint to the system has to answer too many questions, they might lose interest and concentration. This would most likely not be beneficial for the amount and quality of complaints handed to the system, defeating the purpose of the system. This metric is one of the easiest on which the different setups can be compared. This metric can compare systems through the use of simulations of dialogues. By simulating these dialogues an average dialogue length can be calculated and compared. Since more complex argumentation setups will almost always have a longer average dialogue length, we will have to introduce a normalisation factor that takes this into account when comparing the performance of different argumentation setups. As an indication of the complexity of an argumentation setup, we take the amount of queryable literals it contains. We then divide the raw average dialogue length by this factor to obtain the normalised average dialogue length.

Example 3.0.1 *Consider two systems S_1 and S_2. S_1 performs strictly better in dialogue length than S_2 when the normalised average amount of moves made in dialogues of S_1 is smaller than the normalised average amount of moves made in dialogues of S_2.*

Definition 3.0.1 *(Normalised Average Dialogue Length). The normalised average dialogue length of a system is the average dialogue length, divided by the amount of queryables of the related argumentation setup. Formally: $d_{norm} = \frac{d}{|\mathcal{Q}|}$, where d is the average dialogue length.*

Example 3.0.2 *Consider a system with an average dialogue length of six moves.*

The argumentation setup that is being tested has four queryable literals. The normalised average dialogue length then is calculated by $d_{norm} = \frac{6}{4} = 1.5$.

Computational complexity of the dialogues is an important factor to consider, but lies outside the scope of this thesis. Same as with the dialogue length, if the computational complexity of performing dialogues is too high, the end user will experience this as a negative as they will have too wait long times to get an asnwer from the system. It is thus preferable for the computational complexity of dialogues themselves and the learning of any possible policies to be minimal, but we will not further consider this metric for our research.

Expressiveness of a dialogue system is an important measure of performance. The measure of expressiveness of a dialogue system indicates how many different actions and locutions the agents using it have at their disposal. In contrast to dialogue length and computational complexity, expressiveness is less intuitive to define. It is preferable for the agents to express their arguments and reasoning in a multitude of ways. One way to measure the expressiveness is to simply count the number of available locutions for the agents to utter. Another way is to categorize systems together based on the types of locutions they have and give an arbitrary number to these categories. This way you can decide to not count two similar locutions as two additions to the amount of expressiveness, contrary to the model that simply counts all available locutions and would allow both locutions to influence the number. For our choice we will use the categorical way of measuring the expressiveness of a system. We will group systems into three distinct categories: low expressiveness, regular expressiveness and high expressiveness. Logically, systems that fall in a higher category of expressiveness are preferred over systems that fall in a lower category of expressiveness.

Definition 3.0.2 *(System Expressiveness). The expressiveness of a system is measured by fitting in one of the following categories:*

1. *Low expressiveness: systems in this category are able to generate dialogues in which one agent can pose questions for non-defeasible literals and the other agent can reply accordingly.*

2. *Regular expressiveness: systems in this category are able to generate dialogues in which one agent can pose questions for both defeasible and non-defeasible literals and the other agent can reply accordingly to the type of question posed.*

3. *High expressiveness: systems in this category are able to generate dialogues in which one agent can pose questions for both defeasible and non-defeasible literals and the other agent can reply accordingly to the type of question posed. Additionally, the questioning agent can ask for explanations and arguments for the uttered replies to questions regarding defeasible literals. These should be answered accordingly by the replying agent.*

It is preferable for the strategy that the agent uses to be sound and complete, since dialogue systems are often used as support for decision making in

a variety of domains, such as the legal and medicals domains. Soundness and completeness, for the system we will propose in this thesis, depend on whether or not the system finds an argument for the topic literal in the grounded extension of its argumentation setup. Since positive outcomes should get checked by human expert afterwards as well, it is less important for the system to be sound than complete. It is desirable though, however, to minimize the amount of false positives that are given by the system. Soundness and completeness for the proposed dialogue system are defined as follows:

Definition 3.0.3 *(Soundness and Completeness).*

- *Soundness: A system S is sound iff when, given an argumentation setup AS, if the dialogue concludes with an argument in the grounded extension for the topic τ, then an argument for the topic τ would also be in the grounded extension of the union of the two agents' knowledge bases.*

- *Completeness: A system S is complete iff when, given an argumentation setup AS containing the union of the two agents' knowledge bases, if an argument for the topic is in its grounded extension, the system will always conclude with an argument for the topic τ being in the grounded extension.*

Since dialogue systems are often used as support for real-life decision making, it is preferable for the system to be able to explain the decisions it has made transparently. This idea is explored, for example, by Testerink, Odekerken and Bex [23]. They stress that their approach should be able to explain the decisions it makes during its inquiries. Having a system be able to explain itself, likely gives it more support to be used in real-life scenarios. It also gives insight in decisions that are being made that seem counter-intuitive, but would make sense from the point of view of the agent. We thus deem systems that are able to explain the decisions that they make and the outcome that they have generate preferable over systems that do not provide any source of explainability.

In domains where agents of different organizations have to work together and exchange information, there might be a sense of privacy that makes agents not want to give up any more sensitive information than is needed for a conclusion to be reached. Maximizing this sense of privacy can have real-life impact in multiple ways. Firstly, it could be that the information that has to be shared must first be retrieved or calculated by the other party. When minimizing the amount of information shared, the agents thus also minimize the amount of time-costing actions that they must perform parallel to the dialogue. Secondly, the organizations of the agents that are conversing with each other have a guarantee of maximal possible privacy, in the sense that no more of their information will be shared than is necessary for the agents to reach a verdict. It is thus preferable if there are measures built into the dialogue system, that allow the agents to minimize the amount of sensitive information that is being shared.

In this chapter we have looked at different measures of performance that could be used to formalize the performance of an inquiry dialogue system in order to answer our first research question: "What are the best performance

measures for inquiry type dialogues that model a police intake conversation?". We first discussed the average dialogue length as a performance measure and showed how it could be normalised in order to compare systems that perform dialogues using different sizes of argumentation setups. Dialogue length is an important factor for intake conversations, since complainants might not use the system at all if the dialogue length is too large. We briefly discussed the computational complexity of systems and how it can be used to measure performance, but we deemed it unnecessary, within the scope of this thesis, to consider it whilst discussing the performance of our own system. Next we looked at the expressiveness of a dialogue system. Expressiveness indicates in how many ways the agents can express their arguments and ideas. When considering the measure of expressiveness isolated, it always is better to have a higher rate of expressiveness in a system. We defined three categories of expressiveness of systems, based on the dialogue system we will define later in this thesis, that can be used to compare different systems based on expressiveness. Expressiveness can be favourable in intake situating, allowing the complainant to respond in more diverse ways to the agent and the agent being able to ask more types of questions. We then looked at the favourable properties of soundness and completeness, of which it is preferable that a dialogue system should both have. We defined how soundness and completeness can be determined for our proposed dialogue system, which will be given later in this thesis. Since police intake conversations are in the legal domain, it is highly preferable for the system to be both sound and complete. A characteristic that is nice to have, but not a necessity, is that of explainability. It is preferable for a system that is to be deployed in real-life to be able to explain the decisions it has made during a dialogue. Finally, we looked at the amount of information shared as a performance measure. Depending on the domain, it is preferable for the amount of information shared in a dialogue before reaching a conclusion to be minimal. A complainant in a police intake conversation might not want to give away anymore private information than necessary, making this performance measure important for this domain.

Chapter 4

Dialogue system

For our experiments a dialogue system has to be defined. This dialogue system is based on the inquiry system proposed by Testerink, Odekerken and Bex [23] and uses concepts and definitions from their paper.

4.1 Logic

The underlying logic used in the system is similar to that used in ASPIC+ [18] and uses a combination of propositional literals and defeasible rules. The foundation of the system is known as an argumentation setup, which determines what the agents can reason about and what literals can be queried.

Definition 4.1.1 *(Argumentation Setup, AS). An argumentation setup AS is a tuple* $AS = (\mathcal{L}, \mathcal{R}, \mathcal{Q}, \mathcal{K}_I, \mathcal{K}_R, \tau)$ *where:*

- $\mathcal{L}$ *is a logical language consisting of propositional or ground predicate-logic literals.*
- $\mathcal{R}$ *is a set of defeasible rules* $p_1...p_m \Rightarrow q$ *s.t.* $p_1...p_m, q \in \mathcal{L}$. $p_1...p_m$ *are called the antecedents of a rule and q the consequent. The antecedents of a rule are unordered.*
- $\mathcal{Q} \subseteq \{l \in \mathcal{L} | l \neq \neg p\}$ *is a set of queryable literals.*
- $\mathcal{K}_I \subseteq \{l \in \mathcal{L} | l \in \mathcal{Q} \vee \neg l \in \mathcal{Q}\}$ *is a knowledge base of observations belonging to the inquirer agent. This knowledge base is a consistent set of query results.*
- $\mathcal{K}_R \subseteq \mathcal{L}$ *is a knowledge base of observations belonging to the respondent agent.*
- $\tau \in \mathcal{L}$ *is a topic.*

An argumentation setup AS is shared between the participating agents, and defines the possible dialogues that can be performed by the agents. Note that the agents cannot access each others knowledge bases and thus cannot use this information to infer arguments or reason with in any other way. In the future of this chapter when we refer to an argumentation setup as a tuple $\text{AS}_a = (\mathcal{L}, \mathcal{R}, \mathcal{Q}, \mathcal{K}, \tau)$, it is assumed that this is the argumentation setup as known by one of the two agents a, with the knowledge base $\mathcal{K}$ being equal to $\mathcal{K}_a$ from the full argumentation setup. The knowledge base $\mathcal{K}$ then thus corresponds to either $\mathcal{K}_I$ or $\mathcal{K}_R$, depending on which agent is reasoning.

Arguments are of the form of inference trees. Observations from an agents knowledge base $\mathcal{K}$ are the starting point of arguments and they get expanded through the application of rules from $\mathcal{R}$. Arguments cannot be cyclic, this is enforced by forbidding the conclusion of an argument to appear in any of its sub arguments.

Definition 4.1.2 *(Argument, Inference, I). An argument is an inference $A_1....A_m \Rightarrow c$ such that $A_1....A_m$ is a, possibly empty, unordered set of arguments called its premises and $c \in \mathcal{L}$ is its conclusion. Valid inferences are generated by the inference function I. The inference function $I(AS_a)$ infers all arguments of an argumentation setup AS_a. There are two valid cases of inferred arguments:*

- $\emptyset \Rightarrow c \in I(AS_a)$ *iff* $c \in \mathcal{K}$.
- $A_1...A_m \Rightarrow c \in I(AS_a)$ *iff* $A_1....A_m$ *are in* $I(AS_a)$ *and their conclusions are* $c_1...c_m$*, and there is a rule* $c_1...c_m \Rightarrow c \in \mathcal{R}$*, and* c *does not occur in any of the arguments* $A_1...A_m$*.*

Arguments in the system can attack and defend each other. Argument $\mathcal{A}$ attacks argument $\mathcal{B}$ if the conclusion of $\mathcal{A}$ negates a conclusion of one of $\mathcal{B}$'s subarguments or $\mathcal{B}$'s own conclusion. A set of arguments $\mathcal{X}$ can defend an argument $\mathcal{A}$ when elements from that set attack each attacker of $\mathcal{A}$. Note that the system only supports rebutting attacks and does not support undercutting attacks. Arguments, additionally, cannot be attacked on premises and conclusions that are observations.

Definition 4.1.3 *(Attack, Defense). Let $AS_a = (\mathcal{L}, \mathcal{R}, \mathcal{Q}, \mathcal{K}, \tau)$ be an argumentation setup. For two arguments A,B $\in I(AS_a)$ we say that A attacks B iff A's conclusion is c and $\neg c$ occurs in B and $\neg c \notin \mathcal{K}$. A set of arguments $X \subseteq I(AS_a)$ defends an argument $A \in I(AS_a)$ iff for each $B \in I(AS_a)$ that attacks A there is a $C \in X$ that attacks B.*

Acceptability of arguments is determined by Dung's grounded semantics [6]. The grounded extension is a conflict-free and complete set of arguments.

Definition 4.1.4 *(Grounded Extension, G). Let $AS_a = (\mathcal{L}, \mathcal{R}, \mathcal{Q}, \mathcal{K}, \tau)$ be an argumentation setup. The grounded extension $G \subseteq I(AS_a)$ of AS_a is the smallest set of arguments (w.r.t. set inclusion) such that:*

- *There is no pair A,B $\in$ G such that A attacks B*
- $G = \{A \in I(AS_a) | G \text{ defends } A\}$

4.2 Participants

Dialogues are performed by agents. There are two participating agents in dialogues generated by our system with the following roles: *inquirer* $\subseteq \mathcal{I}$, *respondent* $\subseteq \mathcal{I}$ where $\mathcal{I}$ is the set of participants.

Definition 4.2.1 *(Participants, $\mathcal{I}$). The set of participants $\mathcal{I}$ of a dialogue. A participant can have one of the following two roles: inquirer $\subseteq \mathcal{I}$, respondent $\subseteq \mathcal{I}$.*

The *inquirer* tries to form an opinion on the topic by combining their knowledge on the topic and the results of questions they ask the *respondent*. The knowledge base $\mathcal{K}_I$ of the *inquirer* consists of observations that form a consistent set of question results. The *inquirer* expands this knowledge base by asking the *respondent* questions. These resulting observations are non-defeasible and thus, as a result, cannot be challenged by other knowledge. The *inquirer* and the respondent both have full knowledge of the set of defeasible rules $\mathcal{R}$ and of the set of queryable literals $\mathcal{Q}$. The *respondent* starts with knowledge regarding each of the queryable literals. This way, the *respondent* is able to answer each question asked by the *inquirer* by looking in its knowledge base $\mathcal{K}_R$ and extracting the corresponding answer to the question. Note that for the *inquirer* it does not matter how the *respondent* generates the answers to the questions its poses, as long as it will give an answer.

4.3 Moves

To determine what moves agents can make during the dialogue, we first have to define a communication language.

Definition 4.3.1 *(Communication Language, L_c). The communication language L_c for dialogues is a set of locutions. This set consists of the locutions {claim, question}, with as their contents a literal φ, where:*

- *claim φ is the assertion that $\varphi \in \mathcal{L}$ is true, made by the speaker.*
- *question φ is the act in which the speakers asks the other participant's opinion on if $\varphi \in \mathcal{Q}$ is the case or not.*

Now we can define a move in this dialogue system to be as follows:

Definition 4.3.2 *(Move). A move is a tuple $m = \langle x, l, c \rangle$ where:*

- *$x \in$ {inquirer, respondent} is the agent making the move.*
- *$l \in L_c$ is the type of locution being made.*
- *c is the content of the locution of the move.*

When a locution s is made in move m with as its content φ, the argumentation setup is updated. A *claim* move, which can only be made by the *respondent* agent, adds the claimed observation to the knowledge base $\mathcal{K}_I$ of the *inquirer*. Formally: if s(m) = $claim(\varphi)$ then $\mathcal{K}_I \cup \{\varphi\}$. Note that the *inquirer* thus always accept claims made by the *respondent*. This corresponds to the credulous acceptance attitude as defined by Parsons et al. [16], which says that an agent accepts any proposition as long as it is backed by an argument. Since an observation by itself is a valid inferred argument in our system, the claim of an observation will always be accepted by the *inquirer* agent.

4.4 Dialogue

Having defined the argumentation setup, participating agents and moves, we can now define exactly what a dialogue is.

Definition 4.4.1 *(Dialogue, D). A dialogue is a sequence of moves[m_1...m_n] involving two participants in $\mathcal{I}$ = {inquirer, respondent} where $n \in \mathbb{N}$ and $1 \leq n$, such that:*

- *the first move of the dialogue, m_1, is a move of the form of $\langle$inquirer, l, c$\rangle$*
- *the sender of move m_i is not the same as the sender of move m_{i+1}*
- *the nth move of the dialogue is denoted as d[n]*
- *if d is assumed to be n moves long, then $d = d_n$*

Dialogues in this system are unique reply, unique move and have alternating turns. Turn taking is decided by the turn taking rule T.

Definition 4.4.2 *(Turn Taking Rule, T). The turn taking rule, $T : D \to \mathcal{I}$, determines whose turn at a certain point in the dialogue d_n with length n it is. Let $l \in L_c$ be the locution of the last made move and c be its content:*

$$T(d_n) = \begin{cases} inquirer \text{ if } n = 0 \\ inquirer \text{ if } d_n[n] = \langle respondent, l, c \rangle \\ respondent \text{ if } d_n[n] = \langle inquirer, l, c \rangle \end{cases}$$

The outcome of a dialogue in this system is determined by an argument for the topic τ being present in the grounded extension G of the *inquirers* argumentation setup AS_I. If one or more such arguments exist, the outcome is a tuple containing the topic τ and a set containing all such arguments.

Definition 4.4.3 *(Outcome, O). The outcome O of an inquiry dialogue d_n with the inquirers argumentation setup AS_I is defined as follows, where $A^t \subseteq I(AS_I)$ is the set of all arguments with as its conclusion the dialogue topic τ and where G is the grounded extension of AS_I:*

$$O(d_n) = \begin{cases} (null, \emptyset) \text{ if } A^t = \emptyset \\ (\tau, G \cap A^t) \text{ otherwise} \end{cases}$$

4.5 Protocol

Dialogues have to be formed according to a dialogue protocol. The protocol Π is a function that, given the current dialogue and the agent whose turn it is, returns a set of legal moves for that agent to make. The dialogue topic τ is decided before the dialogue starts. The *inquirer* can always make a *question* move containing a literal from the set of queryables $\mathcal{Q}$, which has not been questioned before in the current dialogue. The *respondent* has to answer question moves made in the previous turn by looking up the answer to that question in his knowledge base $\mathcal{K}_R$ and making a *claim* move containing that answer, being either positive or negative. If the *respondent* has no information about the questioned literal, the answer given will give a negative reply for the questioned literal.

Definition 4.5.1 *(Protocol, Π). The inquiry protocol is a function $\Pi : D \times \mathcal{I} \rightarrow \mathcal{P}(M)$. Let d_n be a dialogue of n moves in which $d[n]$ was a move made by the other participating agent $\hat{x}$, with the current argumentation setup $(\mathcal{L}, \mathcal{R}, \mathcal{Q}, \mathcal{K}_I, \mathcal{K}_R, \tau)$, then $\Pi(d_n, x)$ is:*

$$\Pi^{question}(d_n, x) \cup \Pi^{claim}(d_n, x)$$

where

$$\begin{gathered}\Pi^{question}(d_n, x) = \{\langle x, question, c\rangle | \\ c \in \mathcal{Q} \setminus \{k | k = p \wedge (p \in \mathcal{K}_I \vee \neg p \in \mathcal{K}_I)\}, \\ x = inquirer\}\end{gathered}$$

and

$$\begin{gathered}\Pi^{claim}(d_n, x) = \{\langle x, claim, c\rangle | \\ d_n[n] = \langle \hat{x}, question, q\rangle, \\ (c = q \wedge c \in \mathcal{K}_R) \vee c = \neg q, \\ x = respondent\}\end{gathered}$$

The purpose of dialogues generated using this protocol is for the *inquirer* to form an opinion on the dialogue topic, using information from their own knowledge base $\mathcal{K}_I$ and that of the *respondent* $\mathcal{K}_R$. A dialogue terminates when the *inquirer* is at turn, but there are no legal question moves left for him to perform.

Definition 4.5.2 *(Termination). A dialogue is terminated when the following statement holds: $T(d_n) = inquirer \wedge \Pi(d_n, inquirer) = \emptyset$.*

An example of a well-formed inquiry dialogue, a dialogue of which all moves are legal according to its dialogue system, is shown in the table below.

t	$d[t]$	$\mathcal{Q}'$	$\mathcal{K}_I$	$\mathcal{K}_R$
0		{p, cp, w }	{}	{p, ¬cp, w}
1	⟨inquirer, question, p⟩	$\{cp, w\}$		
2	⟨respondent, claim, p⟩		$\{p\}$	
3	⟨inquirer, question, cp⟩	$\{w\}$		
4	⟨respondent, claim, ¬cp⟩		{p, ¬cp}	
5	⟨inquirer, question, w⟩	{}		
6	⟨respondent, claim, w⟩		{p, ¬cp, w}	

Table 4.1: An example inquiry dialogue generated by the proposed system. Updates of the available set of queryables $\mathcal{Q}' \subseteq \mathcal{Q}$ and the knowledge bases $\mathcal{K}_I$ and $\mathcal{K}_R$ of the argumentation setup AS are shown in their respective columns.

4.6 Strategy

For the *inquirer* to be able to perform dialogues, a strategy must be defined that picks what move the agent should make.

Definition 4.6.1 *(Strategy). A strategy is a function $D \rightarrow M$ that, given a list of legal moves in a dialogue, selects one of those moves to make.*

The most basic strategy for the proposed system would be an exhaustive strategy. The *inquirer* will perform the first *question* move from the set of legal moves given by the protocol, until the protocol returns an empty list of legal moves. Having fully exhausted the set of queryables, it can now easily be inferred if the knowledge base $\mathcal{K}_I$ in combination with the defeasible rules $\mathcal{R}$ are either in support of the dialogue topic or not. First we need to define a function that picks the first *question* move from the set of legal moves given by the protocol. The following picking function, and all future picking functions, are inspired by the type of picking functions defined by Black and Hunter [5] for their inquiry system. The defined strategies also draw inspiration from the definitions of Black and Hunter.

Definition 4.6.2 *($pick_{q,exh}$).*
Let $Q = \{\langle inquirer, question, c_1\rangle, ..., \langle inquirer, question, c_j\rangle\}$ be the set of legal question moves that the agent could make according to the protocol. The function $pick_{q,exh}$ returns the selected question move to make. $pick_{q,exh}(Q) = \langle inquirer, question, c_1\rangle$.

Given the current dialogue and the function $pick_{q,exh}$ that picks a legal *question* move from the set of all legal *question* moves given by the protocol, we can now define the exhaustive strategy as follows:

Definition 4.6.3 *(Exhaustive Strategy, S_E). The exhaustive strategy for an inquirer agent x in a dialogue d_n, with the current set of legal question moves Q, is a function $S_E : D \rightarrow M$. Let $T(d_n) = x$, then:*
$S_E(d_n, x) = pick_{q,exh}(Q)$

The strategy that the respondent follows is trivial. Since the protocol only returns one possible move for the respondent in his turn, the respondent simply plays the single *claim* move that answers the question asked in the previous turn. The protocol only returns one legal *claim* move after each question because of constraints that prevents the knowledge base $\mathcal{K}_R$ from having contradicting literals.

Definition 4.6.4 *(Respondent Strategy, S_r). The respondent strategy for a respondent agent is a function $S_r : D \rightarrow M$. Let $T(d_n) = x$ and let $\langle x, claim, c\rangle$ be the single legal move returned by the protocol Π, then: $S_r(d_n, x) = \langle x, claim, c\rangle$.*

We now have defined everything we need for the exhaustiv version of our base system. This exhaustive base system can be used to perform initial experiments and to define and implement further extensions on.

4.7 Extensions of the base system

4.7.1 Future setups and stability

A more intelligent *inquirer* than the one described in the base system could make use of future argumentation setups in order to reason ahead. Future setups are argumentation setups where the knowledge base of the *inquirer* is extended with question results from the still available set of queryables. Testerink, Odekerken and Bex [23] explore the idea of future setups and the related notion of stability in order to be more efficient in inquiry. A state is stable when all of its future setups have the same status of the dialogue topic as the current state. We will use their ideas of a stability based inquiry system and further expand them with additional locutions and resulting strategies, aiming to reach a full-fledgded stability-based dialogue system. First, we must look at the definitions used by Testerink, Odekerken and Bex that describe the notions of future setups and stability.

Definition 4.7.1 *(Future Setups, F). Let $AS_a = (\mathcal{L}, \mathcal{R}, \mathcal{Q}, \mathcal{K}, \tau)$ be an argumentation setup. The set of all future setups F(AS_a) consists of all setups $(\mathcal{L}, \mathcal{R}, \mathcal{Q}, \mathcal{K}', \tau)$ such that $\mathcal{K} \subseteq \mathcal{K}'$.*

An example of how to calculate the future setups of an argumentation setup is shown below.

Example 4.7.1 *Consider an argumentation setup AS_a, where $\mathcal{K} = \{p, cp\}$ and where $\mathcal{Q} = \{p, cp, w, s\}$. The future setups F($AS_a$) then consist of the following argumentation setups with a knowledge base $\mathcal{K}'$, such that $\mathcal{K} \subseteq \mathcal{K}'$:*

1. $\{p, cp\}$
2. $\{p, cp, w\}$
3. $\{p, cp, \neg w\}$
4. $\{p, cp, s\}$
5. $\{p, cp, \neg s\}$
6. $\{p, cp, w, s\}$

7. $\{p, cp, w, \neg s\}$
8. $\{p, cp, \neg w, s\}$
9. $\{p, cp, \neg w, \neg s\}$

Having the set of all future setups, we now want the agent to be able to reason about the status of the topic in these future setups and if it could still change. One concept useful to the agent would be that of stability. When a state is stable, it means that the status of the topic is the same in all of its future setups. For the status we consider four different cases. Firstly, the case where there exists no argument for the topic. Secondly, the case where there exists an argument for the topic in the grounded extension. Finally, the cases where there exists an argument for the topic, but this argument is attacked from either within or from outside of the grounded extension.

Definition 4.7.2 *(Stability). An argumentation setup AS_a is stable iff for each setup AS' in F(AS_a) one of the following cases holds:*

- ***Unsatisfiable:*** *there is no argument for τ in I(AS'), or*
- ***In:*** *there is an argument for τ in the grounded extension of AS', or*
- ***Out:*** *there is an argument for τ in I(AS'), but but all arguments for τ are attacked by an argument in the grounded extension of AS', or*
- ***Blocked:*** *there is an argument for τ in I(AS') but not in the grounded extension of AS' and at least one argument for τ is not attacked by an argument from the grounded extension of AS'.*

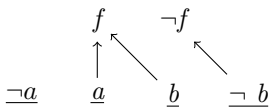
A simple strategy that uses the concept of future setups would be a strategy that only picks moves that change the status of the topic in the resulting argumentation setup. An approach that follows this line of thinking, but uses a different notion of stability than the one described by Testerink, Odekerken and Bex labelling and described in this thesis, is that of Kumeling [14]. Kumeling explores improvements upon the exhaustive strategy defined by Black and Hunter [5]. One of the proposed strategies only makes a new move, if that move would change the status of the node of the dialectical tree, or if it would not affect the tree at all. This can thus be seen as a one-step look-ahead stability based strategy. A more elaborate strategy would consider all future setups, and would only pick a *question* move if the current argumentation setup is not stable. In such a stability based strategy, the agent would select that query from the set of queryables $\mathcal{Q}$ that has the highest probability of leading to a stable state and would stop performing the dialogue when it has reached a stable state. This is the approach used by Testerink, Odekerken and Bex [23]. They model the argumentation setup and the inquiry process using a MDP, for which they can learn a policy that minimizes the amount of moves made before reaching a stable state. We will use and slightly adjust their definitions to model a first version of a stability based system and policy ourselves, after which we will also explore different extensions of this first system. All stability based strategies

should in principle lead to shorter dialogues than exhaustive strategies, since exhaustive strategies play all the legal moves returned by the protocol. First we need to define what an MDP based on an argumentation should look like.

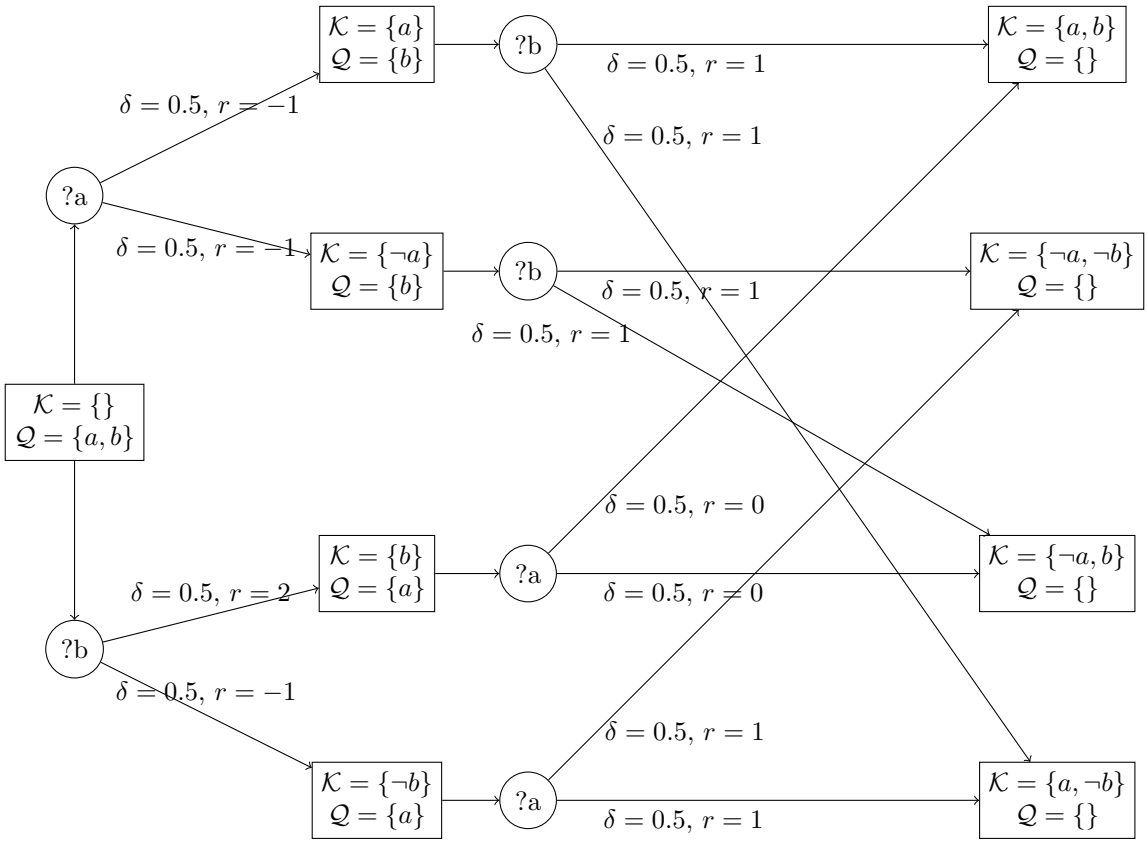
Definition 4.7.3 *(Argumentation MDP, MDP). An argumentation MDP for an argumentation setup AS_a is a tuple (Q,S,δ,r), where:*

- *Q is a set of questions*
- *$S = F(AS_a) \times 2^Q$ is the state space*
- *$\delta : S \times Q \times S \to [0, 1]$ is the transition function which returns the probability of the next state being $s_2 = (AS_2, Q_2) \in S$ given some state $s_1 = (AS_1, Q_1) \in S$ and question $q \in Q$. Furthermore, $\delta(s_1, q, s_2) = 0$ if the knowledge base of AS_1 is not a subset of that of AS_2, or if q is not available ($q \notin Q_1$), or if $Q_2 \neq Q_1 \backslash \{q\}$*
- *$r : S \times S \to \mathbb{I}$ is the reward function transitioning from $s_1 = (AS_1, Q_1) \in S$ to $s_2 = (AS_2, Q_2) \in S$ and is given by: $r(s_1, s_2) = |Q_1|$ if AS_2 is a stable setup, but AS_1 is not, $r(s_1, s_2) = 0$ if AS_1 already was a stable state, else $r(s_1, s_2) = -1$.*

Example 4.7.2 *Consider the following simple argumentation setup with two queryable literals a and b and three rules $a \Rightarrow f$, $b \Rightarrow f$ and $\neg b \Rightarrow \neg f$ where f is the topic.*



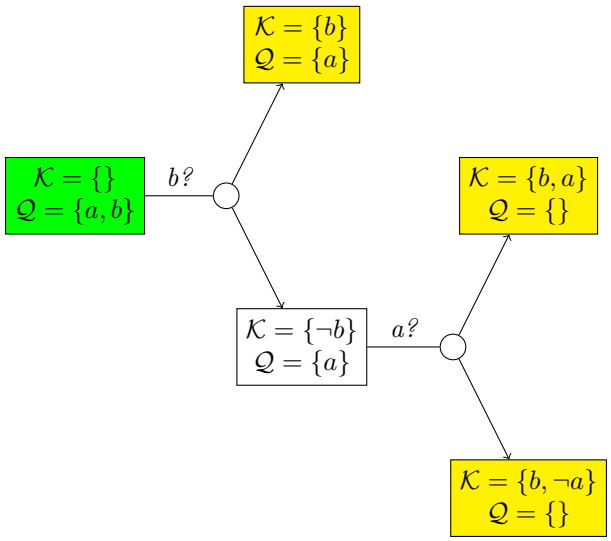
An argumentation MDP based on this argumentation setup, where states are represented by rectangles and actions represented by circles, would then look as follows:



Having this MDP, we can now define a policy π that chooses that action, which has the highest probability of leading to a stable state as fast as possible by maximizing the expected utility over the MDP.

Definition 4.7.4 *(Stability Policy, π). The policy $\pi : S \to Q$ for a MDP is given by $\forall_{s1} = (AS_1, Q_1) \in S : \pi(s_1) = argmax_{q \in Q_1} \Sigma_{s_2 \in S} \delta(s_1, q, s_2)(r(s_1, s_2) + V(s_2))$ where $V(s_2) = \Sigma_{s_3 \in S} \delta(s_2, \pi(s_2), s_3)(r(s_2, s_3) + V(s_3))$*

Example 4.7.3 *Consider the following policy, as an example of a policy calculated for the MDP of the previous example. The start state is indicated with a green colour and the final stable states are indicated with a yellow colour.*



Since it is costly to calculate such a policy, especially when the argumentation setup is large in size, we use a learning method that approximates this function. This policy will be learned by the agent by using the Q-learning algorithm. The stability-based strategy then uses the learned policy π to pick the question q that has the highest probability of leading to a stable state, unless the argumentation setup is already stable or the set of queryables is depleted. The inquirer ends the dialogue when he has to perform a move in a stable setup.

Definition 4.7.5 *(Stability Based Strategy, S_{stable}). The stability based strategy for a dialogue, with argumentation setup AS_a and available queries Q is defined as follows:*
$S_{stable}(d_n, x) = \langle inquirer, question, \pi(AS_a, Q)\rangle$ *whilst AS_a is not stable and Q is not empty.*

4.7.2 Approximating stability

Calculating the stability of an argumentation setup is very costly computationally. Testerink, Odekerken and Bex [23] show that calculating stability by calculating grounded extensions for all future setups has a complexity of $O(2^{|\mathcal{L}|!})$. It is thus preferable to have a more efficient way of calculating stability than brute-forcing it. In the same paper, Testerink, Odekerken and Bex propose an approximation algorithm for stability that is sound, but not complete. This means that the setups that are marked as stable by the algorithm indeed are stable, but some states that are still marked as not stable might actually be stable already. Stability is calculated by labelling the rules and literals of an argumentation setup. When the dialogue topic has received a label, the argumentation setup is stable. Testerink, Odekerken and Bex prove in their paper that the labelling of an argumentation setup can be constructed in $O((|\mathcal{L}| + |\mathcal{R}|)^2)$, which

is considerably more efficient than the brute-force method. The following definition is a slight adaptation of their definition. Instead of the label D, we decided to use the label I, which stands for *In*.

Definition 4.7.6 *(Labelling, L). Let $AS_a = (\mathcal{L}, \mathcal{R}, \mathcal{Q}, \mathcal{K}, \tau)$ be an argumentation setup. A labelling L is a partial function that decorates literals and rules with a label from $\{U, I, O, B\}$. Literals l that are in $\mathcal{Q}$ but not observed $(l, \neg l, \notin \mathcal{K})$ are not labelled. For the other literals and the rules the labelling of AS_a is declaritively defined as follows:*

- ***Case U literal:*** *A literal $l \in \mathcal{L}$ is labelled U iff either: A) No rule exists for l and if $(\neg)l \in \mathcal{Q}$ then $\neg l \in \mathcal{K}$. B) There are rules for l and they are all labelled U.*
- ***Case U rule:*** *A rule $r \in \mathcal{R}$ is labelled U iff any of its antecedents is labelled U.*
- ***Case I literal:*** *A literal $l \in \mathcal{L}$ is labelled I iff either: A) $l \in \mathcal{K}$. B) There is a rule for l labelled I and $\neg l$ is not in $\mathcal{L}$. C) There is a rule for l labelled I and there are rules for $\neg l$ but they are all labelled U or O.*
- ***Case I rule:*** *A rule $r \in \mathcal{R}$ is labelled I iff all its antecedents are labelled I.*
- ***Case O literal:*** *A literal $l \in \mathcal{L}$ is labelled O iff either: A) There are rules for l and they are all labelled I, O or B and $\neg l$ is labelled I. B) There are rules for l of which at least one is labelled O and the rest is either labelled O or U.*
- ***Case O rule:*** *A rule $r \in \mathcal{R}$ is labelled O iff at least one antecedent is labelled O and the rest is labelled I, B or O.*
- ***Case B literal:*** *A literal $l \in \mathcal{L}$ is labelled B iff $l, \neg l \notin \mathcal{Q}$ and either: A) A rule for l and a rule for $\neg l$ is labelled I or B. B) There are rules for l of which one is labelled B and the rest is either labelled U, O or B.*
- ***Case B rule:*** *A rule $r \in \mathcal{R}$ is labelled B iff at least one antecedent is labelled B and the rest is labeled B or I.*

Example 4.7.4 *Consider the argumentation system from example 4.7.2. In the case where $\mathcal{K} = \{\neg a, b\}$ the argumentation setup would be labelled as follows: $U = \{\neg b, \neg f, a, (a => f), (\neg b => \neg f)\}$, $I = \{\neg a, b, (b => f), f\}$, $O = \{\}$, $B = \{\}$.*

4.7.3 Querying for defeasible literals

In the proposed base system, the *inquirer* agent is only allowed to make question moves regarding observations, which are non-defeasible. It would be an interesting addition to the system to allow the *inquirer* to also question the *respondent's* opinion on defeasible literals, in the form of their labels according to

the *respondent*. This way, the *inquirer* can ask the *respondent* for information regarding the dialogue topic without asking the *respondent* to share explicit observations. The system will have to be adjusted in multiple ways to support this new type of questioning. Firstly, the *respondent* agent will need to calculate a labelling for its personal argumentation setup so that they can use the labels from this labelling in their responses. Secondly, the communication language and protocol will have to be adjusted with new moves that question and claim the labels of defeasible literals. Finally, a data structure must be defined that allows the *inquirer* to store and retrieve the uttered literal-label combinations of the *respondent*.

In order for the *respondent* agent to be able to answer questions regarding defeasible literals with labels, the *respondent* has to calculate a labelling for its personal argumentation setup AS_R. The argumentation setup of the *respondent* contains knowledge about each of the queryable literals in its knowledge base $\mathcal{K}_R$, contrary to the knowledge base of the *inquirer* that starts empty. The *respondent's* argumentation setup thus starts with full knowledge of all observables and as a result of this is static and does not change during the dialogue. Since this argumentation setup is static, we only have to calculate a labelling once. This labelling, $L(AS_R)$, is calculated using the function defined in definition 4.7.6 and used by the *respondent* to reply to questions about defeasible literals. The literal-label combinations claimed by the *respondent* have to be stored in the knowledge of the *inquirer*. This is why we extend the knowledge of the *inquirer* with a so-called label store, which is stored in memory separately from the argumentation setup and keeps track of literals and their corresponding labels as uttered by the *respondent*. The label store of the *inquirer* starts out empty, and is extended when the *respondent* makes claim moves for defeasible literals by adding the uttered literal and its label to the label store. Since the *respondent* can only claim labels of which it knows that they are true, and thus exist in the calculated labelling $L(AS_R)$, the label store will always be a subset of this calculated labelling.

Definition 4.7.7 *(label store, LS). A label store, denoted as LS, is a data structure storing tuples (φ, l) of literals φ from the logical language $\mathcal{L}$ and their corresponding label l such that $LS \subseteq L(AS_R)$.*

Example 4.7.5 *An example of a label store containing two items is as follows:* $\{(a, U), (b, I)\}$.

In the systems that we propose in this thesis, the *inquirer* is the only agent that will have a label store. We thus do not have to indicate with a subscript to what agent the label store belongs. If the system were to be extended with a label store for both the agents, one could represent the two different label stores as LS_I and LS_R for the *inquirer* and the *respondent* respectively.

The communication language will have to be extended with locutions that allow for the questioning and claiming of labels of defeasible literals. Firstly, we had to decide whether we wanted to adjust the question locution to support defeasible questions as well, or if we implemented a new locution strictly for

questioning defeasible literals. For clarity's sake, we chose to implement a new separate locution for the questioning of defeasible literals called *questionLabel*.

Definition 4.7.8 *(questionLabel). The locution questionLabel φ is the act in which the speaker asks the other participant's opinion on the label of $\varphi \in \mathcal{L}$.*

Next, we will have to define a locution that allows the respondent to reply to such a *questionLabel* locution. This *claimLabel* locution is defined below.

Definition 4.7.9 *(claimLabel). The locution claimLabel (φ, l) is the act in which the speaker informs the other participant that in his argumentation setup the literal φ is labelled as l.*

When a *claimLabel* locution is uttered by the *respondent*, the knowledge of the *inquirer* is updated as follows: if $\mathrm{s(m)} = claimLabel(\varphi, \mathrm{l})$ then $LS \cup \{(\varphi, l)\}$. Note that the *inquirer* will accept any such utterance made by the respondent. There thus are no requirements to the content of the utterance made that make the *inquirer* accept or reject it, contrary to the three acceptance attitudes defined by Parsons et al. [16].

Having defined the two locutions needed to implement the questioning and claiming of defeasible literals, we can now define when the agents are allowed to make moves using these locutions. For this we will give a definition of the new sub-protocols, which are to be added to the original protocol of definition 4.5.1.

Definition 4.7.10 *(Protocol, Π). The inquiry protocol is a function $\Pi : D \times \mathcal{I} \rightarrow \mathcal{P}(M)$. Let d_n be a dialogue of n moves in which $d[n]$ was a move made by the other participating agent $\hat{x}$, with the current argumentation setup $(\mathcal{L}, \mathcal{R}, \mathcal{Q}, \mathcal{K}_I, \mathcal{K}_R, \tau)$ and label store LS, then $\Pi(d_n, x)$ is:*

$$\Pi^{question}(d_n, x) \cup \Pi^{claim}(d_n, x) \cup \Pi^{questionLabel}(d_n, x) \cup \Pi^{claimLabel}(d_n, x)$$

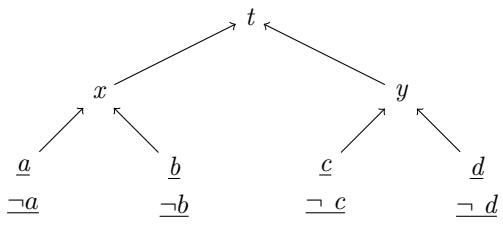
where

$$\begin{aligned}\Pi^{questionLabel}(d_n, x) = \{\langle x, questionLabel, c\rangle | \\ c \in \mathcal{L} \wedge c \notin \mathcal{Q} \wedge \neg c \notin \mathcal{Q} \wedge (c, l) \notin LS, \\ x = inquirer\}\end{aligned}$$

and

$$\begin{aligned}\Pi^{claimLabel}(d_n, x) = \{\langle x, claimLabel, (c, l)| \\ d_n[n] = \langle \hat{x}, questionLabel, q\rangle, \\ c = q \wedge ((c, l) \in L(AS_R) \vee l = Blank), \\ x = respondent\}\end{aligned}$$

Example 4.7.6 *Consider the following simple argumentation setup with four queryable literals a, b, c and d and six rules $a \Rightarrow x$, $b \Rightarrow x$, $c \Rightarrow y$, $d \Rightarrow y$, $x \Rightarrow t$ and $y \Rightarrow t$ where t is the dialogue topic.*



A dialogue that follows the newly defined protocol and uses a strategy that stops when a stable state has been reached would then look as follows:

t	$d[t]$	$\mathcal{Q}'$	$\mathcal{K}_I$	$\mathcal{K}_R$	LS
0		{a,b,c,d}	{}	{a,b,¬c,d}	{}
1	⟨inquirer, questionLabel, y⟩				
2	⟨respondent, claimLabel, (y, I)⟩				{(y,I)}
3	⟨inquirer, question, c⟩	$\{a, b, d\}$			
4	⟨respondent, claim, ¬c⟩		{¬c}		
5	⟨inquirer, question, d⟩	$\{a, b\}$			
6	⟨respondent, claim, d⟩		$\{\neg c, d\}$		

Table 4.2: An example inquiry dialogue generated by the proposed system. Updates of the available set of queryables $\mathcal{Q}' \subseteq \mathcal{Q}$, the knowledge bases $\mathcal{K}_I$ and $\mathcal{K}_R$ of the argumentation setup AS and the label store LS of the *inquirer* are shown in their respective columns.

In order for the *inquirer* agent to perform dialogues efficiently, we must update the earlier defined MDP of definition 4.7.3 to support the new questioning and claiming of labels, as well as observations. This way the *inquirer* agent can learn a policy to optimize dialogues using these new moves as well. Two changes will have to be made to the original MDP. Firstly, a state must now consist of an argumentation setup, a label store LS and the set of available actions in that state. Secondly, actions are no longer just questions regarding observables, but also questions regarding the labels of defeasible literals. These changes are formalised in the following new definition of an argumentation MDP.

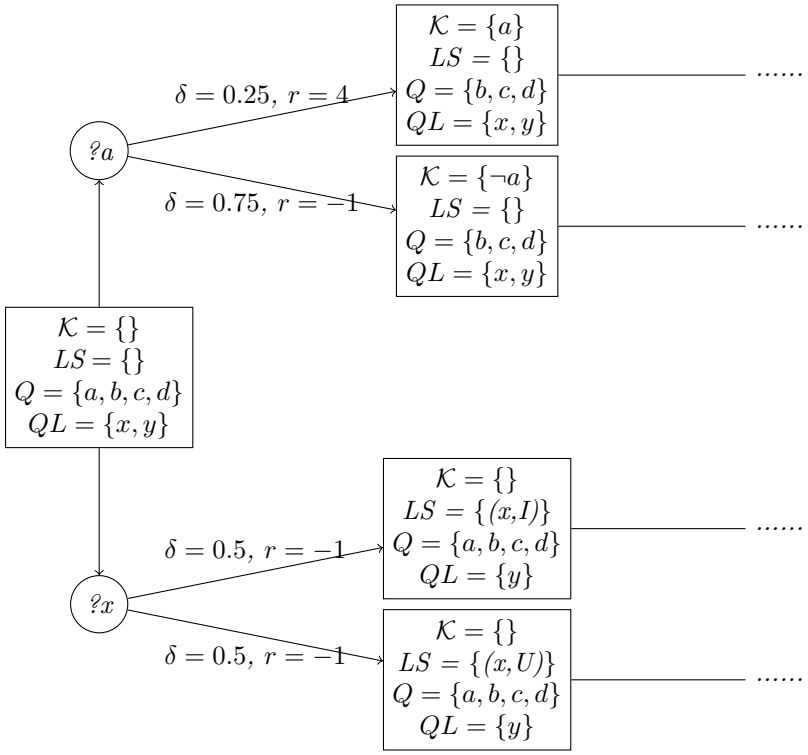
Definition 4.7.11 *(Argumentation MDP, MDP). An argumentation MDP where the knowledge of the agent is represented by an argumentation setup AS_a and a label store LS, is a tuple (A,S,δ,r), where:*

- *A is a set of actions containing question moves (Q) and questionLabel moves (QL).*
- *$S = F(AS_a) \times 2^Q \times 4^{QL}$ is the state space*
- *$\delta : S \times A \times S \to [0, 1]$ is the transition function which returns the probability of the next state being $s_2 = (AS_2, LS_2, A_2) \in S$ given some state $s_1 = (AS_1, LS_1, A_1) \in S$ and action $a \in A$. Furthermore, $\delta(s_1, a, s_2) = 0$ if the*

knowledge base of AS_1 is not a subset of that of AS_2 and LS_1 is not a subset of LS_2, or if a is not available ($a \notin A_1$), or if $A_2 \neq A_1 \backslash \{a\}$

- $r : S \times S \to \mathbb{I}$ *is the reward function transitioning from* $s_1 = (AS_1, LS_1, A_1) \in S$ *to* $s_2 = (AS_2, LS_2, A_2) \in S$ *and is given by:* $r(s_1, s_2) = |Q_1|$ *if* AS_2 *is a stable setup, but* AS_1 *is not,* $r(s_1, s_2) = 0$ *if* AS_1 *already was a stable state, else* $r(s_1, s_2) = -1$.

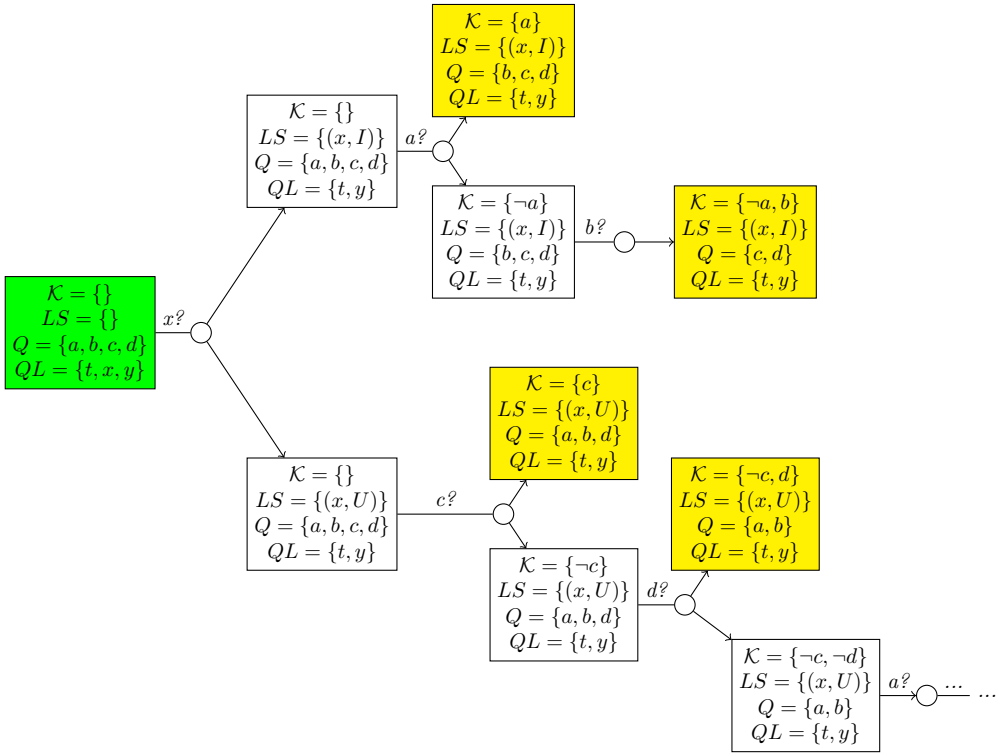
Example 4.7.7 *Consider again the argumentation setup from 4.7.6. A portion of the resulting MDP for this argumentation setup is shown below. Note that for clarity sake the MDP is simplified by only showing two possible actions, namely ?a and ?x that represent the question a move and the questionLabel x move. The resulting states of these actions are shown, but further future states are omitted. We also assume that the respondent has knowledge on all the observables, resulting in the only possible labels in this specific argumentation setup being I and U.*



Having defined this MDP, the *inquirer* can now learn a policy that chooses that action,which has the highest probability of leading to a stable state with as little made moves as possible by maximizing the expected utility over the MDP.

Example 4.7.8 *An example of a policy calculated for the system including questions for defeasible literals. The start state is indicated with a green colour*

and the final stable states are indicated with a yellow colour. Question moves for a and b on the bottom branch of the policy are omitted for brevity's sake, but follow the same logic as the other similar question moves for observables.



The *inquirer* agent in this system uses an adaption of the earlier defined stability based strategy, in which the policy returns the type of the action, *question* or *questionLabel*, and the literal that is the subject of the question.

Definition 4.7.12 *(Stability Based Strategy, S_{stable}). The stability based strategy for a dialogue, with argumentation setup AS_a and available actions A is defined as follows:*

$$S_{stable}(d_n, x) = \begin{cases} \langle inquirer, question, \pi(AS_a, A)\rangle \text{ if } \pi(AS_a, A) \in Q \\ \langle inquirer, questionLabel, \pi(AS_a, A)\rangle \text{ if } \pi(AS_a, A) \in QL \\ end\ dialogue\ otherwise \end{cases}$$

4.7.4 Asking explanations for labelled literals

Labels uttered by the *respondent* give insight in the knowledge and the point of view of the *respondent* and can be used to determine what questions would be more fruitful than others. It would be interesting to, additionally, give the *inquirer* agent means to ask the *respondent* to give an explanation, in the form

of an argument, for a literal-label combination that has been uttered during the dialogue and thus exists in the label store LS. The labels that are of interest are those that are supported by arguments. We will thus give the *inquirer* the means to ask for explanations of literals that are labelled as either I, O or B and we will not consider literals that are labelled as U. To allow the agents to express this kind of dynamic, new locutions need to be introduced and the dialogue protocol has to be extended.

Firstly, we must define a locution that allows the *inquirer* agent to question why a certain literal is labelled as I, O or B in the knowledge base of the respondent. For this purpose, we define a so-called *why* locution, which is inspired by the *challenge* locution as defined by Parsons et al. [16]. A minor difference is, however, that our *why* locution also allows the agent to ask arguments for why a literal could be labelled as O or B, whereas the *challenge* locution only asks the other agent to give an argument that is in support of a given proposition.

Definition 4.7.13 *(why). The locution why(φ, l) is the act in which the speaker asks the other participant's opinion on why the label of $\varphi \in LS$ is $l \in \{I, O, B\}$.*

Example 4.7.9 *Consider a situation where the inquirer has the tuple (y, I) in their label store LS as a result of an earlier questionLabel move. This allows the inquirer to make a why(y, I) move, resulting in the respondent having to give an explanation for why y is considered to be labelled I.*

We now must define a locution that allows the *respondent* to reply to such *why* moves. For this purpose we will now define a *since* locution, which allows the *respondent* to respond to *why* moves with an explanation in the form of an argument. This *since* locution is inspired by other *since* locutions found throughout literature such as Prakken [17] that allow agents to utter arguments in support of a proposition. Explanations consist of rules and their labelled premises that make a literal have the claimed label. For each of the three possible labels that can be explained using such a move, I, O and B, a different method is used to define a valid explanation. We will now define valid explanations for each of these three labels. These definitions of explanations are based on the definitions of the labels as given in definition 4.7.6.

Definition 4.7.14 *(explanation). An explanation is a tuple containing a set of rules R and a set of antecedents A that together adhere one of the following cases, dependent on the label being explained:*

- ***I explanation:*** *a valid explanation for a literal x being labelled I is a rule with as consequent x, which is labelled I, and its labelled antecedents that make it be labelled I.*
- ***B explanation:*** *a valid explanation for a literal x being labelled B is either A) a rule for x and a rule for $\neg x$ that are both either labelled I or B and their labelled antecedents that make them labelled I or B . B) one or more rules for x that are labelled B and their labelled antecedents that make them be labelled B.*

- ***O explanation:*** *a valid explanation for a literal x being labelled O is either A) a rule for the negation of x that is labelled I and its antecedents that make it labelled I B) the literal $\neg x$ being labelled I and the antecedents that make it labelled I. C) one or more rules for x that are labelled O and the antecedents that make them labelled O.*

Having defined what a valid explanation is, we can now define the *since* locution, which allows the *respondent* to utter such an explanation.

Definition 4.7.15 *(since). The locution since(R, A) is the act in which the speaker gives a reasoning for why the literal φ is labelled $l \in \{I, O, B\}$, based on a set of related rules R and a set of labelled antecedents of these rules A.*

Example 4.7.10 *Consider the earlier example where the inquirer makes a why(y, I) move. A possible response to this question would then be a since($\{c \Rightarrow y\}$, $\{(c, I)\}$) move.*

When a *why* move is uttered, the knowledge base of the *inquirer* has to be updated accordingly. For each of the uttered antecedents in the explanation, it must be determined whether this literal is defeasible or non-defeasible. The agent can determine this by looking at the queryables and the rules of the argumentation setup AS. Each non-defeasible antecedent will be added to the knowledge base $\mathcal{K}_I$ of the *inquirer*, positively if the label is I and negatively if the label is O. Each defeasible antecedent will be added to the label store LS with its related label. Formally: for each $\phi \in A$, if ϕ is non-defeasible and labelled I: $\mathcal{K}_I \cup \phi$, if ϕ is non-defeasible and labelled O: $\mathcal{K}_I \cup \neg\phi$ and else: $LS \cup (\phi, l)$. Note that, as with the *claimLabel* locution, the *inquirer* will accept any such utterance containing an argument made by the *respondent*. There thus are no further requirements to the argument in the utterance made that make the *inquirer* accept or reject it. This behaviour corresponds to the credulous acceptance attitude defined by Parsons et al. [16], as did the acceptance of *claim* moves by the *inquirer* defined earlier in this thesis.

Next we must update our protocol with conditions that allow the uttering of *why* and *since* moves. *Why* moves can be made for each literal in the label store LS that is labelled I, O or B and *since* moves can be made when the previous move of the dialogue is a *why* move. We now give additional protocol rules, which are to be added to the protocol defined earlier in definition 4.7.10.

Definition 4.7.16 *(Protocol, Π). The inquiry protocol is a function $\Pi : D \times \mathcal{I} \rightarrow \mathcal{P}(M)$. Let d_n be a dialogue of n moves in which d[n] was a move made by the other participating agent $\hat{x}$, with the current argumentation setup $(\mathcal{L}, \mathcal{R}, \mathcal{Q}, \mathcal{K}_I, \mathcal{K}_R, \tau)$ and label store LS, then $\Pi(d_n, x)$ is:*

$$\Pi^{question}(d_n, x) \cup \Pi^{claim}(d_n, x) \cup \Pi^{questionLabel}(d_n, x) \cup \Pi^{claimLabel}(d_n, x) \cup \Pi^{why}(d_n, x) \cup \Pi^{since}(d_n, x)$$

where

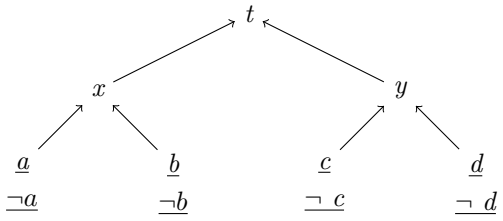
$$\Pi^{why}(d_n, x) = \{\langle x, why, (c, l)\rangle$$
$$(c, l) \in LS,$$
$$x = inquirer\}$$

and

$$\Pi^{since}(d_n, x) = \{\langle x, since, (R, A)|$$
$$d_n[n] = \langle \hat{x}, why, (c, l)\rangle,$$
$$(R, A) \textit{ is a valid explanation for } c \textit{ being labelled } l,$$
$$(\phi, l) \in L(AS_R) \textit{ for each } (\phi, l) \in A$$
$$x = respondent\}$$

An example of a dialogue formed according to this protocol, including a *why* and a *since* move, is shown below.

Example 4.7.11 *Consider the following simple argumentation setup with four queryable literals a, b, c and d and six rules $a \Rightarrow x$, $b \Rightarrow x$, $c \Rightarrow y$, $d \Rightarrow y$, $x \Rightarrow t$ and $y \Rightarrow t$ where t is the dialogue topic.*



A dialogue that follows the newly defined protocol and uses a strategy that stops when a stable state has been reached would then look as follows:

t	$d[t]$	$\mathcal{Q}'$	$\mathcal{K}_I$	$\mathcal{K}_R$	LS
0		{a,b,c,d}	{}	{a,b,¬c,d}	{}
1	⟨inquirer, questionLabel, y⟩				
2	⟨respondent, claimLabel, (y, I)⟩				{(y,I)}
3	⟨inquirer, why, (y,I)⟩				
4	⟨respondent, since, $(\{d \Rightarrow y\}, \{(d, I)\})$⟩	$\{a, b, c\}$	$\{d\}$		

Table 4.3: An example inquiry dialogue generated by the proposed system. Updates of the available set of queryables $\mathcal{Q}' \subseteq \mathcal{Q}$, the knowledge bases $\mathcal{K}_I$ and $\mathcal{K}_R$ of the argumentation setup AS and the label store LS are shown in their respective columns.

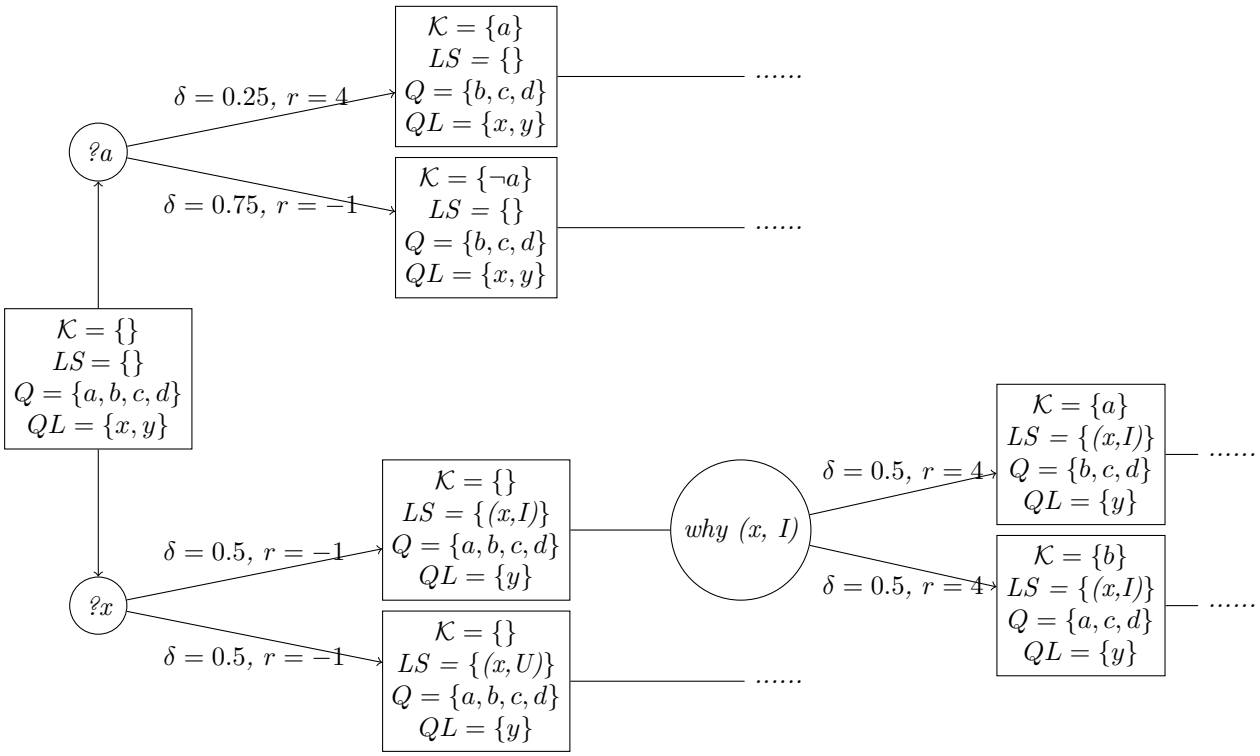
Having updated the dialogue protocol, we must now also update the MDP that the agent uses to calculate a policy. This MDP differs from the MDP defined

in definition 4.7.11 in the amount of actions that the agent has at its disposal. Besides the *question* moves and the *questionLabel* moves, the agent is now also allowed to make *why* moves. The amount of states stay the same, but the transitions towards these states increase due to the added actions. It now also is possible for an action to add multiple pieces of information to either the knowledge base $\mathcal{K}_I$ of the inquirer agent or the label store LS, so the transition function has to be loosened up slightly. This, for example, happens when the agent makes a *why* move for a literal and gets a response based on a rule with two antecedents, which are both to be added to the agents knowledge.

Definition 4.7.17 *(Argumentation MDP, MDP). An argumentation MDP where the knowledge of the agent is represented by an argumentation setup AS_a and a label store LS, is a tuple (A,S,δ,r), where:*

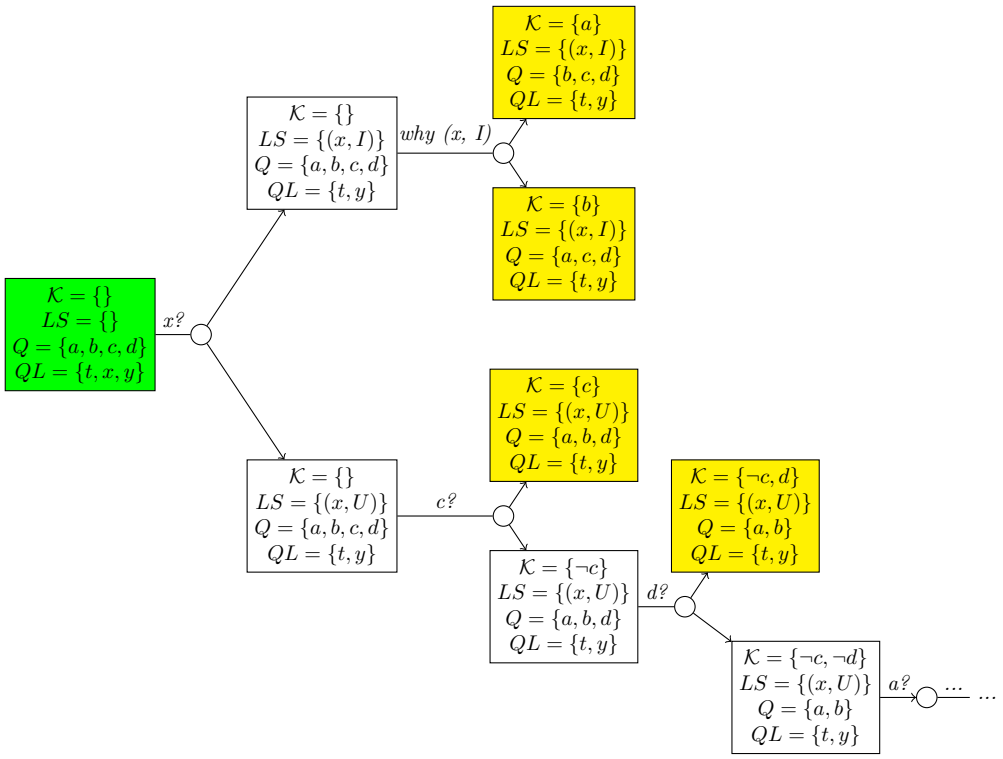
- *A is a set of actions containing question moves (Q),questionLabel moves (QL) and why moves (W).*
- *$S = F(AS_a) \times 2^Q \times 4^{QL}$ is the state space*
- *$\delta : S \times A \times S \to [0,1]$ is the transition function which returns the probability of the next state being $s_2 = (AS_2, LS_2, A_2) \in S$ given some state $s_1 = (AS_1, LS_1, A_1) \in S$ and action $a \in A$. Furthermore, $\delta(s_1, a, s_2) = 0$ if the knowledge base of AS_1 is not a subset of that of AS_2 and LS_1 is not a subset of LS_2, or if a is not available ($a \notin A_1$), or if $A_2 \not\subseteq A_1 \backslash \{a\}$*
- *$r : S \times S \to \mathbb{I}$ is the reward function transitioning from $s_1 = (AS_1, LS_1, A_1) \in S$ to $s_2 = (AS_2, LS_2, A_2) \in S$ and is given by: $r(s_1, s_2) = |Q_1|$ if AS_2 is a stable setup, but AS_1 is not, $r(s_1, s_2) = 0$ if AS_1 already was a stable state, else $r(s_1, s_2) = -1$.*

Example 4.7.12 *Consider again the argumentation setup from 4.7.11. A portion of the resulting MDP for this argumentation setup is shown below. Note that for clarity sake the MDP is simplified by only showing three possible actions, namely ?a, ?x and why (x, I) that represent the question a move, the questionLabel x move and the why x move. The resulting states of these actions are shown, but further future states are omitted. We also assume that the respondent has knowledge on all the observables, resulting in the only possible labels in this specific argumentation setup being I and U.*



Having defined this MDP, we can now learn a policy that chooses that action,which has the highest probability of leading to a stable state with as little made moves as possible by maximizing the expected utility over the MDP.

Example 4.7.13 *An example of a policy calculated for the system including questions for defeasible literals and the inclusion of why and since moves. The start state is indicated with a green colour and the final stable states are indicated with a yellow colour. Question moves for a and b are omitted for brevity's sake, but follow the same logic as the nodes for questions of c and d.*



The *inquirer* agent uses an adaption of the earlier defined stability based strategy, in which the policy returns the type of the action, *question*, *questionLabel* or *why* and the literal that is the subject of the question.

Definition 4.7.18 *(Stability Based Strategy, S_{stable}). The stability based strategy for a dialogue, with argumentation setup AS_a and available actions A is defined as follows:*

$$S_{stable}(d_n, x) = \begin{cases} \langle inquirer, question, \pi(AS_a, A)\rangle \text{ if } \pi(AS_a, A) \in Q \\ \langle inquirer, questionLabel, \pi(AS_a, A)\rangle \text{ if } \pi(AS_a, A) \in QL \\ \langle inquirer, why, \pi(AS_a, A)\rangle \text{ if } \pi(AS_a, A) \in W \\ end \text{ dialogue otherwise} \end{cases}$$

4.7.5 Calculating utility in argumentation MDP's

In the current versions of the three earlier defined MDP's, negative utility is added for each move made and positive utility is rewarded for each observable that did not have to be questioned before reaching a stable state. This can be seen as a hybrid utility function that aims to optimize a combination of dialogue length and minimal amount of observables shared. One could choose to only use one of these two factors in their utility function, thus minimizing either the dialogue length or the amount of observables shared.

Variations in calculating utility for each action and for the final state can also be thought of. Currently each action always costs exactly one utility unit, variations could be implemented where the three different kind of actions, *question*, *questionLabel* and *why*, would all have different costs, allowing for different dynamics to come forth. The utility of an action could also be made equal to the amount of observables that it shares. Another way in which we can vary the calculations of utility is by adjusting the positive utility rewarded when reaching a stable state. Different functions could be considered that include the size of the label store LS as well as the amount of shared observables or the final reward could be omitted entirely.

The most fitting utility function to use in a system depends on the domain in which it is deployed and the goal that it is trying to achieve. In a situation where the system is used only between agents and the dialogues regard sensitive information of different departments and organizations, it might be best to ignore the dialogue length in the utility function and only punish the agents for each observation that they share. In an environment where the agent has to deal with a human counterpart, the dialogue length becomes more important and intuitive to include in the utility function. In this situation one could choose for an utility function that always uses the dialogue length in the utility function and also gives negative utility for each observable shared, in order to not have the persons give more information than is absolutely necessary. If the information that has to be shared is not personal or private at all, it can be chosen to ignore the amount of observables shared in the utility function. The exact utility functions that we chose to use in our experiments will be defined in the next chapter.

Chapter 5

Experiments

In this chapter, multiple experiments will be performed in which we compare the performance of different versions of our stability-based inquiry dialogue system with each other and with an exhaustive strategy.

5.1 Utility functions

In our experiments we will use two different utility functions for the defined MDP's. The agents will then learn a policy that optimizes this specific type of utility, and use that policy to perform dialogues as effective as possible. The first utility function is straightforward: dialogue length. In this utility function, agents have to pay utility for each move they make, before reaching a stable state. When a stable state is reached, the agent gains a positive utility that is proportional to the size of the argumentation setup used, more specifically: the amount of queryable literals in $\mathcal{Q}$. This way, the agent will learn to minimize the dialogue length. This utility function is defined below, in the same form as the utility functions of the earlier defined MDP's, with states being represented by a tuple of an argumentation setup AS, a label store LS and a set of available actions A. This way, the utility function can easily be plugged into the MDP. All following utility functions are defined with this in mind.

Definition 5.1.1 *(Dialogue length utility function).* $r : S \times S \to \mathbb{I}$ *is the reward function transitioning from* $s_1 = (AS_1, LS_1, A_1) \in S$ *to* $s_2 = (AS_2, LS_2, A_2) \in S$ *and is given by:* $r(s_1, s_2) = |\mathcal{Q}|$ *if* AS_2 *is a stable setup, but* AS_1 *is not,* $r(s_1, s_2) = 0$ *if* AS_1 *already was a stable state, else* $r(s_1, s_2) = -1$.

The second utility function is aimed at privacy. When two agents of different organizations communicate with each other, they might have to share sensitive data which they would rather prevent from being shared if possible. Another scenario is that of a police intake conversation where a complainant might not want to give away more personal information than needed. In the case of the system proposed in this thesis, that would mean sharing more observables than

is absolutely needed to form a verdict. When this utility function is used, the agent gains positive utility for all observables that the respondent did not have to share when reaching a stable state and gains direct negative utility for each observable that is being shared as a result of an action. This utility function is defined below in the same manner as earlier utility functions.

Definition 5.1.2 *(Observables shared utility function).* $r : S \times S \to \mathbb{I}$ *is the reward function transitioning from* $s_1 = (AS_1, LS_1, A_1) \in S$ *to* $s_2 = (AS_2, LS_2, A_2) \in S$ *and is given by:* $r(s_1, s_2) = |Q_1|$ *if* AS_2 *is a stable setup, but* AS_1 *is not, where* $Q_1 \subseteq A_1$ *is the set of question moves in* A_1*,* $r(s_1, s_2) = 0$ *if* AS_1 *already was a stable state, else* $r(s_1, s_2) = |AS_1(\mathcal{K}_I)| - |AS_2(\mathcal{K}_I)|$.

5.1.1 Instantiating

Different rules of instantiating can be considered for the test cases during the learning of the policy and the knowledge base of the *respondent* $\mathcal{K}_R$. In all cases we assume that the *inquirer* starts with no knowledge about any of the observables, and the *respondent* starts with knowledge about all the observables. We can then define how we want to instantiate the knowledge base of the *respondent*, according to one of the following two functions:

- **Random**: each observable is randomly instantiated as true or false.
- **Unique argument**: the observables are instantiated in such a way that always one and only one rule for the topic is labelled *I*.
- **Conflicting argument**: the observables are instantiated in such a way that there always exists an argument for the topic and there may be an argument attacking this argument, resulting in the topic always being either *I*,*B* or *O*.

We will use the random instantiation function for all of the the cases we will define later on. Additionally, for the test cases in which there exist no attacks on possible arguments in support of the topic, we will use the unique argument instantiation function. Finally, we will use the conflicting argument instantiation function for the setups in which attacks on arguments in support of the topic can occur.

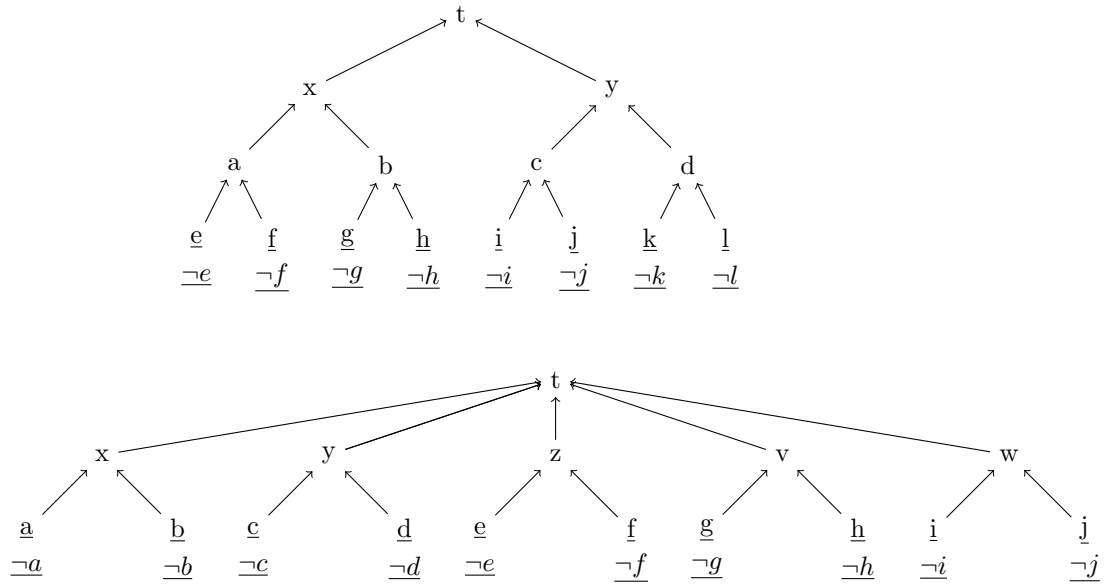
5.2 Test classes

In this section we will define the test cases on which we will run our experiments and explain why they have been chosen.

5.2.1 Standard tree

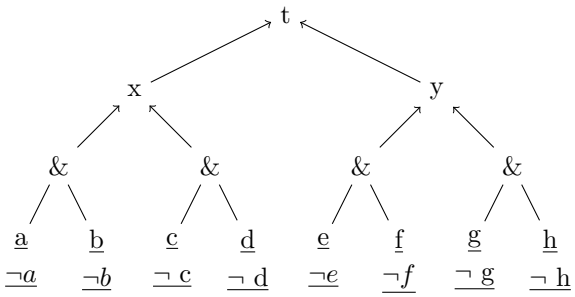
The standard tree is the most straightforward tree we can use to test the agent and policy on. The standard tree consists of observations and rules with 1

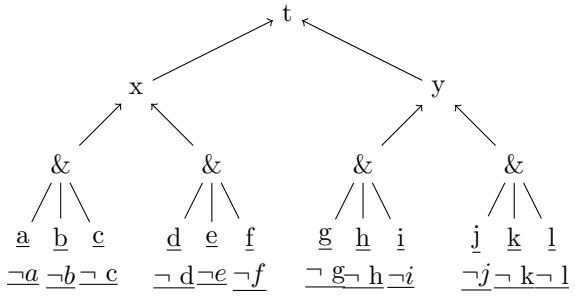
antecedent. The standard tree contains no conflicting rules and possible arguments. This tree can be generated with different widths and depths, where the width indicates the amount of subtrees under the root of the tree and the depth indicates the amount of layers of children the root node of the tree has. We will do experiments on two types of standard trees: one deep tree that primarily has long arguments and one wide tree that has short arguments, but more arguments in number. Both setups are shown below, first the deep tree and then the wide tree. The deep tree has 8 observable literals and 14 rules. The wide tree has 10 observable literals and 14 rules.



5.2.2 Multi-rule tree

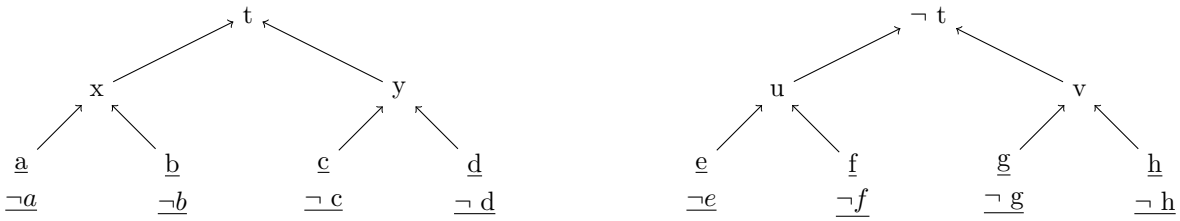
The multi-rule tree adds a layer of complexity to the standard trees. In multi-rule trees, each layer beyond the first layer of children of the root consists of rules with multiple antecedents. The amount of antecedents can be customized. For our experiments we will use two setups: one setup in wich all rules have two antecedents and one setup in which all rules have three antecedents. The two argumentation setups are shown below. The setup with two antecedents has 8 observables and 6 rules. The setup with three antecedents has 12 observables and 6 rules.



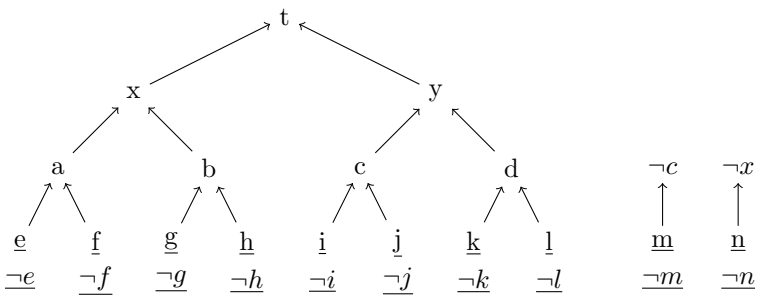


5.2.3 Ambiguity

These setups are meant to determine the ability of the systems to deal with setups containing sources of ambiguity. The inspiration of these setups come from the experiments of Kumeling [14], who also tested strategies on a set of setups that were meant to monitor behaviour in ambigiuous situations. Firstly, the negation tree setup consists of two trees. One tree that is in favour of the dialogue topic and one tree that is attacking the dialogue topic. Both threes can be considered as any of the earlier determined tree types, and the amount of observables is thus given by the sum of the two individual trees. The specific negation tree setup chosen for the experiments has 8 observables and 12 rules.



Secondly, we defined an ambiguous setup where the tree in support of the topic is attacked by multiple arguments on different levels within the tree. The thought behind this setup is that we can determine the ability of the system to deal with multiple possible attacks on multiple levels at the same time. This setup contains 10 observables and 16 rules.



5.2.4 Complex scenarios

As the final test, we have two complex scenarios that combine all the earlier defined features. These scenarios are large in size and contain rules with multiple antecedents and multiple possibilities for conflicts happening. The first complex scenario is a handpicked argumentation setup that aims to combine all these features. This setup contains 16 observables and 12 rules. The fraud intake scenario is an argumentation setup based on a rule set used in a real life scenario. This scenario is vast in size and should be a good benchmark for how the systems would perform in real life. This setup contains 23 different queryables, contains 51 rules and has one topic. Because of the sheer size of these setups and the difficulty of compactly depicting them in a graph, we do not show their layouts here. Instead, both of the rule sets are shown in appendix A. In these tables, the queryable literals are underlined and the topic literal is in bold font. Both setups contain rules containing multiple antecedents, have attacks between rules on different levels of the setup and are thus an ideal combination of all factors that the system should be able to deal with. The challenge in the fraud intake scenario will most likely be in the learning of the policy. The adapted MDP's have a larger complexity than the base MDP defined by Testerink, Odekerken and Bex [23], with large setups this may become a problem, even when using approximation techniques such as q-learning.

5.3 Systems

In our experiments we will compare the performances of four different systems. Currently, exhaustive options such as Black and Hunter [5], are the de facto standard. These systems, however, also can serve as a worst case scenario comparison, since each move will have to be made before closing the dialogue. For this reason, we have included an exhaustive option in our experiments that simply makes a *question* move for each of the observable literals. This system is denoted as EXH in the results. The second system we include in our experiments is a stability-based system. We had the choice of including a one-step stability based system such as that of Kumeling [14], to include a full stability-based system such as that of Testerink, Odekerken and Bex [23] or both. We made

the decision to only include the base system of Testerink, Odekerken and Bex in the comparison. This system is denoted as TB in the results. The reason for this is twofold: firstly, research has already been done by Kumeling on the performance of the system. This research has shown that their strategies either did not have an impact on the dialogue length, or lost the properties of soundness and completeness. The second reason is the results that they found, in our comparison we want to compare systems that all are sound and complete. The final two systems we include in our experiments are extensions of the base Testerink, Odekerken and Bex based system. Firstly, the system that extends the base system of Testerink, Odekerken and Bex with a *questionLabel* locution and secondly, the system that extends this last system with an additional *why* locution. These systems are denoted as TB+?L and TB+?L+WHY in the results respectively. Having these four systems in place, we can determine the effectiveness of the stability-based strategy in general and the effects of the extensions on this stability-based system.

5.4 Results

In this section we will discuss the results of the experiments by looking at the impact of the additions to the base system on the dialogue length and the amount of observables shared. We will also look at how the different setups used impacted performance and how the different instantiation functions used impacted performance. Firstly, we will look at how the additions to the system have impacted the dialogue length in the different argumentation setups, when dialogue length is used as the performance metric whilst learning the policy. Secondly, we will look at how the additions to the system have impacted the amount of observables shared in the different argumentation setups, when that number is used as the performance metric whilst learning the policy. We will look at the findings for each of the different classes of argumentation setups, and where relevant we will look at the different types of instantiation functions used. The found results are presented in the form of box-plots, in which the line in the box indicates the median value of a series and the square indicates the average value of a series.

5.4.1 Dialogue length

5.4.1.1 Random instantiation

The first results we will look at are of the experiments done to minimize the dialogue length, using the random instantiation function as the generator for our test cases. This means the agent learns a policy based on cases that are fully randomly generated, and the test cases on which we measure the results are randomly generated in the same way. These results can be seen in appendix B.1.

In the two standard tree cases, as shown in appendix B.1.1, the results for all three non-exhaustive systems are comparable. For the most part, all three

systems learned similar policies and the *questionLabel* and *why* moves were not used much. The slight usage of those moves does bring forth small differences. It can be seen that the system with added *questionLabel* moves and the system with both added *questionLabel* and *why* moves have worse worst-case scenarios than the base Testerink, Odekerken and Bex system and even the exhaustive strategy. The median of the new systems, however, is lower in the deep tree example and the same in the wide tree scenario. The averages of the three systems are comparable on both setups.

In the Multi-Rule cases, as shown in appendix B.1.2, it can be seen that the system with added *questionLabel* moves performs about the same as the base system of Testerink, Odekerken and Bex, except for having some more outliers. The averages and the medians are similar enough to not warrant an improvement. The system with added *why* locution has comparable averages as the other two systems, but the medians are lower in both the 2 antecedent rule set and the 3 antecedent rule set. Something to note is that the worst case of the two new proposed system is worse than the exhaustive strategy.

In the ambiguous setups, as shown in appendix B.1.3, similar results to the earlier setups can be seen. The medians of the three systems are the same in both the setups. The averages, however, are higher for both the extensions of the base system in both of the setups. Outliers again are worse than the results given by an exhaustive strategy, as could be seen in the other classes of setups as well.

Finally, we will look at the two complex scenarios, of which the results can be seen in appendix B.1.4. The first setup, labelled as the complex scenario, shows similar results to what we have been seeing so far: the medians of the three systems are the same and the averages are comparable, where the system with only the *questionLabel* locution added has a slightly worse average. It can again be seen that the two extended systems have worse outlier cases than systems performing exhaustive strategies. The fraud intake scenario, shows significantly different results for the first time during the experiments. The base system of Testerink, Odekerken and Bex has the best performance with a low median and a low average. The two extended systems, however, perform very poorly in comparison and have extreme outlier worst-case scenarios. They perform only slightly better than the exhaustive option on average and the median is slightly better in both cases. The best case scenarios of the two extended systems also are worse than that of the base system. These poor results can be explained by the significantly larger size of this argumentation setup compared to previous experiments. The MDP and its related Q-values that are stored during the learning of the policy grow so large during the learning phase, that it would not fit into memory anymore. The systems are thus trained on the maximal amount of rounds that would still allow the MDP to fit into memory, which as can be deducted from the outcome of the experiment, is not enough to result in an efficient policy.

5.4.1.2 Unique argument instantiation

Secondly, experiments were performed using the unique argument instantiation function. These experiments are performed for the argumentation setups that do not contain any possible conflicting arguments and can thus always be instantiated with one single argument that makes the topic *In*. The agents learn a policy based on randomly generated cases in which always one and only one argument for the topic can be found. They are then tested on a set of such cases afterwards. The results of these experiments can be seen in appendix B.2.

Firstly, we will look at the two standard tree cases again, instantiated using the unique argument function. The results of these experiments can be seen in appendix B.2.1. The results from these experiments clearly differ from the results we have seen earlier for the fully random generated cases. The base system of Testerink, Odekerken and Bex has more trouble with this type of instantiation than the random instantation. The two extended systems however, especially the most complex system with *why* moves, adapt themselves better. In the deep tree setup, the system with added *questionLabel* move performs slightly better than the base system, in both average and median. In the wide tree setup the system with added *questionLabel* move performs exactly the same as the base system. This is a result of the learned policy not containing any *questionLabel* moves in general. The system with added *why* locution, however, clearly outperforms both of the other two systems in both the two tree scenarios. The average and median are the same value and there are very few outliers, if any. In the generated dialogues, it can be seen that the combination of *questionLabel* and *why* moves result in a policy that is similar to backward chaining algorithms. The agent starts by questioning the labels of literals that are the antecedent of a rule with as its consequent the topic, or in case of the wide tree the agent starts questioning the label of the topic itself, and when it finds an I label it works its way down the tree using *why* moves. If the label of one such antecedents is U, the agent knows that it will have to go down other branches of the tree. This results in a really consistent strategy that outperforms the strategies of the other two stability-based systems. An example of a commonly found dialogue for the deep tree that uses such a backward chaining like approach, is shown below in table 5.1.

t	$d[t]$
1	$\langle$inquirer, questionLabel, y$\rangle$
2	$\langle$respondent, claimLabel, (y, U)$\rangle$
3	$\langle$inquirer, questionLabel, b$\rangle$
4	$\langle$respondent, claimLabel, (b, I)$\rangle$
5	$\langle$inquirer, why, (b, I)$\rangle$
6	$\langle$respondent, since, $(\{(h \Rightarrow b)\}, \{(h, I)\})\rangle$

Table 5.1: An example inquiry dialogue on the deep tree setup, generated by the system with *questionLabel* and *why* moves.

The dialogue in table 5.1 shows that the agent starts out by asking the label of y, which is the antecedent of one of two rules that have as its consequent the dialogue topic t. By the response of the respondent, the label of y being U, the agent can deduct that it has to go down the tree of the other rule with its antecedent x. The agent then takes a guess at which of the two premises of rules for x would be I, similarly to what it did for the rules of t. The agent asks for the label of b and gets a result saying that the label of b is I. This label allows the agent to make a $why(b,\ I)$ move, resulting in h being added to $\mathcal{K}_I$ and a stable state being reached.

Now we will look at the two Multi-Rule scenarios again, but this time with the unique argument instantiation. The results of these experiments can be seen in appendix B.2.2. The results from these two setups are similar to the results shown earlier for the standard trees instantiated using the unique argument instantiation function. Again the base system and the system with added *questionlabel* locution show quite similar results. The system with the added *why* locution again outperforms both systems. The average is lower than the other two systems and the median is the same as that of the *questionLabel* system in the 2 antecedent setup, but lower in any other case. This again is a result of the backward chaining like behaviour, which could also be seen in the standard tree setups. An example of a commonly found dialogue for the Multi-Rule tree with three premises that uses such a backward chaining like approach, is shown below in table 5.2.

t	$d[t]$
1	⟨inquirer, questionLabel, x⟩
2	⟨respondent, claimLabel, (x, U)⟩
3	⟨inquirer, questionLabel, y⟩
4	⟨respondent, claimLabel, (y, I)⟩
5	⟨inquirer, why, (y, I)⟩
6	⟨respondent, since, $(\{(g, h, i \Rightarrow b)\}, \{(g, I), (h, I), (i, I)\})$⟩

Table 5.2: An example inquiry dialogue on the Multi-Rule with 3 antecedent setup, generated by the system with *questionLabel* and *why* moves.

The dialogue in table 5.2 shows that the agent starts out questioning the labels of literals that are the antecedent of a rule with as the consequent the topic t. When the agent sees the response that x is U, it knows that it must then ask the label of y, which it could already deduce as being I in a unique argument case. Having the respondent claim the label of y as I then allows the agent to make a why(y,I) move, resulting in the three literals g,h and i being uttered and a stable state being reached.

5.4.1.3 Conflicting argument instantiation

Finally, we performed experiments using the conflicting argument instantiation function. These experiments were performed on the argumentation setups that

contain possibly conflicting arguments. The cases generated using this function always have a status for the topic that is either *In*, *Out* or *Blocked*. The agents learn a policy based on these generated cases and then are tested on a set of similarly generated test cases. The results of these experiments can be seen in appendix B.3.

Firstly, we will look at the results of the two setups in the ambiguous class. The results of these experiments can be seen in appendix B.3.1. The experiments show similar results to that of the experiments using the random instantiation function. All three stability-based systems show comparable results, with a marginal preference for the base system. The two extended systems only minimally made use of their newly added locutions and thus all three stability-based systems had similar policies, resulting in comparable performance. The stability-based systems all heavily outperformed the exhaustive option in both two setups.

Secondly, we will look at the results of the two complex setups. These results can be seen in appendix B.3.2. These results show the same effect as the experiments done using the random instantiation function, but more extreme. In both the two setups, and especially the fraud intake scenario, the extended systems are outperformed by the base system. The extended systems even perform significantly worse than the exhaustive option in the fraud intake scenario. The effect of the large state space space hindering the learning of an optimal policy seems to be amplified by using the conflicting argument instantiation function. The base system, however, clearly outperforms both the two extended systems and the exhaustive option.

5.4.2 Observables shared

5.4.2.1 Random instantiation

We will now look at results from the experiments on minimizing the amount of shared observables during a dialogue, using the random instantiation function as the generator for our test cases. This means the agent learns a policy based on cases that are fully randomly generated, and the test cases on which we measure the results are randomly generated in the same way. These results can be seen in appendix C.1.

The results of the standard tree setups, which can be found in appendex C.1.1, already show that the added locutions are able to effectively minimize the amount of observables shared during a dialogue. The system with only the added *questionLabel* locution already shows a small decrease in the average amount of observables shared and the system with the *why* locution added as well seems to be very consistent in sharing the minimum amount of observables needed for the setup to become stable. Since the sole performance metric in these experiments was the amount of observables being shared, it could be seen that the extended system first tried to gather as much information on the defeasible literals as needed using *questionLabel* and *why* moves, before asking the other agent for observables. An example of a commonly found dialogue for

the deep tree that uses such an approach, is shown below in table 5.3.

t	$d[t]$
1	$\langle$inquirer, questionLabel, t$\rangle$
2	$\langle$respondent, claimLabel, (t, I)$\rangle$
3	$\langle$inquirer, why, (t, I)$\rangle$
4	$\langle$respondent, since, $(\{(x \Rightarrow t)\}, \{(x, I)\})\rangle$
5	$\langle$inquirer, why, (x, I)$\rangle$
6	$\langle$respondent, since, $(\{(b \Rightarrow x)\}, \{(b, I)\})\rangle$
7	$\langle$inquirer, why, (b, I)$\rangle$
8	$\langle$respondent, since, $(\{(g \Rightarrow b)\}, \{(g, I)\})\rangle$

Table 5.3: An example inquiry dialogue on the deep tree setup, generated by the system with *questionLabel* and *why* moves.

As shown above in table 5.3, the agent is able to limit the amount of shared observables to only 1 if there exists an argument for the topic t, using *why* and *questionLabel* moves. This is a great improvement over the base system, which is fully dependant on guessing what observables are labelled I based on past information and cannot gather any more information before questioning those observables. The system with only the added *questionLabel* locution can already outperform the base system by first gathering information using *questionLabel* moves, but in the final moves must still make a guess for what rules make a literal labelled I since it cannot make *why* moves.

The experiments done on the Multi-Rule setups, which can be found in appendix C.1.2, show similar results as that of the standard tree experiments. The added *questionLabel* locution alone already gives a small performance increase and this effect is further increased when the *why* locution is added as well. Generally the agent would use *questionLabel* moves to figure out what rule for the topic is labelled I, if any, and would then either question for the antecedents or perform a *why* move for the consequent of that rule. This behaviour results in lower averages and lower medians when measuring performance in the amount of observables shared. The strategy that the agent uses is similar to that shown earlier for the standard tree cases, but the results are slightly less positive for the extended systems. This is because of the random instantiation function that was used in these experiments. The chance for the topic being *In* is a lot higher in the case of standard trees than in Multi-Rule trees, since all the antecedents of the rule have to be generated positively by the instantiation function. This results in more cases being *Unsatisfiable* in the experiments done on the Multi-Rule trees and thus more observables having to be shared before concluding the dialogues and finding out that there might not be an argument.

The experiments done on the two setups of the ambigiuous group, which can be found in appendix C.1.3, again show similar results to previous experiments. It can be noted, however, that for the negation tree setup, just the added *questionLabel* locution does not provide a large performance increase and in the case of the multi-level attacks setup it even performs slightly worse than the base

system of Testerink, Odekerken and Bex. The *questionLabel* locution is able to find out in what parts of the setups ambiguity lies, but in itself it is not able to solve it more efficiently than the base system. The system with the additionally added *why* locution, clearly outperforms the other two systems. In case of the negation tree setup, this system is able to very consistently perform dialogues with just 2 observables shared. In case of the multi-level attack setup, all three systems have the same median but the average of the system with why locution is lower than the other two systems. An example of a dialogue on the negation tree setup, in which the system with *questionLabel* and *why* locution performs a backward chaining like behaviour for the positive tree and the negative tree is shown below in table 5.4.

t	$d[t]$
1	⟨inquirer, questionLabel, t⟩
2	⟨respondent, claimLabel, (t, B)⟩
3	⟨inquirer, why, (t, B)⟩
4	⟨respondent, since, $(\{(x \Rightarrow t), (u \Rightarrow \neg t)\}, \{(x, I), (u, I)\})$⟩
5	⟨inquirer, why, (x, I)⟩
6	⟨respondent, since, $(\{(b \Rightarrow x)\}, \{(b, I)\})$⟩
7	⟨inquirer, why, (u, I)⟩
8	⟨respondent, since, $(\{(f \Rightarrow u)\}, \{(f, I)\})$⟩

Table 5.4: An example inquiry dialogue on the negation tree setup, generated by the system with *questionLabel* and *why* moves.

The dialogue shown in table 5.4 shows that the agent starts out by asking the label of the topic t. When the agent gets a response that the topic is labelled B, he can perform a *why* move to retrieve a rule for the topic and a rule against the topic that caused this label. From there on, the agent uses *why* moves to retrieve the observations that lie on the start of the two opposing arguments.

Finally we will look at the two complex scenarios, of which the results can be found in appendix C.1.4. These two argumentation setups are larger than those of the other experiments. This can be seen in the results of the experiments, similar to what we have seen in the experiments on dialogue length. In the first complex scenario, the medians of all the three systems are the same. The averages differ slightly, but no significant differences can be seen between the three stability-based systems. The fraud intake scenario, contrary to as in the experiments on dialogue length, was decently learnable for the agents using the two extended systems. The agents again had to prematurely stop learning their policies, since the MDP and its related Q-values did not fit into memory anymore, but this time the performance of the agents actually already was slightly better than the agent using the base system. It can be seen that the system with only the *questionLabel* locution added performs the best, having the lowest median and average values. The system with the added *why* locution has a worse median than the two other systems, but its average is lower than that of the base system. Another interesting observation that can be seen is that the

two extended systems are able to close dialogues with an amount of observables shared that is lower than the lowest value achieved by the base system.

5.4.2.2 Unique argument instantiation

Next, experiments were performed using the unique argument instantiation function. These experiments were performed the same as the experiments on dialogue length using the unique argument instantiation function, only with a different measure of performance. The results of these experiments can be seen in appendix C.2.

The standard tree setups, of which the results can be seen in appendix C.2.1, show similar results as to the same experiment done using random instantiation, except that the system with added *why* locution consistently is able to conduct dialogues sharing only 1 observable. It can be seen that with increased complexity of the system, the median and average become lower. The agents are able to create a very efficient policy, knowing that there always is a valid argument to be found for the topic. This way, the agent can probe the tree using *questionLabel* moves and then ask for the relevant observables using *why* moves. An example of a commonly found dialogue for the deep tree that uses such an approach, is shown below in table 5.5.

t	$d[t]$
1	$\langle$inquirer, questionLabel, y$\rangle$
2	$\langle$respondent, claimLabel, (y, U)$\rangle$
3	$\langle$inquirer, questionLabel, b$\rangle$
4	$\langle$respondent, claimLabel, (b, U)$\rangle$
5	$\langle$inquirer, questionLabel, a$\rangle$
6	$\langle$respondent, claimLabel, (a, I)$\rangle$
7	$\langle$inquirer, why, (a, I)$\rangle$
8	$\langle$respondent, since, $(\{(f \Rightarrow a)\}, \{(f, I)\})\rangle$

Table 5.5: An example inquiry dialogue on the deep tree setup, generated by the system with *questionLabel* and *why* moves.

The dialogue in table 5.5 shows the agent probing the respondent for the labels of different literals, until it finds a literal that is labelled I and a direct consequent of a rule with an observation as its antecedent, in this case the literal *a*. The agent then performs a *why*(*a*,I) move that results in f being added to the knowledge base $\mathcal{K}_I$ and a stable state being reached.

The Multi-Rule setups, of which the results can be seen in appendix C.2.2, again show similar results as what we have seen before. With increased complexity of the system and the locutions available to the agents, the agent will perform better in dialogues aimed at minimizing the amount of observables shared. For both the 2 antecedent and 3 antecedent setups similar results can be seen. The added *questionLabel* locution already allows the agent to perform better than the base system and the added *why* locution allows the agent to

always achieve the minimum amount of observables shared needed for a stable setup. An example of a commonly found dialogue for the Multi-Rule tree with 3 antecedent that uses such an approach, is shown below in table 5.6.

t	$d[t]$
1	$\langle$inquirer, questionLabel, x$\rangle$
2	$\langle$respondent, claimLabel, (x, U)$\rangle$
3	$\langle$inquirer, questionLabel, y$\rangle$
4	$\langle$respondent, claimLabel, (y, I)$\rangle$
5	$\langle$inquirer, why, (y, I)$\rangle$
6	$\langle$respondent, since, $(\{(g, h, i \Rightarrow b)\}, \{(g, I), (h, I), (i, I)\})\rangle$

Table 5.6: An example inquiry dialogue on the Multi-Rule with three antecedent setup, generated by the system with *questionLabel* and *why* moves.

The dialogue in table 5.6 shows that the agent starts out questioning the labels of literals that are the premise of a rule with as the consequent the topic t. When the agent sees the response x is U, it knows that it must then ask the label of y, which it could already deduce as being I in a unique argument case. Having the respondent claim the label of y as I then allows the agent to make a $why(y,I)$ move, resulting in the three literals g,h and i being uttered and a stable state being reached.

5.4.2.3 Conflicting argument instantiation

Finally, experiments were performed using the conflicting argument instantiation function. The results of these experiments can be seen in appendix C.3.

The ambiguous setups, of which the results can be seen in appendix C.3.1, show similar results as the experiments done on these setups using the random instantiation function. All the stability-based systems clearly outperform the exhaustive option. Additionally, it can be seen that by increasing the complexity of the system, the performance increases as well. The conflicting argument instantiation function thus did not have a different impact on performance than the random instantiation function.

The complex setups, of which the results can be seen in appendix C.3.2, again show similar results as the same experiments done using the random instantiation function. All three systems perform comparable, with a slight edge for the base system. All the stability-based systems clearly outperform the exhaustive option. One thing that can be noted is that the performance of the stability-based systems is worse in comparison to the experiments done with the random instantiation function, whereas the exhaustive option logically performs exactly the same.

5.5 Discussion of results

In this section we will discuss the previously shown results of the experiments. First we will discuss the general choices made for the experiments and how they could have effected the given results. Secondly, we will look at what the given results mean for our research questions. Thirdly, we will look at the differences in performance between the different instantiation functions and look at how the different systems dealt with this additional information. Finally, we will discuss limitations that were found during and after the experiments.

5.5.1 Experiment setup and choices

In this section we will discuss the general setup of the experiments and the choices that were made. The argumentation setups for the experiments were chosen in such a way, that the experiments started with the most simple argumentation setups and gradually the complexity would increase. The thought behind this was that it would be possible to pinpoint at what point of argumentation setup complexity each of the different systems would excel at. For this purpose, we first tested simple setups using rules with only one antecedent and no conflicting arguments. We then experimented with rules that have multiple antecedents, setups that have conflicting arguments and finally setups that have all these characteristics and are larger in size. In real life situations, these final large and complex setups would be most representable and thus the results seen for these setups should be most representable for the real life performance of the systems as well. The full set of setups is chosen in such a way that the most common situations should be covered in them, but the set is in no way exhaustive. It can thus be argued that more experiments with different setups would still be desirable, but not needed. The random instantiation function might not be the most representable function to initiate the knowledge base of the respondent with and to learn a policy with. In real life scenarios, it is highly unlikely that the prior probability of all observables is exactly a 50% chance. Because of this, additional experiments were performed using the unique argument and the conflicting argument instantiation functions. These functions are based on the idea that a dialogue will most likely be started when there is at least an idea for an argument for the topic existing.

5.5.2 Effects on performance

In this section we will look at what the effect on performance is of the added complexity to the base system of Testerink, Odekerken and Bex and how the stability-based systems perform compared to the exhaustive system. First we will look at the results based on the performance metric being dialogue length and secondly we will look at the results based on the performance metric being the amount of observables that are shared during the dialogue. After discussing these found results, we will be able to answer our research questions on the performance of the inquiry systems.

When looking at the results based on the dialogue length, it can be seen that in the simpler setups there is not a large difference in performance to be found between the stability-based systems. The averages and medians stay similar, even though the more complex systems do have outlier cases where they perform worse than exhaustive options. The first setup of the complex scenarios set shows similar results, the fraud intake scenario, however, performs significantly worse in both of the two extension systems. There could be two possible causes for this drop in performance. Firstly, the agents using the two extended sets of locutions have a significantly larger state space and set of actions at their disposal for each state. The downside of this fact is double. The increased state space has as a result that the agent has to spend significantly more time learning a policy, since there is a larger state space to explore and learn. The other downside of this is that the MDP and its related Q-values, that are used for Q-learning, can grow so large that it will not fit into memory anymore. As a result of this, the agent has to stop learning sooner than is desirable, to prevent the system running out of memory. This results in sub-optimal policies and worse performance than the simpler base system of Testerink, Odekerken and Bex, that does not have this problem of a too large state space. This also means that it is yet unsure, whether or not the two extended systems are capable of generating more efficient policies than the base system when measuring dialogue length. One can make a prediction that they can, based on the results of the smaller scenarios, but it is in no way confirmed by our experiments so far. The setups in which the most expressive system did excel at, were that of the Multi-Rule trees. The stability-based systems were consistently able to outperform their exhaustive alternative. Table 5.7 gives a brief summary of the findings of the experiments on dialogue length using random instantiation and indicates what system would be most suitable to use in what test cases.

Test Class	Most Suitable System
Standard Trees	Indifferent between stability-based systems
Multi-Rule Trees	TB+?L+WHY
Ambiguous Setups	Indifferent between stability-based systems
Complex Setups	TB

Table 5.7: An overview of what system has shown to be most suitable for what setup classes, taking into account randomly generated cases based on minimizing dialogue length.

Generally, the results based on the unique argument instantiation test cases show more promise than the results of the randomly generated cases. In these cases it can clearly be seen that the extended systems benefit from their added locutions and are able to perform more efficiently compared to the base system. Table 5.8 gives a brief summary of the findings of the experiments on dialogue length using unique argument instantiation and indicates what system would be most suitable to use in what test cases.

Test Class	Most Suitable System
Standard Trees	TB+?L+WHY
Multi-Rule Trees	TB+?L+WHY

Table 5.8: An overview of what system has shown to be most suitable for what setup classes, taking into account cases generated using the unique argument instantiation function and based on minimizing dialogue length.

The results of the experiments done using the conflicting argument instantiation function were similar to that of the experiments done using the random instantiation function. In the setups of the ambiguous class, the performance of the three stability-based systems was comparable. In the case of the complex classes though, an amplification of the effect that was shown for the randomly instantiated experiments was seen. The performance of the two extended systems was lower than that of the base system. In the case of the fraud intake scenario, the two extended systems performed even worse than the exhaustive option. Table 5.9 gives a brief summary of the findings of the experiments on dialogue length using conflicting argument instantiation and indicates what system would be most suitable to use in what test cases.

Test Class	Most Suitable System
Ambiguous Setups	Indifferent between stability-based systems
Complex Setups	TB

Table 5.9: An overview of what system has shown to be most suitable for what setup classes, taking into account cases generated using the conflicting argument instantiation function and based on minimizing dialogue length.

The results based on the amount of observables shared are more promising than the results based on the dialogue length. It can be seen that the system with the added *questionLabel* move performs equally or better than the base system and the system with added *why* locution steadily outperforms both the other two systems. The agent can effectively minimize the amount of observables shared by first probing for the labels of different defeasible literals. Based on the information it gathers by uttering those moves, it can then work its way down towards an argument using either *question* or *why* moves. This results in longer dialogues, but since in these experiments the dialogue length was not a performance metric, this does not affect the performance of the agents. The same problem as in the agents that learned a policy based on dialogue length can be seen again here in the two complex setups. The state space of the MDP grows so large, that whilst learning the agent will run out of memory long before reaching an optimal policy. However, contrary to what we have seen in the experiments on dialogue length, the two extended systems were already able to calculate a policy that performs better than the base system in the most complicated setup. It is possible that if the agents were able to learn for more rounds than that they did in the experiments that this performance increase

would grow even larger. Table 5.10 gives a brief summary of the findings of the experiments on shared observables using random instantiation and indicates what system would be most suitable to use in what test cases.

Test Class	Most Suitable System
Standard Trees	TB+?L+WHY
Multi-Rule Trees	TB+?L+WHY
Ambiguous Setups	TB+?L+WHY
Complex Setups	Indifferent between stability-based systems

Table 5.10: An overview of what system has shown to be most suitable for what setup classes, taking into account randomly generated cases based on minimizing the amount of observables shared.

We can again see that when using the unique argument instantiation function, that the added complexity of the agents does benefit the performance. The system with only the added *questionLabel* locution already steadily outperforms the base system and this result is further shown by the system with the added *why* locution. The system including the *why* locution is enable to consistently minimize the amount of observables shared in a dialogue, without having any outlier cases. This is a result of the backward chaining like behaviour that the agent again learned themself. Table 5.11 gives a brief summary of the findings of the experiments on shared observables using unique argument instantiation and indicates what system would be most suitable to use in what test cases.

Test Class	Most Suitable System
Standard Trees	TB+?L+WHY
Multi-Rule Trees	TB+?L+WHY

Table 5.11: An overview of what system has shown to be most suitable for what setup classes, taking into account cases generated using the unique argument instantiation function and based on minimizing the amount of observables shared.

The results of the experiments done using the conflicting argument instantiation function were similar to the results of the experiments done using the random instantiation function. In the case of the ambiguous setups, it could be seen that the larger the set of locutions that the agent has at its disposal, the better the performance was. In the complex setups the three stability-based systems performed similar to each other and clearly outperformed the exhaustive option. Table 5.12 gives a brief summary of the findings of the experiments on shared observables using conflicting argument instantiation and indicates what system would be most suitable to use in what test cases.

Test Class	Most Suitable System
Ambiguous Setups	TB+?L+WHY
Complex Setups	Indifferent between stability-based systems

Table 5.12: An overview of what system has shown to be most suitable for what setup classes, taking into account cases generated using the conflicting argument instantiation function and based on minimizing the amount of observables shared.

Having found these results, we are now able to answer our research questions. Firstly, how does the set of available locutions for both agents affect the performance of inquiry dialogues? From our experiments we can conclude that having a wider range of locutions available to the agents in their dialogues, can lead to better performance regarding dialogue length and the amount of observables shared. This result did not show for all the cases that we have looked at in our experiments, but this does not have to do with the addition of the locutions themselves. This was more than likely caused by the large action-state space of the MDP used to learn a policy for the larger rule sets. Theoretically speaking, the system with the most locutions would always perform the best or equal to the best system, when given enough time and resources to learn a policy. This holds for the simple reason that the most expressive system, could always learn a policy using only moves from the less expressive systems, if they perform better. This behaviour could be seen, for example, in the standard tree experiments on dialogue length, in which the more expressive systems created policies similar to that of the base system. Besides the two metrics of dialogue length and amount of observables shared, the added expressiveness as a result of the new locutions can be considered as a performance gain of its own as well.

Next, we will look at how the rule set used in inquiry dialogues affect the performance of inquiry dialogues. In our experiments it showed that performance varied greatly between the different rule sets used, especially when considering the extended two systems. There are two factors of importance here: firstly, the size of the rule set greatly affected the performance of the two extended systems, since the state space of the MDP's grew so large that the agents couldn't learn an optimal policy. Secondly, the structure of the rule set impacted performance of the different systems used. It can be seen that some of the rule sets allowed the extended systems to learn and use the structure of the rule set in their advantage, where in the simplest setups this did not give a large advantage.

Finally, we will look at how the stability-based systems performed compared to the exhaustive system. As expected, the stability-based systems systematically outperformed the exhaustive system. This can be seen in all experiments done on both dialogue length and observations shared. In some experiments, the extended systems performed worse than the exhaustive system, but then the base system would still perform best out of all systems. It can thus be argued that at least one stability-based option is always preferred over the exhaustive option and thus stability-based systems as a class are preferred over exhaustive systems.

5.5.3 Instantiation functions

In this section we will discuss the influence of the instantiation functions used on the results of the experiments. For our experiments we used three different instantiation functions to learn policies and to instantiate the knowledge base of the respondent with for each dialogue. Firstly, we used the random instantiation function. This function randomly gives a truth value of true or false to each observable literal. Using this instantiation function, the system is able to learn each possible case of instantiation for the observables and thus the agent should be able to learn a general policy that would hold itself up in real life. The downside of this instantiation function is that in real life, there almost never is such a random distribution of true and false for each possible observable. In real life the truth values of certain literals increase or decrease the chances of other literals being true or false as well. It would most likely affect the performance of the agent positively if we were able to feed it cases that happened in real life situations, instead of randomly generating cases. Another downside of the randomly generated cases is that a lot of cases will be generated that end up in unsatisfiable outcomes. It is very unlikely that a dialogue between agents will be started without one of the agents having some suspicion of an argument for the topic existing. In an effort to test the hypothesis that the agents would perform better when there is a lead for an argument of the topic existing, we also introduced the unique argument instantiation function and the conflicting argument instantiation function. The idea behind the unique argument instantiation function is that it creates cases in which there always is exactly one argument for the topic. The agents were able to learn this fact and create policies that use this knowledge to their advantage. It could be seen that as a result of these added constraints to training and testing cases, the extended systems were more effective at creating policies than the base system. The results of the conflicting argument instantiation function differed less from the results of the random instantiation function. In general, the performance of all stability-based systems was comparable to the performance in the case of randomly generated dialogues. The most notable difference was that the two extended systems performed significantly worse in the fraud intake scenario with conflicting argument instantiation than in the experiments with the random instantiation function. All in all, a small case can be made that when the agent can make certain assumptions about the knowledge base of the respondent, they can abuse this knowledge in order to perform more effectively.

5.5.4 Limitations

In this section we will discuss the limitations of the experiments performed and the limitations of the proposed systems that are brought to light by the experiments.

Firstly, we will look at the limitations of the experiments we have performed. In the selection of setups, we tried to select them in such a way that the different characteristics found in argumentation setups could be tested isolated and com-

bined. It was tested how the width and depth of standard tree like setups affects the performance, how rules with multiple antecedents affect performance, how setups that contain conflicting arguments affect performance and finally how complex systems that combine all these characteristics and systems that could be used in real life perform as well. A downside of this approach was that primarily relatively small and focused setups were experimented with. In real life, it is more likely that these setups are larger and contain many exception cases and attacking arguments. We tried to experiment with setups that were used in real life, but they were too complex for the two extended systems to effectively learn optimal policies for. A missing element from the set of test cases, and thus a limitation of the experiments, are smaller setups that are based on real life situations. Using such setups we would have been able to measure the real life performance for smaller situations, in which the agents would be able to reach an optimal policy before running out of memory.

In our experiments we tested with two different utility functions, using which the agents learn their policies and conduct their dialogues. Realistically speaking, however, in real life there usually is not a clear preference for only dialogue length or only the amount of observables shared. It is very likely that a combination of the two utilities actually is preferred when such a system is used in real life.

Finally, we will look at the limitations of the proposed systems themselves that the experiments brought forth. By giving the agents access to more actions and more information, the state space of the MDP grew exponentially large. Whilst in theory this could be a good thing, more efficient strategies could be found in this enlarged state space, in our experiments it showed that it grew so large that it prevented the agents from learning the optimal policies and thus strategies on the larger argumentation setups. The agents were not able to perform the necessary amount of rounds needed for the policies to stabilize and thus were left with sub-optimal policies that sometimes performed even worse than, or similar to, exhaustive options. A large limitation of the systems in their current form thus is the inability to efficiently deal with larger argumentation setups. Another downside of this increased state space and the increased amount of actions is that it takes significantly longer time-wise to learn an efficient policy. However, since a policy only has to be learned once, this only is a slight inconvenience compared to the limitation of the MDP growing too large to fit into memory.

Chapter 6

Discussion

In this chapter, we will discuss the findings that we have made in this thesis. We will first discuss the feasibility of the stability-based inquiry systems in real life situations. Secondly, we will discuss and propose solutions to some of the limitations that were found in the experiments and in the systems themselves.

6.1 Feasibility in real life situations

How feasible would the current implementations of the proposed systems be in real life situations? As explained earlier in the limitations section, that partly depends on the size of the argumentation setup it would have to use for its dialogues. When the argumentation setup is too large, the two extended systems are not able to learn the optimal policy and it would be preferable to use the base system of Testerink, Odekerken and Bex, which is able to handle such larger setups efficiently. This is especially true if the only measure of performance is the average dialogue length. Another factor in whether or not the systems are feasible are the assumptions the agents can make for the responses. We saw earlier that when the agent can assume that there always one and only one valid argument for the topic to be found, the more complex systems perform significantly better.

6.2 Possible solutions for the found limitations

In this section we will again look at the limitations we have found and briefly discuss possible solutions for them.

Firstly, we will look at the largest limitation of the two extended systems that showed during experiments. The size of the MDP and its related Q-values grows so large during the learning phase, that the objects do not fit into memory anymore before reaching a stable policy. Multiple possible solutions for this problem do exist. One solution is to reduce the state space on which the agents

are acting. This can be achieved by, instead of having a state be all the knowledge the agent has at its disposal, having states be defined by features over the knowledge base of the agent. Another option is to simply prune certain parts of the MDP, which show such unpromising results that it can be assumed that an optimal policy would never have to use these state-action pairs. Another solution would be to learn the optimal policy in a different way than Q-learning. There are many algorithms that allow agents to learn policies of an MDP. The agents could for example learn the policies using dynamic programming.

Secondly, we will discuss the limitation of the selection of argumentation setups used in the experiments. The setups were chosen to represent possible features that could be found in more complex setups, but then tested separately. We also tested complex setups in which all the tested features were present combined. It is difficult to justify the set of setups chosen to be truly representable for setups that would be used in real life. We used one scenario, the fraud intake setup, that is representable for a setup that is used in real life decision making. This, however, is only one example of such a setup and a rather large one. It can be argued that more experiments, using argumentation setups based on small and large scenarios used in real life decision making, would be desirable.

Thirdly, we will address the limited choice of instantiation functions used in the experiments. Our experiments mostly used a random instantiation of knowledge to learn cases and conduct dialogues. For the simple setups without conflicting arguments we also used an instantiation function that instantiates knowledge bases in which always one and only one in argument for the topic could be found. For the setups in which conflicting argument could exist, we additionally used the conflicting argument instantiation function that generates cases that are either *In*, *Out* or *Blocked*. Theoretically speaking, the random instantiation function can return any possible case of knowledge and should thus give a decent representation of cases that can be found in real life. Realistically though, a lot of bogus instantiations are created by this function as well. This gives the agent a lot of bogus cases during the learning phase, costing precious resources that could be used for representable cases, and also creates a lot of unrepresentable cases in the experiments. It is hard to imagine, for example, that a dialogue would be started in which the respondent knows that none of the arguments could ever be in and all the observables have a value of false. We tried to experiment with the idea that the respondent would always start an argument with at least one argument for the topic being *In*, which is represented by the unique argument instantiation function and the conflicting argument instantiation function. Results showed that the different instantiation function could boost the performance of both two extension systems, but in the case of the most complex setup it actually hindered the performance of the agent. Ideally, we would have trained and tested the agents using cases that come from real life data in which there exist conditional probabilities between the different queryable literals. This way it can become more clear how well the different systems would perform in real life scenarios.

Finally, we will discuss the limitations of the amount of utility functions used to learn and measure the performance of dialogues with. In our experiments

we trained our agents to either minimize the dialogue length or minimize the amount of observables shared. These are two good metrics, but they were tested isolated from each other. One can imagine that in real life, a trade off exists between the two metrics. This could be tested and trained using a hybrid utility function, in which a weighted sum of observables shared and dialogue length would be used as the measure of performance.

6.3 Contribution

This section gives an overview of contributions that this thesis has given. Firstly, in this thesis we have given an overview of research done on inquiry dialogues. We examined and compared the different methods with each other and used this information as a basis and inspiration for our own research. Secondly, we extended a stability-based querying system into a stability-based inquiry dialogue system, creating the first of its kind and a viable competitor for exhaustive dialogue systems. Multiple versions of this stability-based dialogue system were defined and implemented, each with a different level of expressiveness. Finally, we performed experiments to measure the performance of the different systems in different situations and the impact on performance of the two extensions.

Chapter 7

Conclusion

This chapter will summarize the findings of this thesis.

This thesis aimed to make a contribution on the field of agent argumentation and more specifically inquiry dialogues. In inquiry dialogues, two or more agents try to prove a proposition together [24]. Such dialogues follow a protocol that dictates what moves can be made and when they can be made. The goal of this thesis was to come up with new ideas that increase the performance of agents in inquiry dialogues. To achieve this goal, we first had to do a literature research of current works done on inquiry dialogues and how they each measure their performance.

In our literature research, we have looked at different works done on the topic of inquiry dialogues. The first work we have looked at is that of Black and Hunter [5], which can be considered the de facto standard in inquiry dialogue systems. Black and Hunter defined an inquiry dialogue system that can perform two sub-types of inquiry dialogues. The strategy the agents use in their system is exhaustive, which means that the agents make all moves at their disposal that could have a possible impact on the outcome. Due to this exhaustiveness, their approach has the benefits of being both sound and complete. The downside of this approach is that dialogues are long and include a lot of redundant moves. Kumeling [14] proposed multiple improvements upon the exhaustive strategy defined by Black and Hunter. Two of the proposed strategies generated smaller dialectical trees than the exhaustive strategy, but they were not sound and complete anymore and the size of the actual dialogues was generally not smaller. The other strategies were sound and complete but only showed a marginal increase in dialectical tree size or dialogue length. Yan et al. [25] also proposed a system that is based on the work of Black and Hunter. Their aim is to improve feasibility of using the system of Black and Hunter in real life situations. In their system, Yan et al. use a different type of logic than Black and Hunter. The system of Black and Hunter uses defeasible logic, where Yan et al. chose to use possibilistic logic. Yan et al. give the agents slightly different data structures to store their knowledge in and the protocol returns one legal move at a time, in contrast to an entire list of all moves. Yan et al. claim that their system

provides simpler implementations that generate clearer and faster dialogues. Parsons et al. [16] give protocols for inquiry, information-seeking and persuasion dialogues. The goal of their research was to give detailed characteristics of outcomes of their generated dialogues. Their agents use so called attitudes, which determine when agents are allowed to assert arguments and accept the assertions of another agent. They show that by adjusting the protocols used in the dialogues, the generated dialogues either have the properties of soundness and completeness or they do not have them. The preference for having these properties differs between the domains in which the system is used and Parsons et al. show that these characteristics can be adjusted. Testerink, Odekerken and Bex [23] propose a querying system for inquiry that aims to be more efficient than exhaustive approaches. Their method uses the concept of stable states, states after which the status of the topic cannot change anymore, in order to calculate a policy that can be used for strategies. They define an approximation algorithm that is a sound approximation of stability. The policy learned by the agents allows them to reach a stable state with as little queries made as possible. Their method shows promise, but is not a fully defined dialogue system yet. Fan and Toni [10] propose a single dialogue model that is able to perform multiple types of dialogues. The model they propose is based on game-theory and does not use separate protocols for each dialogue type, as a traditional system would. They show that their system is able to perform information-seeking and inquiry dialogues using the same single dialogue model.

After the literature research, we looked into the different measures of performances that were found in literature in order to answer our first research question: "What are the best performance measures for inquiry type dialogues that model a police intake conversation?". The first measure that was discussed was the dialogue length, which is almost an universal measure of performance in literature. It is desirable that the system generates as short as possible dialogues whilst conveying the same message. A system that has shorter dialogues on average is preferable to a system that has longer dialogues. Another measure of performance that was discussed, but not deemed relevant for this thesis on police intake conversations, is the computational complexity of the dialogue system. Systems that have a high complexity will take a long time in practice to calculate dialogues, which of course is not desirable. The expressiveness of a system indicates in how many ways agents can share and create arguments. This can be classified by the amount of different locutions that the agents are able to utter during a dialogue. It is preferable for agents to converse in as many diverse ways as possible and this could be seen as beneficial for police intake conversations. It also was found to be important to minimize the amount of information that needs to be shared before terminating a dialogue. This performance measure of privacy could be important for police intake conversations. Another measure that could be important is that of explainability. It is preferable for a system to explain the decisions that it has made and how it came to its verdict. Finally, it is desirable for systems to be both sound and complete. This gives certainties on the outcomes that the system generates.

After discussing the performance measures of inquiry dialogue systems, we

started defining our own stability-based dialogue system based on the works of Testerink, Odekerken and Bex [23]. By doing so, we have defined and implemented the first known stability-based inquiry dialogue system. The first thing we did was define a base system, not yet taking into account stability, based on their works. This system consists of two agents of which one is able to question for observable literals and the other agent is able to either deny or confirm them. Definitions were created for the participants, their moves and the dialogue itself. This was all formalised along with a protocol and strategies that the agents could use. When the base system was in place, we defined how the agents could reason using the concept of stability and use this concept in order to learn an optimal policy. Because of the complexity of calculating the stability of a state, we adopted the approximation algorithm for stability as given by Testerink, Odekerken and Bex in their paper. Having defined the base version of a stability based dialogue system, we started exploring extensions of this system. The first extension that we made was giving the inquiring agent the possibility to question the labels of defeasible literals as well, in stead of only being able to question non-defeasible literals. The set of locutions, the protocol, the knowledge bases and the MDP all had to be adjusted in order to support this newly added locution. The final extension made to the system were a *why* and a *since* locution. These locutions are inspired by *why* and *since* locutions from traditional dialogue systems such as that of Parsons et al. [16], but adjusted in order for it to work for labelled literals instead of for arguments. The *why* locution allows the inquiring agent to ask for an explanation of why a label is considered to be the label it is as uttered by the respondent in the form of an argument. As a response to such a *why* move, the respondent is able to utter an explanation in a *since* move. In this explanation the respondent gives a set of rules and labelled literals that explain why the given label was assigned to the literal.

Having defined the base system and the extensions of the base system, we then defined our experiments. The goal of these experiments was to measure and compare the performance of the base stability-based system and its extensions along with an exhaustive option. We first defined the two utility functions that we would use to measure the performance of the different systems. As a result, two sets of experiments were performed. One measuring the performance in the form of dialogue length and another experiment measuring the performance in the form of the amount of observables shared during the dialogue. After defining the utility functions we defined what classes of argumentation setups we were going to experiment with and how we would instantiate the knowledge base of the respondent during training and testing.

Results of the experiments performed using the random instantiation function showed that for dialogue length there was not a significant improvement in performance for the smaller sized argumentation setups. In the larger argumentation setups it became evident that the more complex systems had trouble reaching an optimal policy. This is a result of the larger size of the state-action space and due to the fact that the agents were not able to calculate an optimal policy before the MDP and its Q-values not fitting into memory anymore. The

experiments that measure the amount of observables shared gave better results and showed that the extended systems were able to outperform the base system, especially the system with *why* locution. This system was the single best performer in all classes except the complex setups, where it performed similar to the other two stability-based systems.

The experiments performed using the unique argument instantiation function showed that the extended systems profited more of this instantiation function than the base system. Results of the experiments done on the amount of observables shared were the most promising. The first extension, only having the additional *questionLabel* locution, performed equal to or better than the base system in almost all argumentation setups. The system with the added *why* locution was almost consistently the best performer in these experiments. This system also was the best performer in the experiments done on minimizing the dialogue length.

The experiments performed using the conflicting argument instantiation function showed similar results to the experiments done using the random instantiation function. The extended systems were not able to significantly increase their performance compared to the base system using the additional information that this instantiation function gives. The result that stood out the most, was that the extended systems performed significantly worse on the fraud intake scenario setup when measuring the dialogue length.

In general, the system with the added *why* locution performed the best, especially in the cases where the unique argument instantiation function was used. It was also shown that the stability-based systems as a group were consistently able to outperform the exhaustive system in both utility functions. The experiments did show, however, that as the argumentation setups grow in size, the extended systems have trouble learning their optimal policies. This is because of the fact that the MDP and its Q-values grows so large during the learning phase, that this phase has to be ended prematurely. It can thus be concluded that the extension systems, just as the base system, show promising results, but for larger setups alternatives would have to be found for the current Q-learning method of learning policies.

7.1 Future research

This thesis brings forth multiple ideas and possibilities for future research. The experiments currently done all used a certain division of knowledge, where the respondent has knowledge of all the observables and the agents have a shared rule set. It would be interesting to see what would have to be changed to the systems in order to effectively support the observables being distributed amongst the agents and the agents having separate, possibly overlapping, rule sets. Along the lines of this research, it would possibly be needed for the initiative during the dialogue to be swapped around during the dialogue. The current dialogues all start with the initiative for the *inquirer* and it does not ever shift towards the *respondent*. Having a mixed initiative could possibly bring forth interesting

new dynamics and findings. Finally, the ideas brought forth in the section on possible solutions for the found limitations could be researched in order to determine their effectiveness at solving those limitations.

Bibliography

[1] Alahmari, S., Yuan, T., & Kudenko, D. (2017). Reinforcement learning for abstract argumentation: Q-learning approach. In Adaptive and Learning Agents workshop (at AAMAS 2017).

[2] Amgoud, L., Maudet, N., Parsons, S. (2000). Modelling dialogues using argumentation. In Proceedings Fourth International Conference on MultiAgent Systems (pp. 31-38). IEEE.

[3] Bex, F., Peters, J., & Testerink, B. (2016, August). AI for online criminal complaints: From natural dialogues to structured scenarios. In Artificial Intelligence for Justice Workshop (ECAI 2016) (p. 22).

[4] Black, E., Coles, A. J., & Hampson, C. (2017, May). Planning for persuasion. In Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems (pp. 933-942). International Foundation for Autonomous Agents and Multiagent Systems.

[5] Black, E., & Hunter, A. (2009). An inquiry dialogue system. Autonomous Agents and Multi-Agent Systems, 19(2), 173-209.

[6] Dung, P. M. (1995). On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. Artificial intelligence, 77(2), 321-357.

[7] Dung, P. M., Kowalski, R. A., Toni, F. (2009). Assumption-based argumentation. In Argumentation in artificial intelligence (pp. 199-218). Springer, Boston, MA.

[8] Fan, X., Toni, F. (2011, June). Assumption-based argumentation dialogues. In Twenty-Second International Joint Conference on Artificial Intelligence.

[9] Fan, X., Toni, F. (2014). A general framework for sound assumption-based argumentation dialogues. Artificial Intelligence, 216, 20-54.

[10] Fan, X., & Toni, F. (2015, October). Mechanism design for argumentation-based information-seeking and inquiry. In International Conference on Principles and Practice of Multi-Agent Systems (pp. 519-527). Springer, Cham.

[11] García, A. J., & Simari, G. R. (2004). Defeasible logic programming an argumentative approach. Theory and Practice of Logic Programming, 4(1–2), 95–138.

[12] Hadjinikolis, C., Siantos, Y., Modgil, S., Black, E., & McBurney, P. (2013, August). Opponent Modelling in Persuasion Dialogues. In IJCAI (pp. 164-170).

[13] Hadoux, E., Beynier, A., Maudet, N., Weng, P., & Hunter, A. (2015, July). Optimization of Probabilistic Argumentation with Markov Decision Models. In IJCAI (pp. 2004-2010).

[14] Kumeling, W.D. (2018, November) Asking useful questions in information gathering dialogues.

[15] McBurney, P., Parsons, S. (2009). Dialogue games for agent argumentation. In Argumentation in artificial intelligence (pp. 261-280). Springer, Boston, MA.

[16] Parsons, S., Wooldridge, M., & Amgoud, L. (2003, July). On the outcomes of formal inter-agent dialogues. In Proceedings of the second international joint conference on Autonomous agents and multiagent systems (pp. 616-623). ACM.

[17] Prakken, H. (2005). Coherence and flexibility in dialogue games for argumentation. Journal of logic and computation, 15(6), 1009-1040.

[18] Prakken, H. (2010). An abstract framework for argumentation with structured arguments. Argument and Computation 1, 2 (2010), 93–124.

[19] Rienstra, T., Thimm, M., & Oren, N. (2013, August). Opponent Models with Uncertainty for Strategic Argumentation. In IJCAI (pp. 332-338).

[20] Schraagen, M., Testerink, B., Odekerken, D., & Bex, F. (2018). Argumentation-driven information extraction for online crime reports. International Workshop on Legal Data Analysis and Mining (LeDAM 2018).

[21] Testerink, B. & Bex, F. (2017, October). Specifications for peer-to-peer argumentation dialogues. In International Conference on Principles and Practice of Multi-Agent Systems (pp. 227-244). Springer, Cham.

[22] Testerink, B., Bex F. (2017). Developing Argumentation Dialogues for Open Multi-Agent Systems. Demo at the 20th International Conference on Principles and Practice of Multi-Agent Systems (PRIMA 2017).

[23] B. Testerink, D. Odekerken F. Bex (2019) A Method for Efficient Argument-based Inquiry 13th International Conference on Flexible Query Answering Systems (FQAS 2019). Lecture Notes in Artificial Intelligence, Springer, to appear.

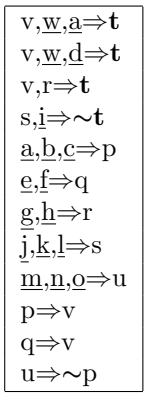
[24] Walton, D., Krabbe, E. C. (1995). Commitment in dialogue: Basic concepts of interpersonal reasoning. SUNY press.

[25] Yan, C., Lindgren, H., & Nieves, J. C. (2018). A dialogue-based approach for dealing with uncertain and conflicting information in medical diagnosis. Autonomous Agents and Multi-Agent Systems, 32(6), 861-885.

Appendix A

Complex Argumentation Setups

A.1 Complex Scenario



A.2 Fraud Intake Scenario

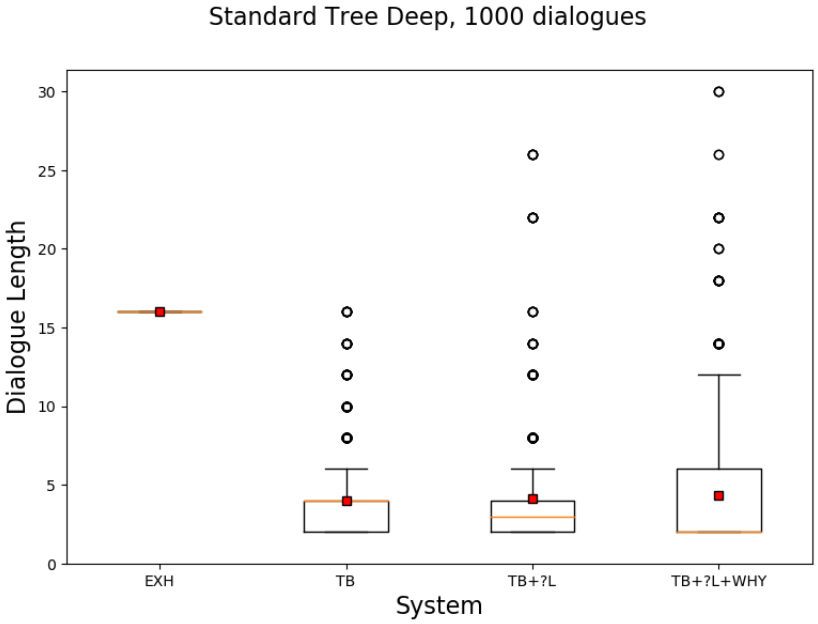
∼bf,mh,ob⇒**f**	Avl,Okl⇒vh
bgh,Owt⇒∼bf	Oaov,Avft⇒vh
Onp⇒∼bf	Ant,Ots⇒gt
wp⇒bf	Atv,Otc⇒gv
∼tp,Ogotp,ob⇒bgh	Ovo,Avv⇒vv
ag,iwvb⇒ob	Oao,Ava⇒vv
Ovm⇒wp	Ows⇒ Avn
Odp⇒wp	∼Oahgo⇒∼Avn
Ows⇒iwvb	∼Opb⇒∼Avv
Ohw⇒iwvb	Opb⇒∼Ava
Oapddda⇒ag	Ows⇒∼Aidfv
Oahgo⇒ag	∼Opb⇒ Atv
Obo⇒tp	Ohw⇒∼Avl
Odp⇒tp	Ows⇒∼Ant
Ovm⇒tp	Ows⇒∼Ava
Opb,Onp⇒∼tp	OAm⇒Avv
∼Og,Opb⇒∼tp	OAm⇒Aidfv
Obdwa,∼Obo⇒∼tp	OAm⇒Avn
gv⇒mh	OAm⇒Ava
gt⇒mh	OAm⇒Aga
vh⇒mh	OAm⇒Avl
vv⇒mh	OAm⇒Ant
Anlb,Obg⇒vh	OAm⇒Avft
Owpla,Aga⇒vh	OAm⇒Atv
Ora,Avn⇒vh	OAm⇒Anlb
Aidfv,Oidf⇒vh	

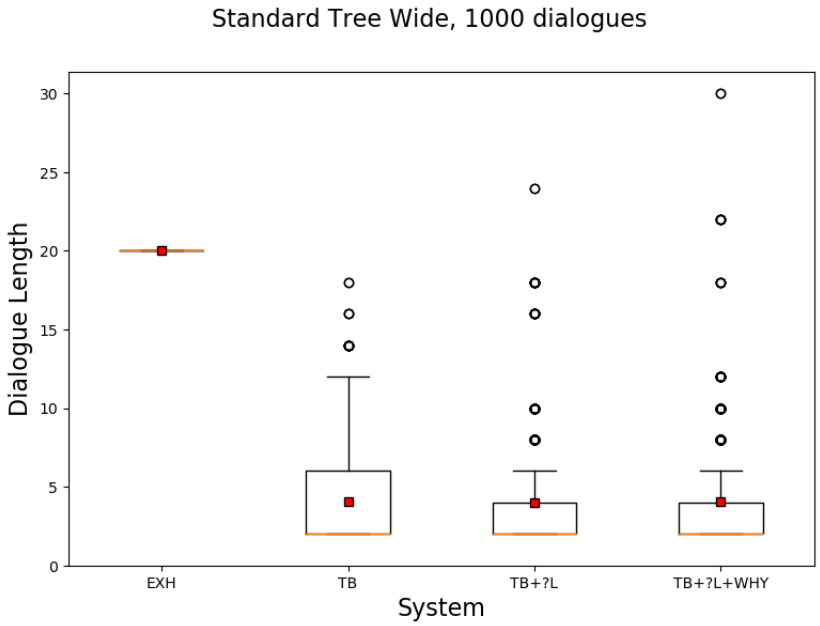
Appendix B

Dialogue Length

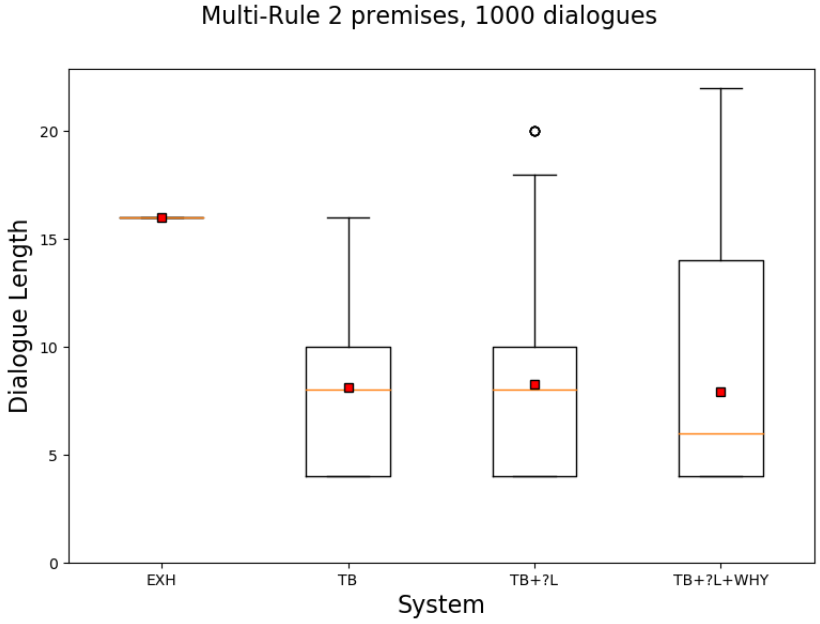
B.1 Random Instantiation

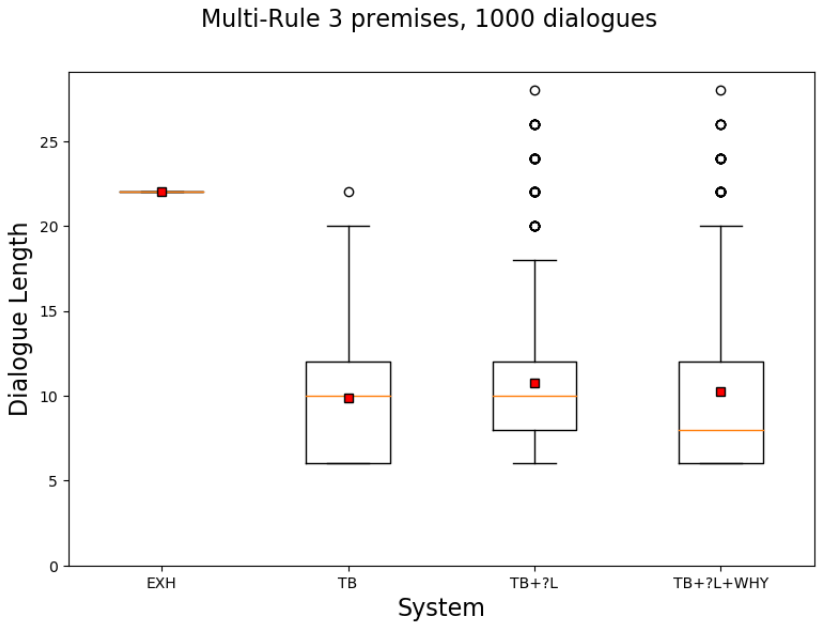
B.1.1 Standard Trees



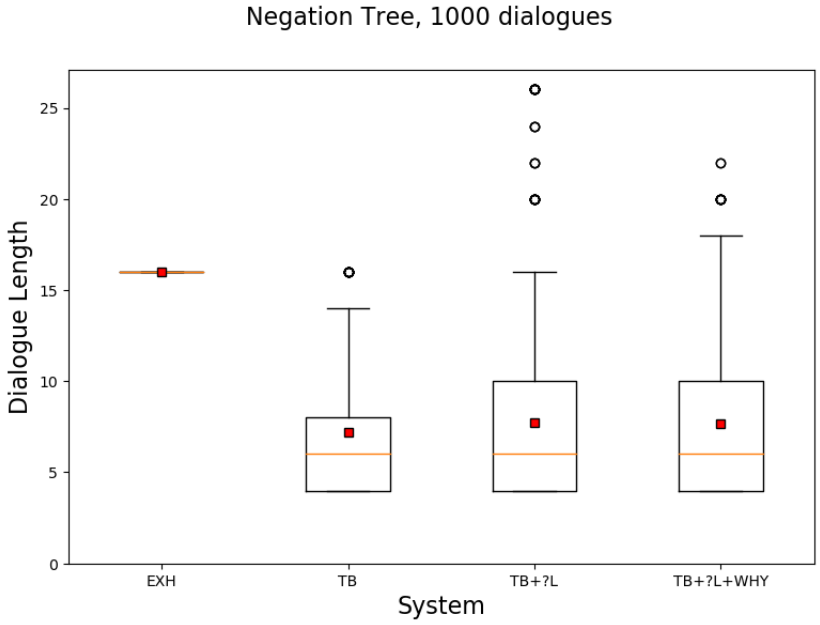


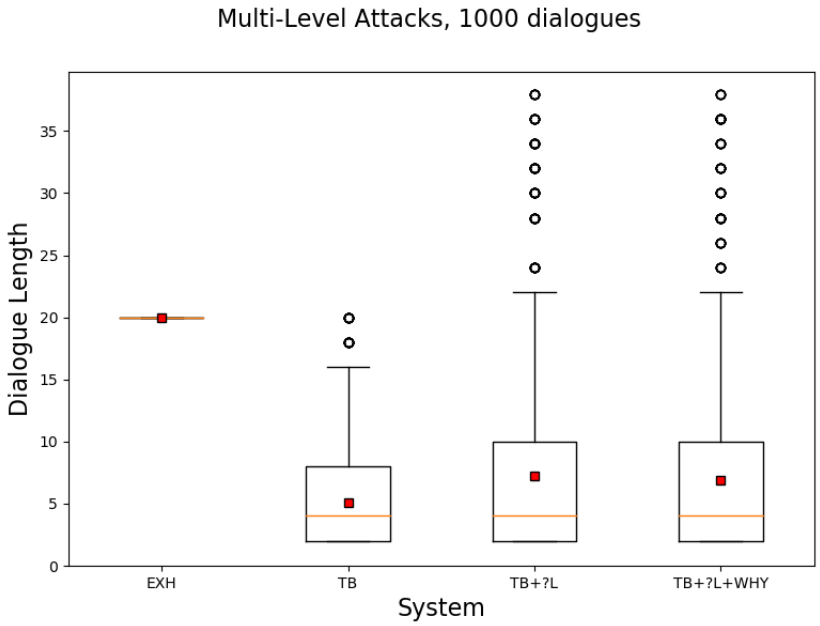
B.1.2 Multi-Rule Trees



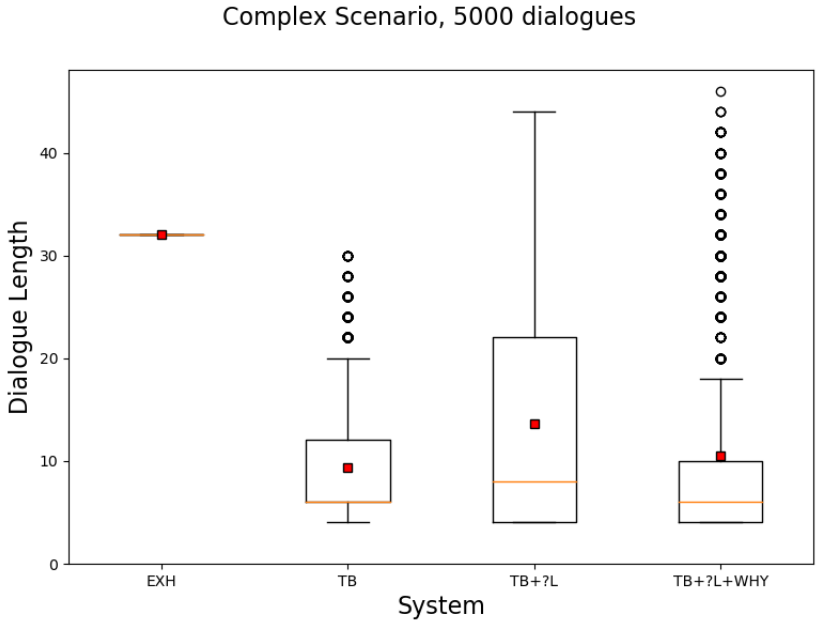


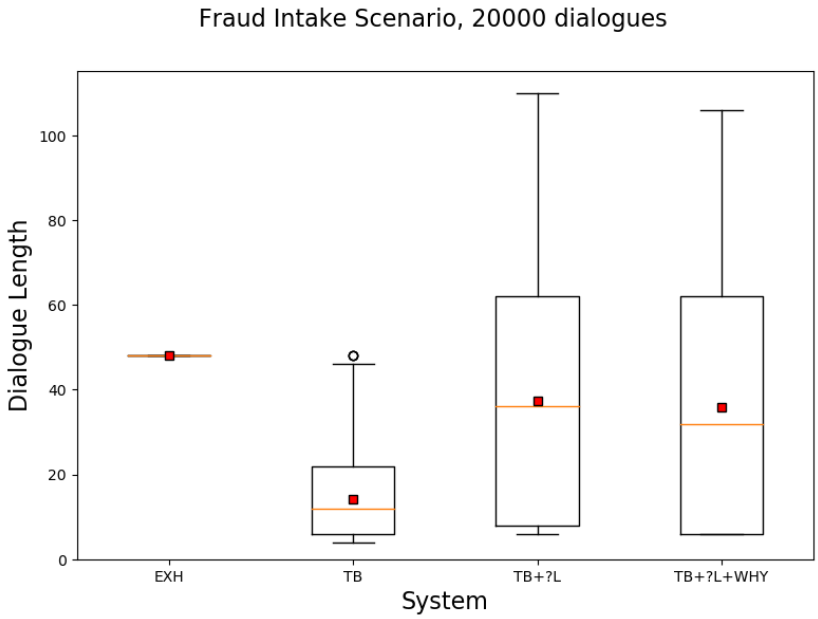
B.1.3 Ambiguous Setups





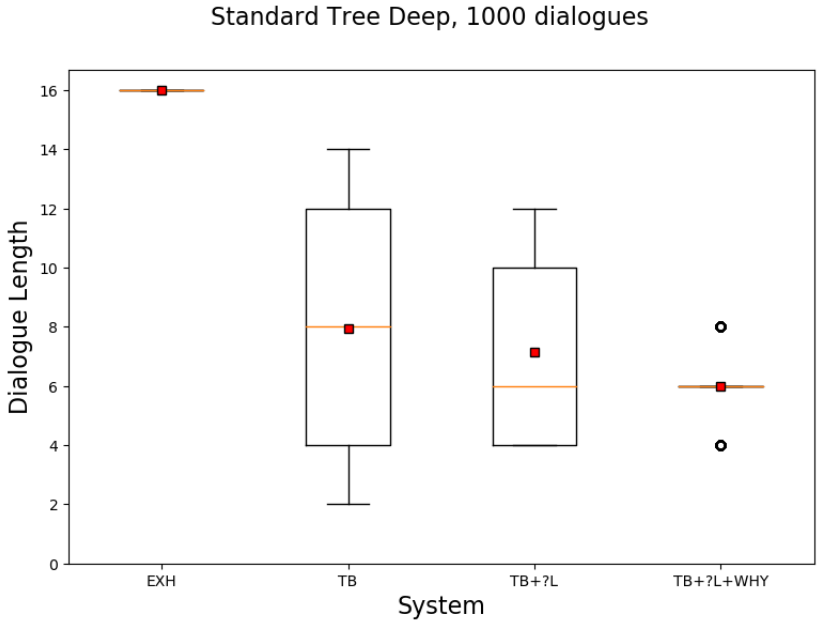
B.1.4 Complex Setups

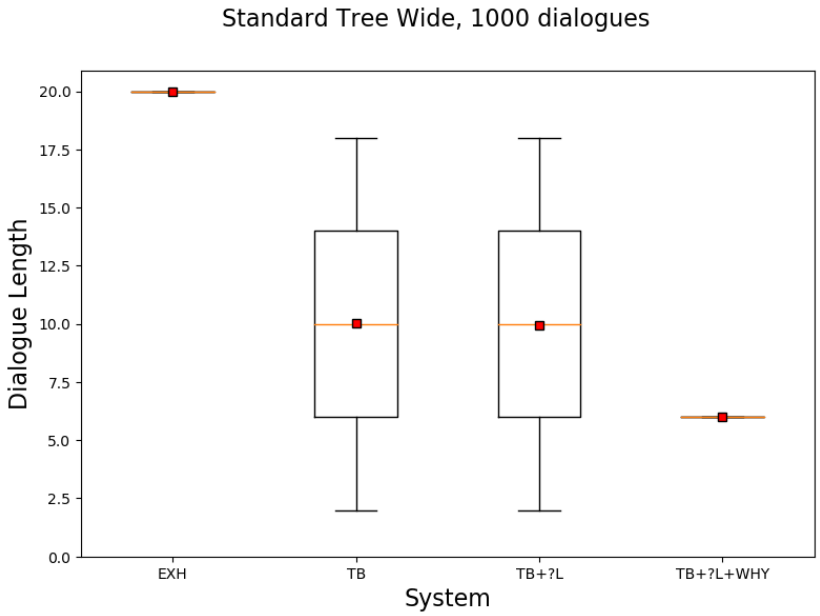




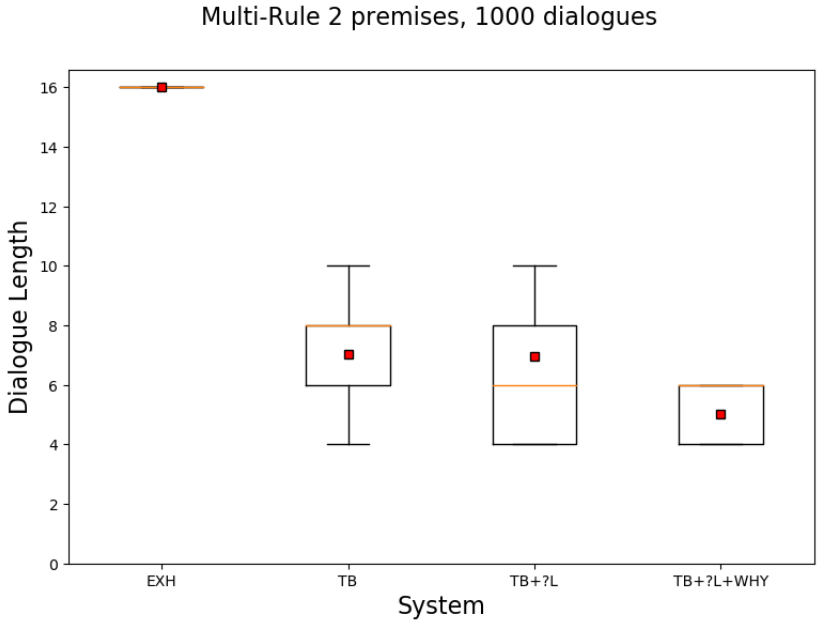
B.2 Unique Argument Instantiation

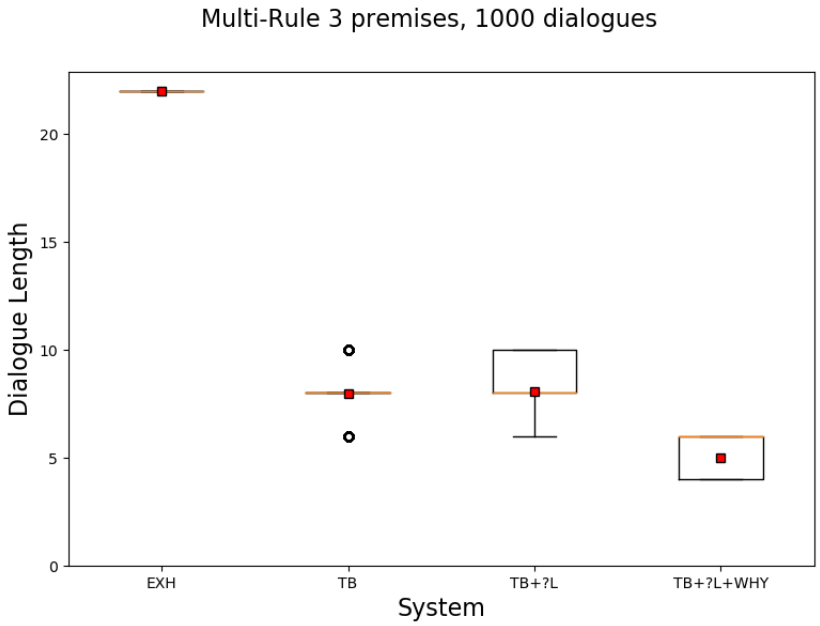
B.2.1 Standard Trees





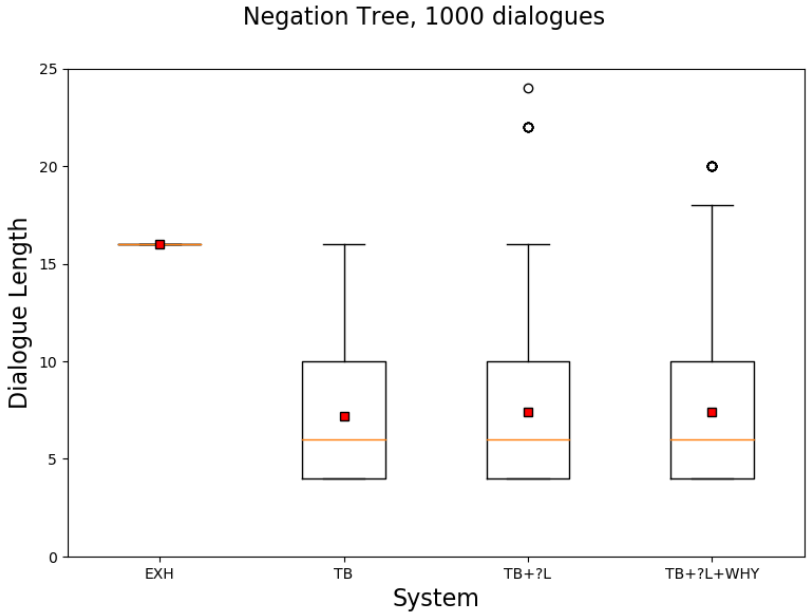
B.2.2 Multi-Rule Trees

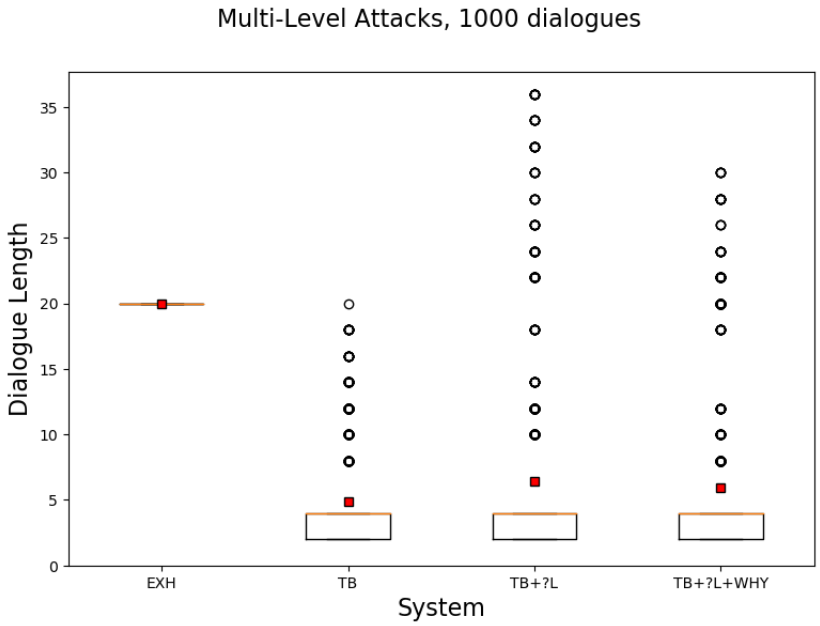




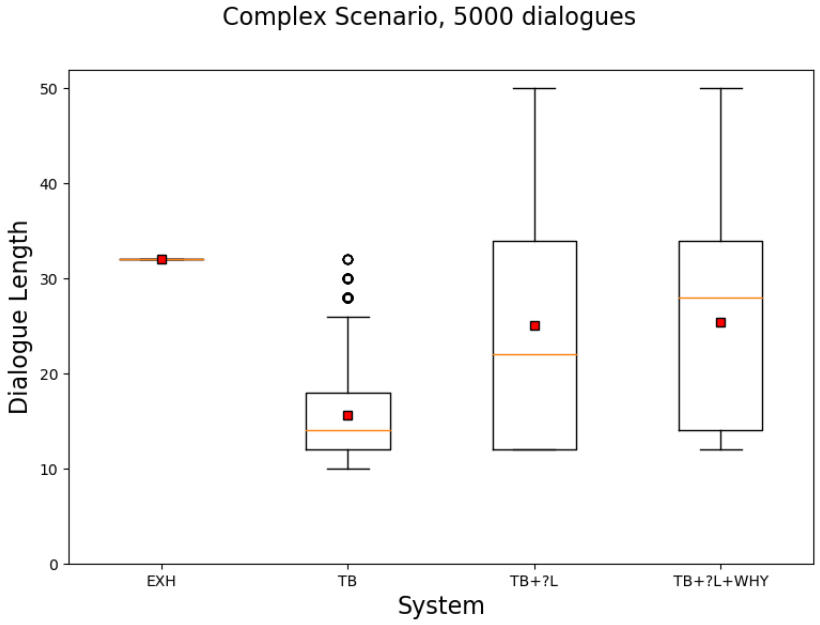
B.3 Conflicting Argument Instantiation

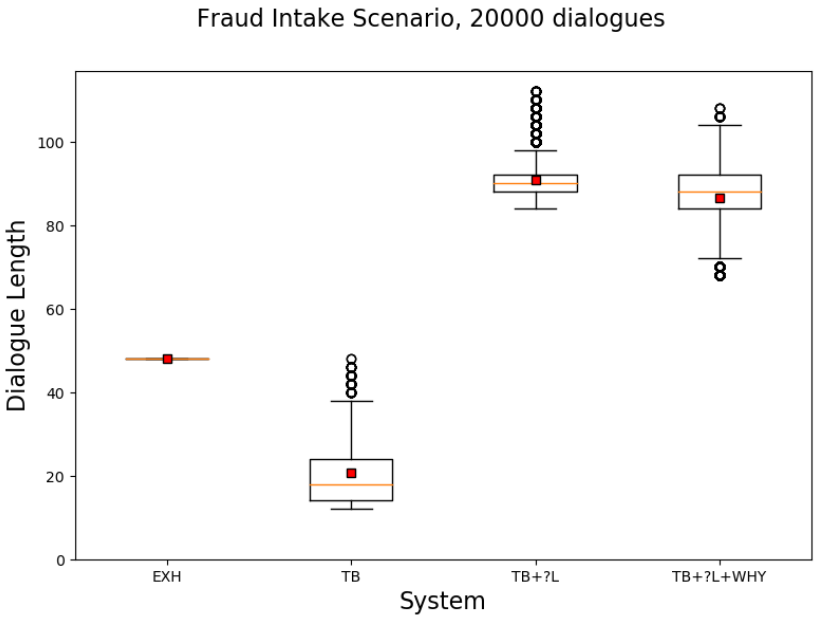
B.3.1 Ambiguous Setups





B.3.2 Complex Setups



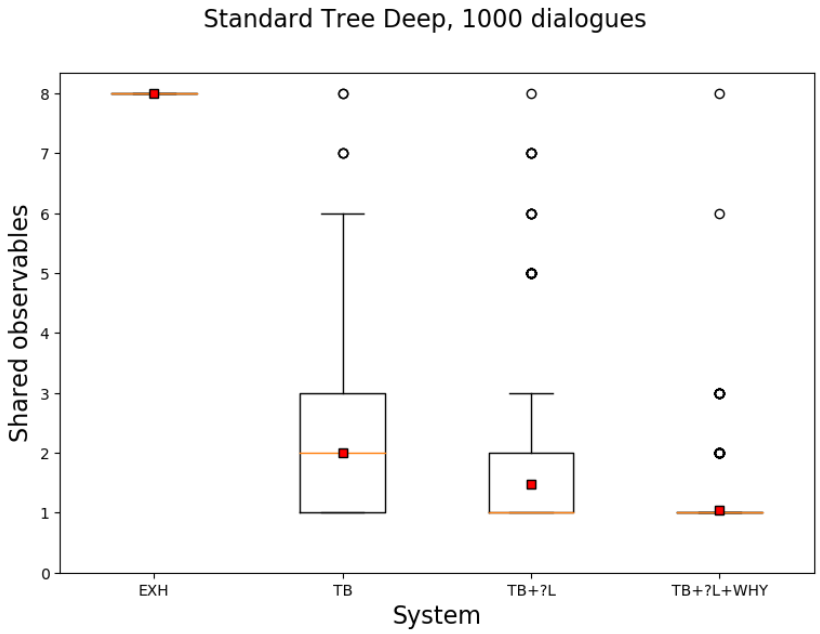

Fraud Intake Scenario, 20000 dialogues
Dialogue Length
0
20
40
60
80
100
EXH
TB
TB+?L
TB+?L+WHY
System

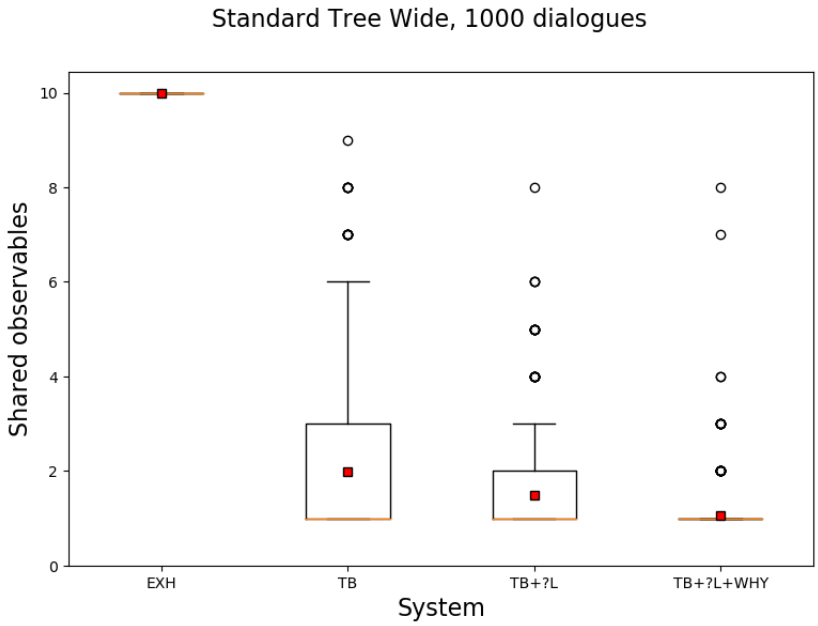
Appendix C

Shared Observables

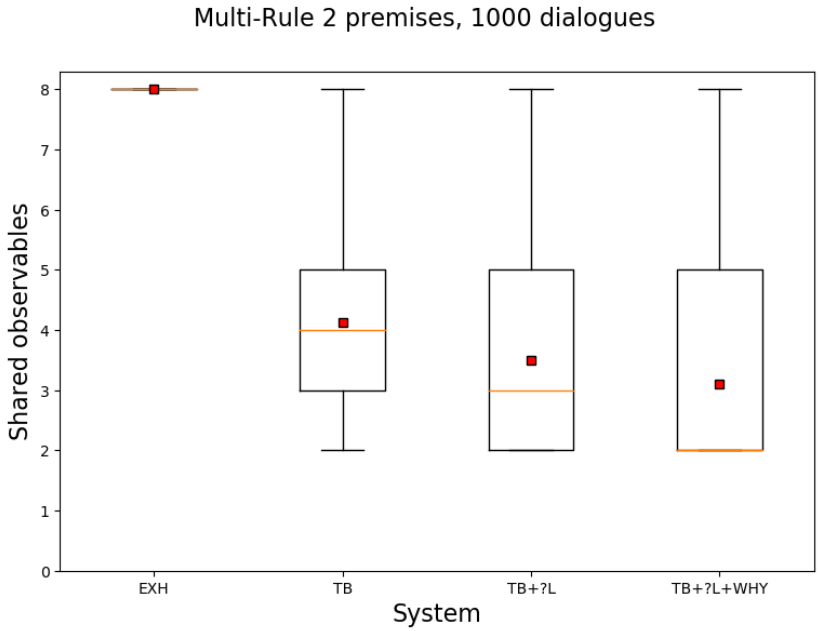
C.1 Random Instantiation

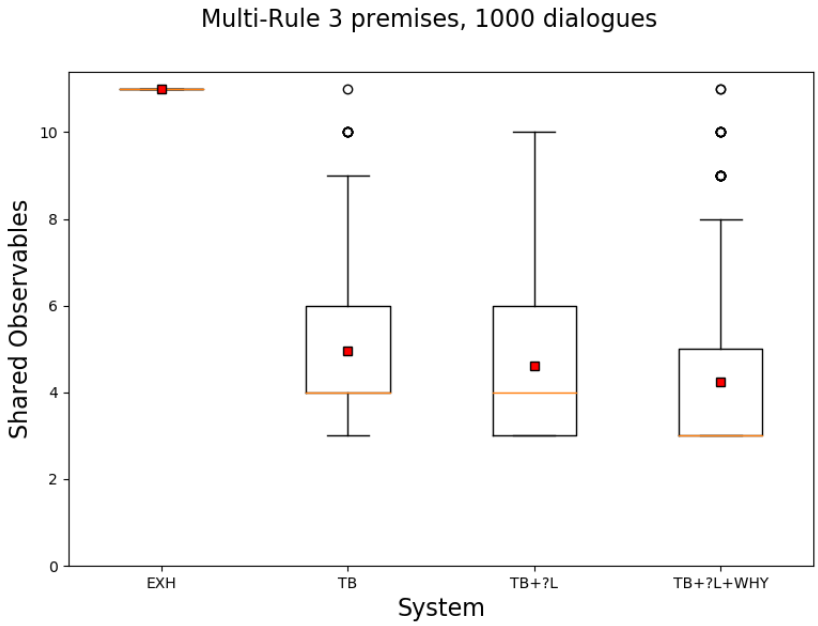
C.1.1 Standard Trees



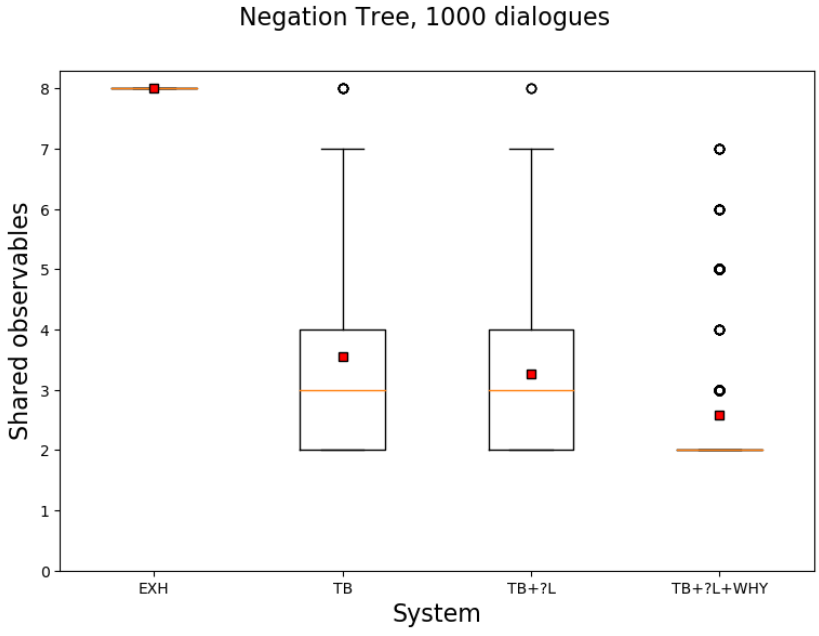


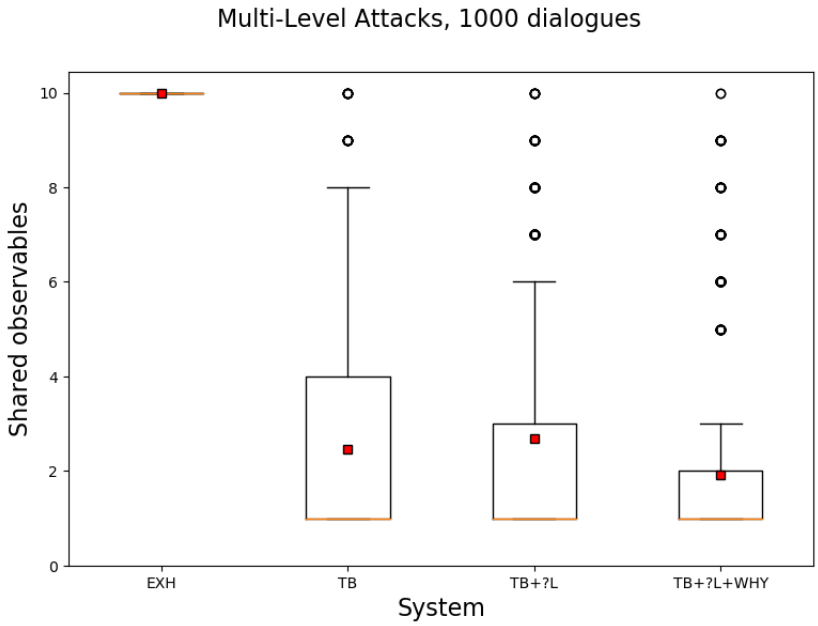
C.1.2 Multi-Rule Trees



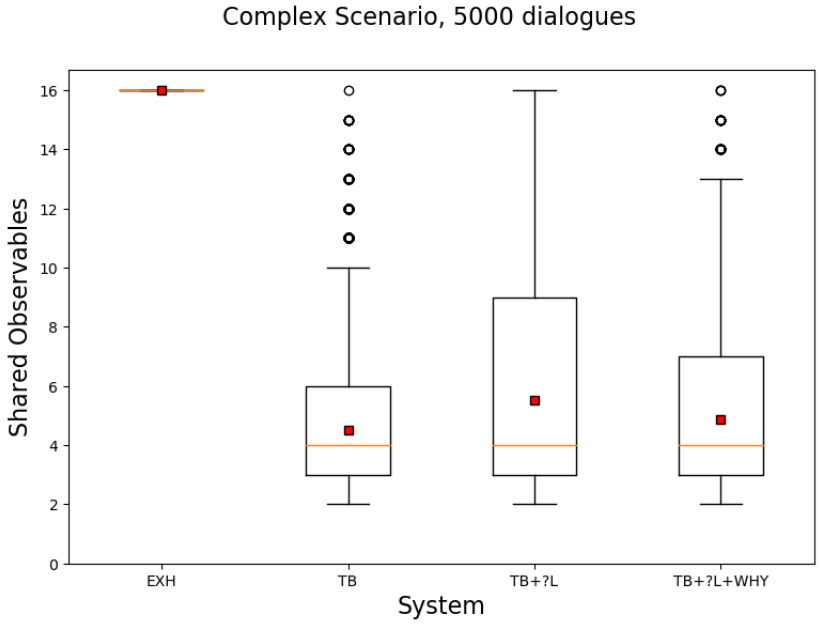


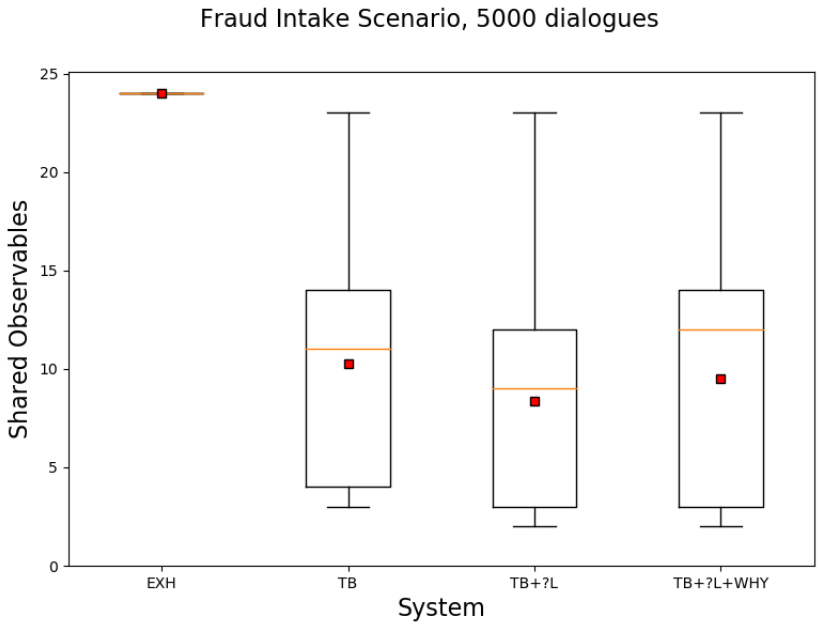
C.1.3 Ambiguous Setups





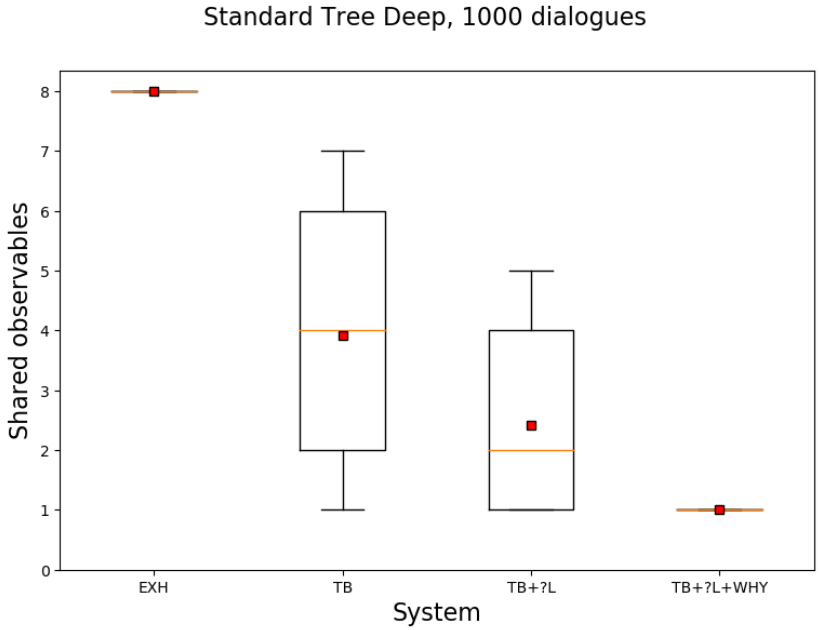
C.1.4 Complex Setups

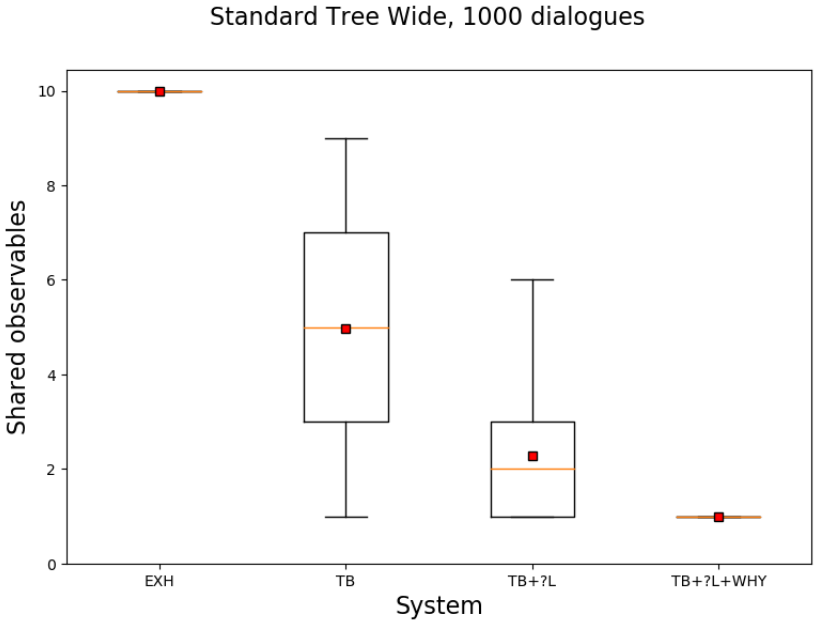




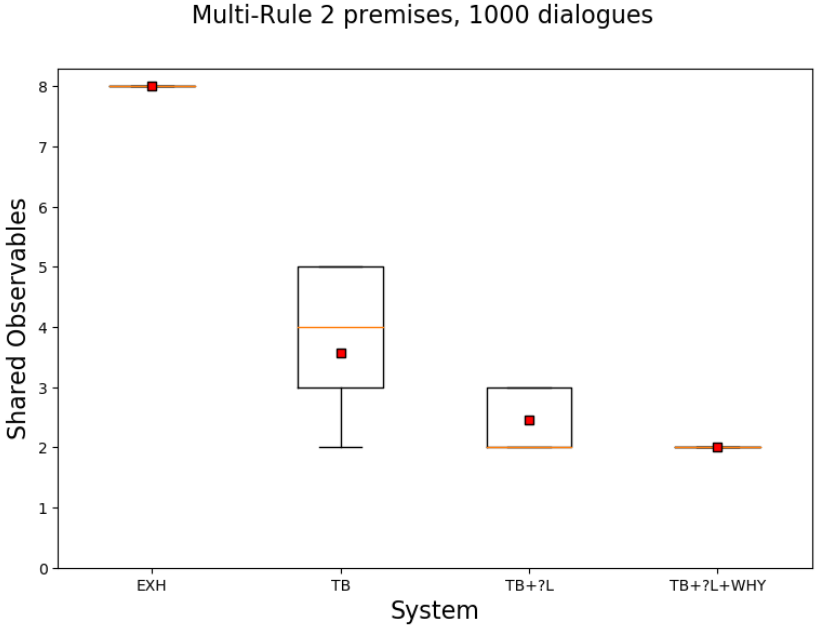
C.2 Unique Argument Instantiation

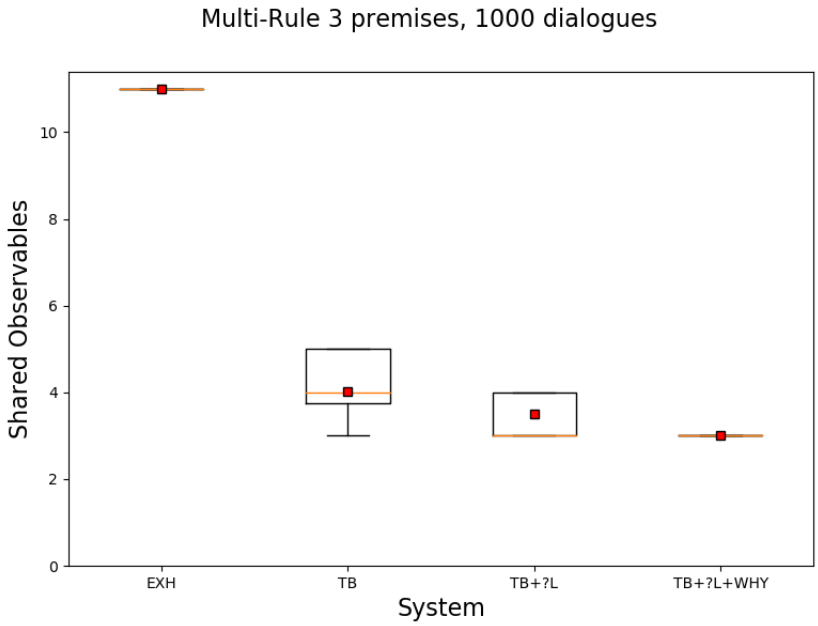
C.2.1 Standard Trees





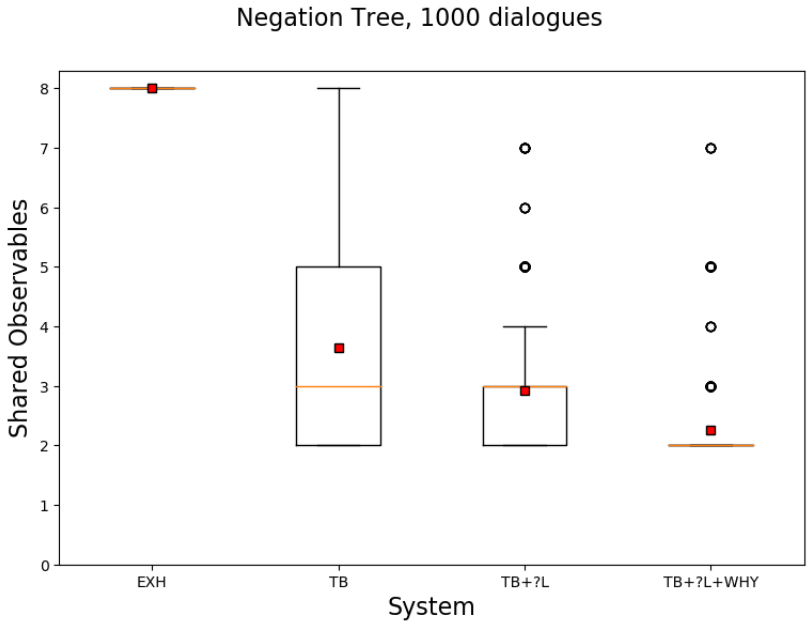
C.2.2 Multi-Rule Trees

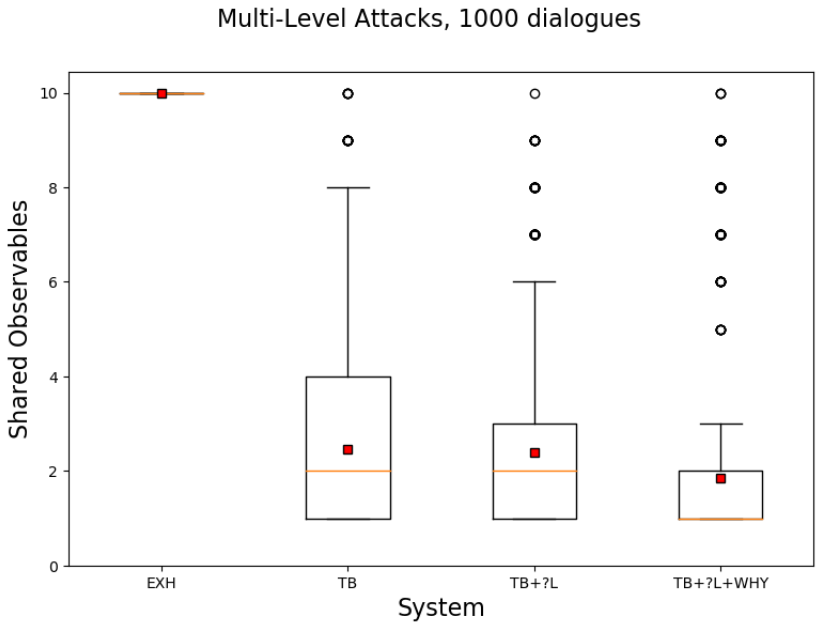




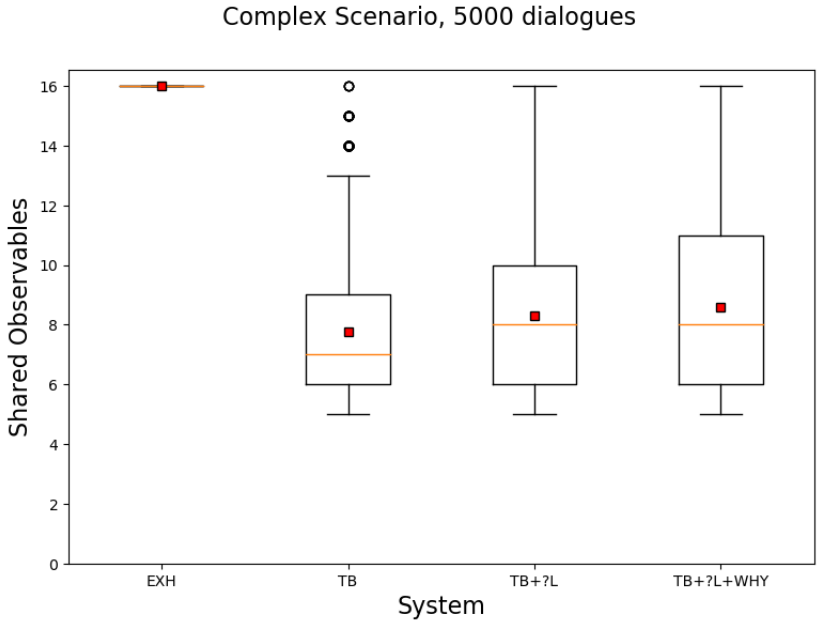
C.3 Conflicting Argument Instantiation

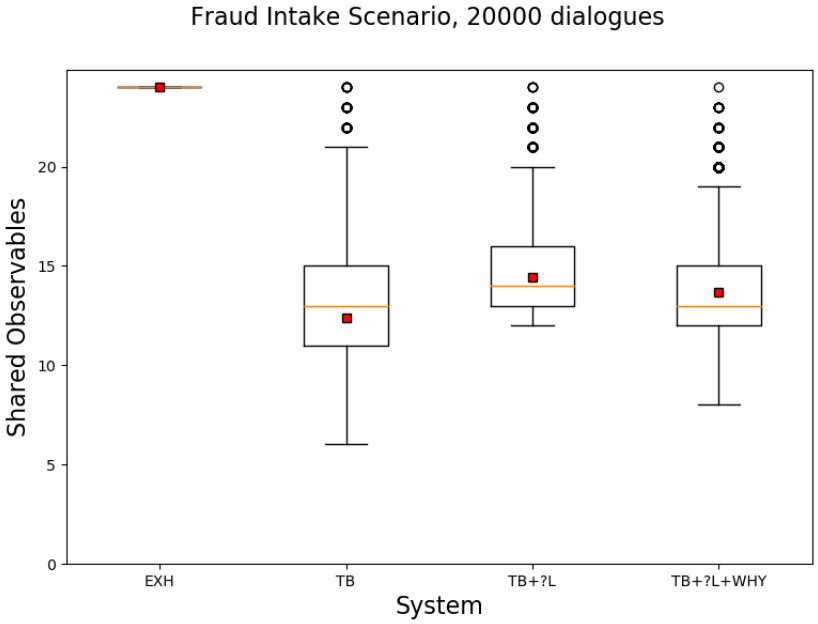
C.3.1 Ambiguous Setups





C.3.2 Complex Setups




Fraud Intake Scenario, 20000 dialogues
Shared Observables
0
5
10
15
20
EXH
TB
TB+?L
TB+?L+WHY
System