Universiteit Utrecht

Artificial Intelligence
Graduate School of Natural Sciences
Intelligent Systems

Asking useful questions in information gathering dialogues

W.D. Kumeling

1. Reviewer Floris Bex
Department of Information and Computing Sciences
Utrecht University

2. Reviewer Henry Prakken
Department of Information and Computing Sciences
Utrecht University

Supervisor Floris Bex

April 5, 2019

Abstract

Inquiry dialogues exist to gather data about a given topic. The dialogue systems of Black and Hunter (2009) generate such dialogues. Those dialogues are, however, large, due to the exhaustive strategy. The limited commitment, limited dialogue, smart dialogue and smart original strategy are proposed as modifications of that strategy to reduce the exhaustiveness of these dialogues. These strategies are investigated in three ways. By simulating dialogues, the practical effect of the modifications is investigated. By proving soundness and completeness, the correctness of the strategies is investigated. Lastly, the effect of the problem domain, an intake with the police, is briefly investigated. The smart dialogue and the two limited strategies generate significantly smaller dialectical trees, but are not sound or complete. Moreover, the generated dialogues were generally *not* smaller. The smart original strategy was found to be sound and complete, but only a marginal improvement. This thesis provides, furthermore, evidence that the effectiveness of a strategy highly depends on the knowledge base reasoned upon. It also demonstrates that building 'smart' strategies, that are also sound and complete, with no extra information is hard.

Contents

1 Introduction **4**

2 Argumentation **7**
2.1 Logic . . . 7
2.2 Arguments . . . 8
2.3 Dialectical tree . . . 10
2.4 Conclusion . . . 12

3 Dialogues **13**
3.1 Dialogue systems . . . 13
3.2 Types of dialogue . . . 14
3.3 The intake as a dialogue . . . 14
3.4 Conclusion . . . 15

4 Strategies **16**
4.1 Exhaustiveness and planning . . . 16
4.2 Machine learning . . . 17
4.3 User models . . . 18
4.4 Relevancy . . . 19
4.5 Conclusion . . . 20

5 An inquiry dialogue system **21**
5.1 Order of operations . . . 21
5.2 Moves . . . 22
5.3 Dialogues . . . 23
5.4 Effect rules and query store . . . 24
5.5 Turn taking . . . 25
5.6 Termination . . . 25
5.7 Dealing with sub-dialogues . . . 26
5.8 Knowledge base . . . 26
5.9 Argument inquiry . . . 27
5.10 Warrant inquiry . . . 28
5.11 Conclusion . . . 30

6 Soundness, completeness and the exhaustive strategy **32**
6.1 Soundness and completeness . . . 32
6.1.1 Argument inquiry . . . 33
6.1.2 Warrant inquiry . . . 33
6.2 Exhaustive strategy . . . 34

6.3 Conclusion . . . 35

7 Modified exhaustive strategies 36
7.1 Commitment store versus dialogue . . . 37
7.1.1 One argument per proposition . . . 37
7.1.2 Checking the status . . . 38
7.2 The strategies . . . 39
7.2.1 The limited strategies . . . 40
7.2.2 Smart strategies . . . 40
7.3 Conclusion . . . 41

8 Proving soundness and completeness 43
8.1 Limited strategies . . . 43
8.1.1 Argument inquiry . . . 43
8.1.2 Warrant inquiry . . . 44
8.2 Smart strategy . . . 47
8.2.1 Soundness . . . 47
8.2.2 Completeness . . . 49
8.3 Conclusion . . . 50

9 Experiments 52
9.1 Research questions . . . 53
9.2 Methods . . . 53
9.2.1 The data sets . . . 53
9.2.2 Splitting the data sets . . . 55
9.2.3 The implementation . . . 57
9.3 The setup . . . 60
9.4 Results . . . 61
9.4.1 Ambiguity handling . . . 62
9.4.2 Team defeat . . . 66
9.4.3 Floating conclusions . . . 69
9.5 Discussion of experiments . . . 72
9.5.1 General setup . . . 72
9.5.2 Effects of the modified strategies . . . 73
9.5.3 Influence of the intake setting . . . 74
9.5.4 Behaviour of the modified strategies . . . 75
9.5.5 Limitations . . . 76

10 Discussion 78
10.1 Discussion of results . . . 78
10.2 Applicability of DeLP and Black and Hunter . . . 80
10.3 Contribution . . . 81

11 Conclusion 82
11.1 Future work . . . 84

References 85

A Intake setting **87**
A.1 Ambiguity 87
A.2 Team defeat 89
A.3 Floating conclusions 91

B Root arguments in the intake setting **93**
B.1 Ambiguity 93
B.2 Team defeat 94
B.3 Floating conclusions 94

C Random setting **95**
C.1 Ambiguity 95
C.1.1 Size of dialogues 95
C.1.2 Size of dialectical trees 97
C.2 Team Defeat 99
C.2.1 Size of dialogues 99
C.2.2 Size of dialectical trees 101
C.3 Floating conclusions 103
C.3.1 Size of dialogues 103
C.3.2 Size of dialectical trees 105

Chapter 1

Introduction

How can agents communicate effectively in order to accomplish a given task? In general, agents communicate to exchange information. One form of communication is via dialogues. In dialogues, agents respond to each other until some goal is met. By exchanging information in the form of arguments and propositions, agents engage in what are called argumentation dialogues. Those arguments and propositions are used by agents, for example, to exchange reasons for their beliefs try to convince their opponent, or do other things. This thesis will assume that such dialogues are between two agents.

Argumentation dialogues are formed by agents uttering moves. For example, let two agents argue over whether a certain incident involved fraud. To begin such a dialogue, one of the agents could utter $assert(FraudOccured)$. With this, the agent asserts that it believes that fraud took place. The other agent may choose to reply with $why(FraudOccured)$, asking for the reason that the other agent believes that there was a fraud. Here, $why(FraudOccured)$ is the content, or locution, of the move. Moves contain, besides a content, also the agent who uttered it and some information about why the move was uttered. Moreover, locutions like $why(FraudOccured)$ are of the form $p(c)$, where p is a performative and c an argument or proposition.

By uttering (or replying to) locutions, agents are making moves. By replying to each other's moves until some goal is met, a dialogue is formed. This thesis focuses on argumentation dialogues that are formed according to rules set out by dialogue systems. Dialogue systems contain rules on when and which locutions may be uttered, how locutions affect the state of agents, when the dialogue ends, and how agents take turns.

Argumentation dialogues are used to solve conflict. This conflict may entail various things, such as agents having a conflict over resource allocation, or simply disagreeing over some sort of topic. According to Prakken (2017) research on argumentation dialogues can be divided into two areas. Firstly, research on the dialogues themselves, on their formal definition and properties. Secondly, research on the behaviour of agents within those dialogues. In general, the overarching goal of the research is that argumentation dialogues should "*promote fair and effective resolution of conflicts*" (Prakken, 2017).

This thesis treats argumentation dialogues wherein there exists a conflict on the provability of a proposition (called inquiry dialogues). This means that the goal of the dialogue is to cooperatively exchange information in order to

(dis)prove a proposition.

Inquiry dialogues are used because of the Intelligence Amplification for Cybercrime (IAC) project. The Dutch national police have since 2016 with Floris Bex and Bas Testerink been involved in the IAC project to enhance the quality of online reports of crimes. The focus lies on automating the reporting of fraud (e.g. fraud on websites like 'marktplaats.nl').

The project has chosen an agent-based approach (Bex et al., 2016). One aspect of the system is the intake (software) agent, with which the complainant communicates using a web interface. The intake agent collects the responses from the complainant, asks him questions and shares the answers with a dialogue manager.[1] The agent asks the complainant for more information if needed and interprets the information given. However, choosing what to ask the complainant is not a trivial task. For one, the intake should be as short as possible. The police rather have a lower quality report than an unfinished one. Secondly, the questions should be relevant. If the complainant says he was the victim of a fraud, he should not be asked about whether he was the fraudster. Note that these requirements may conflict with one another.

IAC is therefore modelling a dialogue in which the goal is to find whether an acceptable argument for fraud exists. Inquiry dialogues have a similar goal. As such, this thesis therefore treats inquiry dialogues in an effort to find out how such dialogues can be made 'smarter'.

Making dialogues 'smarter' is not straightforward. Normally it is already hard for an agent to determine the best thing to do any time it needs to communicate. The same holds true for agents communicating to each other. An agent therefore would like to follow a strategy which helps it communicate more efficiently and accurately. Coming up with such a strategy is not trivial, as each specific situation often requires different things of a strategy. Moreover, there is still relatively few work done on strategies for argumentation dialogues. The work that is done, often does so in the context of a persuasion dialogue (i.e. when two agents try to convince each other) (Prakken, 2017, p. 2236).

However, the agent in the inquiry dialogue *needs* to have some sort of strategy for deciding on a relevant question. The aim of this research is therefore to improve on existing work on inquiry dialogues, and find out how such dialogues can be navigated more smartly by agents. In the literature several approaches are used in order to create smart strategies.

Firstly, the idea is to represent the state of the dialogue in such a way that a planner can find an 'optimal' strategy, which they define as a strategy using which the proponent has got the highest possible chance of making the dialogue succeed. In inquiry dialogues, for example, the goal would be to find an acceptable argument as quickly as possible. Black et al. (2017) used propositional planning to find optimal strategies for persuasion dialogues. Alahmari et al. (2017), on the other hand, used Q-learning to let the agent find an optimal strategy on his own. The downside of such an approach is, however, that a training phase must be held before going into a dialogue.

Another approach is using a user model, such that an agent can make better decisions, using a model of his opponent. Such a user model can be build upon previous experiences, the history of the dialogue or other things. Hadoux et al. (2015) put this information into the knowledge base itself (using probabilistic

[1] See for an in depth explanation of the system Bex et al. (2016).

rules), such that an external tool (after converting the dialogue into a MOMDP) could 'plan' an optimal strategy. Hadjinikolis et al. (2013) and Rienstra et al. (2013) both tried to predict the state of the opponent, by adding recursive user models. Hadjinikolis et al. learned the user model from previous experiences, while Rienstra et al. kept track of the current state of the dialogue.

Note that these approaches are mostly defined upon persuasion dialogue systems. Persuasion dialogues, however, differ from inquiry dialogues in their goal. Instead of cooperatively trying to prove a proposition, the goal is to convince the opponent of a certain viewpoint. Due to this differing goal, it may not be trivial to apply the discussed approaches to another types of dialogue systems. Therefore, it is hard to know beforehand whether a strategy will be effective upon another dialogue system.

But what should the inquiry dialogues that will be used look like? As already mentioned, the preliminary goal of IAC is to automate online intakes. The inquiry dialogues used will thus be loosely modelled after intakes. This thesis will take an existing inquiry dialogue system, and (try to) improve on its corresponding strategy.

For this, the dialogue systems as described by Black and Hunter (2009) are used. Black and Hunter defined two dialogue systems, for argument and warrant inquiry dialogues. In an argument inquiry dialogue the goal is to find an argument for a proposition. In warrant inquiry dialogues an acceptable argument is sought.

Black and Hunter also defined a corresponding strategy, called the exhaustive strategy. With this strategy, agents will first assert any known arguments, then open sub-dialogues, or otherwise try to close the current dialogue.

The exhaustive strategy generates, however, dialogues that are quite long. Therefore, two modifications on the exhaustive strategy are devised to reduce the exhaustiveness of these dialogues. Those strategies are tested using simulations. It is also investigated whether they are sound and/or complete as defined by Black and Hunter.

The modifications on the exhaustive strategy are mainly tested in a setting inspired by the intake. Herein, it is assumed that the proponent knows all the rules, and the opponent all facts, to model the asymmetricity of the intake. This setting is called the intake setting.

The influence of this intake setting is also investigated by simulating the dialogues in a setting with randomized knowledge bases.

Chapter 2 will explain structured argumentation. Chapter 3 explains the theory behind dialogues, a way to formally model communication between agents, and what is known about the intake as a dialogue. Next, Chapter 4 describes related work on strategies which may be relevant to an intake dialogue. In Chapter 5 the used dialogue system is explained in detail, with Chapter 6 describing the exhaustive strategy. Chapters 7 and 8 respectively define the modified strategies investigated in this thesis, and investigate whether they are sound or complete. These strategies are used to simulate dialogues, as specified in Chapter 9, wherein also the results of the simulations are discussed. The results from the proofs and simulations are combined and discussed in Chapter 10. Lastly, Chapter 11 concludes the thesis.

Chapter 2

Argumentation

Before discussing how agents are assumed to communicate with another, first we need to define *what* information can be exchanged. Specifically, the argumentation logic that is used must be defined. This chapter will discuss the logic used, and how it is used to check the acceptability of arguments. Together, they form what is called an argumentation framework. Argumentation frameworks are used to enable reasoning with inconsistent knowledge bases and help decide between multiple (conflicting) scenarios.

Two often used argumentation frameworks are abstract argumentation frameworks and structured argumentation frameworks. While abstract argumentation frameworks merely look at arguments and their relations (often based upon Dung (1995)), structured argumentation frameworks also specify the contents of the arguments reasoned about. This chapter explains a modification of DeLP, adopted from Black and Hunter (2009).

First, the modified logic itself it treated. Next, the relation of the logic to arguments is discussed. Finally, it is explained how arguments are assumed to interact with one another.

2.1 Logic

An adoption of DeLP (Garcia & Simari, 2004), using only defeasible facts and rules, is used as the underlying logic. This adoption of DeLP assumes that defeasible facts are atoms, not predicates as in normal DeLP.

First is defined what defeasible facts and rules are, and how they are notated.

Definition 2.1.1. *Defeasible facts* are literals, denoted by α, β, etc. *Defeasible rules* are of the form $\beta_1 \wedge \cdots \wedge \beta_n \Rightarrow \alpha$ where $\beta_1, \ldots, \beta_n$ and α are defeasible facts.

Next, the topic language is defined. Generally, the topic language defines what agents are able to talk about in a dialogue. The topic language for DeLP consists of defeasible facts and rules, coupled with preference levels.

Definition 2.1.2. The *topic language* L_t consists of beliefs, denoted by the pair (ϕ, L), where ϕ is a defeasible fact or rule (i.e. a proposition) and L a label that denotes the preference level of the belief. $plevel(\phi)$ gives the preference level of ϕ, such that $plevel(\phi) = L$.

It is assumed that a lower preference level is more preferred. Therefore, if, for example, the beliefs $(a \Rightarrow b, 4)$ and $(a \Rightarrow c, 2)$ exist, then $a \Rightarrow c$ is more preferred than $a \Rightarrow b$. The preference levels of beliefs will later be used to compare arguments on their weakest links. First needs to be defined how new knowledge can be derived.

New knowledge is defeasibly derived from a minimal set of premises. This way, all sets of premises are finite.

Definition 2.1.3. A *defeasible derivation* of ϕ from Φ, denoted $\Phi \mathrel{|\!\sim} \phi$, is a finite sequence $[\beta_1, \ldots, \beta_n]$ such that β_n is the defeasible fact ϕ and each literal β_m $(1 \leq m \leq n)$ is in the sequence because:

1. (β_m, L) is a belief in Φ with β_m being a defeasible fact, or;
2. $\exists(\alpha_1 \wedge \cdots \wedge \alpha_j \Rightarrow \beta_m, L') \in \Phi$ such that every α_i $(1 \leq i \leq j)$ is an element β_k, with $k < m$, preceding β_m in the sequence $[\beta_1, \ldots, \beta_n]$.

Example 2.1.4. *Let* $\Phi = \{(p,1), (r,1), (p \Rightarrow q, 1), (q \wedge r \Rightarrow s, 1)\}$. *Then,* $\{p, r, p \Rightarrow q, q \wedge r \Rightarrow s\} \mathrel{|\!\sim} s$, *since it is equivalent to the sequence* $[p, p \Rightarrow q, r, q \wedge r \Rightarrow s, s]$.

2.2 Arguments

Defeasible derivation is used to form arguments. If a belief is derived from a set of premises, then there exists an argument with that set of premises supporting the derived belief.

Definition 2.2.1. An *argument* A constructed from a set of beliefs $\Psi \subseteq L_t$ is a tuple $\langle \Phi, \phi \rangle$ where ϕ is a defeasible fact, and Φ a set of beliefs such that:

1. $\Phi \subseteq \Psi$;
2. $\Phi \mathrel{|\!\sim} \phi$;
3. $\forall \phi, \phi'$ s.t. $\Phi \mathrel{|\!\sim} \phi$ and $\Phi \mathrel{|\!\sim} \phi'$, it is not the case that $\phi \cup \phi' \bot$;
4. There is no subset of Φ that satisfies 1–3.

An argument is therefore a tuple consisting of a minimally consistent set premises from which the conclusion can be defeasibly derived. The conclusion is the second element of the tuple.

The following are definitions to more easily talk about arguments.

Definition 2.2.2. Let $A = (\Phi, \phi)$ be an argument. Φ is then called the *support* of the argument and is denoted by $support(A)$. ϕ is called the *conclusion* of the argument and is denoted by $conc(A)$. Argument A_1 is a sub-argument of A iff $support(A_1) \subseteq support(A)$. The *argumentation theory* of a set beliefs Ψ, denoted by AT_Ψ, contains all arguments that can be constructed from Ψ.

Example 2.2.3. *Let* Φ *be as defined in Example 2.1.4. The following is then an argument:* $A = (\{p, r, p \Rightarrow q, q \wedge r \Rightarrow s\}, s)$. $support(A) = \{p, r, p \Rightarrow q, q \wedge r \Rightarrow$

$s\}$, *and* $conc(A) = s$. *The argumentation theory constructed from* Φ *is:*

$$\begin{aligned} AT_\Phi = \{&(\{p\}, p),\\ &(\{p, p \Rightarrow q\}, q),\\ &(\{r\}, r),\\ &(\{p, r, p \Rightarrow q, q \wedge r \Rightarrow s\}, s)\} \end{aligned}$$

Note that Black and Hunter use $\mathcal{A}(\Psi)$ to denote all arguments constructable from Ψ. Here, however, it is denoted by AT_Ψ. Furthermore, an argument is said to be a sub-argument if its support is a subset of another argument.

An argument on its own, however, is not very interesting. A set of arguments, and their interactions are more interesting, since arguments may conflict.

Definition 2.2.4. Two arguments A_1 and A_2 *conflict* iff $conc(A_1) \cup conc(A_2) \vdash \bot$.

Arguments therefore conflict if their conclusions conflict. Using this notion of conflict, the attack relation between arguments can be defined.

Definition 2.2.5. Let A_1, A_2, A_3 be arguments such that A_3 is a sub-argument of A_2. A_1 *attacks* A_2 at A_3 iff A_1 conflicts with A_3.

Note that using this definition, arguments can be rebut as well as be undercut.

Now that the attack relation is defined, there needs to be defined when an attack succeeds (i.e. results in defeat). This will be done via the notion of preference levels.

Definition 2.2.6. Let A be an argument. The *preference level* of the argument A, denoted $plevel(A)$, is equal to $plevel(\phi)$ such that:

1. $\phi \in support(A)$;
2. $\forall \phi' \in support(A) : plevel(\phi') \leq plevel(\phi)$.

The preference level of an argument is that of its *least* preferred proposition from its support. As conflicting arguments will be compared using their preference levels, this results in a weakest link ordering.

Definition 2.2.7. Let A_1, A_2, A_3 be arguments such that A_3 is a sub-argument of A_2, and that A_1 attacks A_2 at A_3. A_1 is a *proper defeater* for A_2 iff $plevel(A_1) < plevel(A_3)$. A_1 is a *blocking defeater* for A_2 iff $plevel(A_1) = plevel(A_3)$.

Example 2.2.8. *Let* $\Psi = \{(s, 2), (r, 1), (p, 2), (p \Rightarrow q, 2), (s \Rightarrow \neg q, 2), (r \Rightarrow \neg q, 1)\}$. *Also, let* $A = (\{r, r \Rightarrow \neg q\}, \neg q)$, $B = (\{s, s \Rightarrow \neg q\}, \neg q)$ *and* $C = (\{p, p \Rightarrow q\}, q)$. *Then* $plevel(A) = 1$, $plevel(B) = 2$ *and* $plevel(C) = 2$. *Here,* A *is a proper defeater of* C, *as* $plevel(A) < plevel(C)$. *However,* C *is* not *a defeater of* A, *as* $plevel(C) > plevel(A)$. *Furthermore,* B *is a blocking defeater of* C, *as* $plevel(B) = plevel(C)$.

An attack therefore succeeds if the preference level of the attacking argument is equal or lower than the attacked argument. The defeat is considered to be proper if the preference level is lower, otherwise the defeat is said to be blocked.

Given arguments and a notion of defeat, a way is needed to analyse the status of arguments. For example, if in a set of arguments A, B, C B defeats C, but A also defeats B, is argument B then still credible as a defeater of C?

2.3 Dialectical tree

The status of arguments is checked using so-called dialectical trees. Black and Hunter use dialectical trees to check the acceptability of an argument in warrant inquiry dialogues. A path from the root of a dialectical tree to one of its leaves is called an argumentation line.

Definition 2.3.1. If $\Delta = [\langle\Phi_0, \phi_0\rangle, \langle\Phi_1, \phi_1\rangle, \langle\Phi_2, \phi_2\rangle, \dots]$ is a sequence of arguments such that each element of the sequence $\langle\Phi_i, \phi_i\rangle$ is a defeater of its predecessor $\langle\Phi_{i-1}, \phi_{i-1}\rangle$, then Δ is an *argumentation line.* Δ is an *acceptable argumentation line* iff

1. $|\Delta| < \infty$;
2. $\langle\Phi_0, \phi_0\rangle \cup \langle\Phi_2, \phi_2\rangle \cup \langle\Phi_4, \phi_4\rangle \cup \cdots \nvdash \bot$ and $\langle\Phi_1, \phi_1\rangle \cup \langle\Phi_3, \phi_3\rangle \cup \langle\Phi_5, \phi_5\rangle \cup \cdots \nvdash \bot$;
3. No argument $\langle\Phi_k, \phi_k\rangle$ appearing in Δ is a sub-argument of an argument $\langle\Phi_j, \phi_j\rangle$ that appears earlier in Δ $(j < k)$;
4. $\forall i$ s.t. $\langle\Phi_i, \phi_i\rangle$ is a blocking defeater for $\langle\Phi_{i-1}, \phi_{i-1}\rangle$, if $\langle\Phi_{i+1}, \phi_{i+1}\rangle$ exists, then $\langle\Phi_{i+1}, \phi_{i+1}\rangle$ is a proper defeater for $\langle\Phi_i, \phi_i\rangle$.

An argumentation line is therefore a sequence in which each argument defeats its predecessor. It is an acceptable line if the sequence is finite, if the players that constructed the line did not contradict themselves, if no argument is repeated, and if the argumentation line only consists of proper defeaters.

Definition 2.3.2. Let $\Psi \subseteq L_t$ be a set of propositions in dialogue d, and A_0 an argument such that $A_0 \in AT_\Psi$. A *dialectical tree* for A_0, constructed from Ψ, denoted $T(A_0, \Psi)$, is defined as follows

1. The root of the tree is labelled with A_0;
2. Let N be a node of the tree labelled A_n, and let $\Delta_i = [A_0, \dots, A_n]$ be the sequence of labels on the path from the root to node N. Let arguments $B_1, B_2, \dots, B_k$ be all defeaters for A_n that can be formed from AT_Ψ.

   For each defeater B_j $(1 \leq j \leq k)$, if the argumentation line $\Delta'_i = [A_0, \dots, A_n, B_j]$ is an acceptable argumentation line, then the node N has a child N_j that is labelled B_j. If there is no defeater for A_n or there is no B_j such that Δ'_j is acceptable, then N is a leaf node.

A dialectical tree is therefore constructed given a root argument, and a set of propositions. It furthermore is 'complete', in the sense that no new defeating arguments can be added to any argumentation line from the root argument to any leaf.

Example 2.3.3. *Let Ψ, A, B and C be the set of beliefs and the arguments as defined in Example 2.2.8. The dialectical tree $T(C, \Psi)$ is shown in Figure 2.3.1. C is the root of the tree, with arguments A and B defeating C.*

The tree is used to find out whether an argument is warranted (i.e. acceptable) given a set of beliefs. In Chapter 5 dialogues are defined in which dialectical trees are compared. Therefore, a notion of equality is needed.

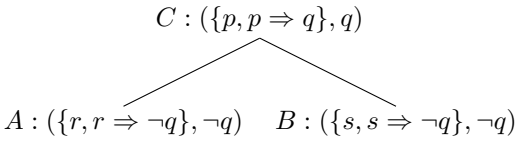


Figure 2.3.1: The dialectical tree $T(C, \Psi)$.

Definition 2.3.4. Two dialectical trees T_1 and T_2 are *equal* iff

1. The root of T_1 is A_0, and the root of T_2 is A_0;
2. If node N_1 is a node in T_1 and $[A_0, \ldots, A_n]$ is the argumentation line in T_1 from A_0 to N_1, then there is a node N_2 in T_2 such that $[A_0, \ldots, A_n]$ is in T_2 the argumentation line from A_0 to N_2;
3. If node N_2 is a node in T_2 and $[A_0, \ldots, A_n]$ is the argumentation line in T_2 from A_0 to N_2, then there is a node N_1 in T_1 such that $[A_0, \ldots, A_n]$ is in T_1 the argumentation line from A_1 to N_1.

Two dialectical trees are therefore equal if they share the same root node, and if each argumentation line of one tree has an equivalent in the other tree.

A dialectical tree, however, says nothing about the status of the contained arguments. For that, the arguments need to be assigned a status, i.e. marked. It is using this mark that the acceptability of an argument can be determined.

Definition 2.3.5. Let $T(A, \Psi)$ be a dialectical tree, with A as an argument and $\Psi \subseteq L_t$. The corresponding *marked dialectical tree* is obtained by marking every node as follows

1. All leaves in $T(A, \Psi)$ are marked *in*;
2. If N is a node of $T(A, \Psi)$ and N is not a leaf node, then N is marked *in* iff all its children are marked *out*. N is marked *out* iff at least one of its children is marked *in*.

The marking of an argument in the tree determines its status. This definition is modified to refer to AT_Ψ instead of $\mathcal{A}(\Psi)$. Also, instead of using U (undefeated) or D (defeated) as a status, *in* and *out* are used. They still mean the same thing, however.

Definition 2.3.6. The *status* of an argument A given a set of rules and facts $\Psi \subseteq L_t$, and the corresponding argumentation theory AT_Ψ, is returned by the function $status : AT_\Psi \times \mathcal{P}(\Psi) \to \{in, out\}$. $status(A, \Psi) = in$ iff A is marked *in* in $T(A, \Psi)$. Otherwise, $status(A, \Psi) = out$ (i.e. if A is marked *out* in $T(A, \Psi)$.

Example 2.3.7. *Continuing the example from Example 2.3.3, Figure 2.3.2 shows the marked dialectical tree $T(C, \Psi)$. Here, argument C is marked* out*, as all its defeaters are* in*.*

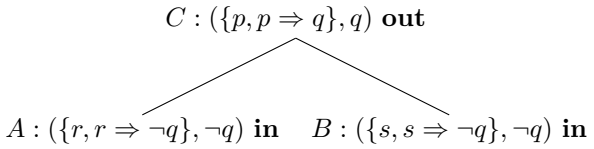


Figure 2.3.2: The *marked* dialectical tree $T(C, \Psi)$.

2.4 Conclusion

The used logic, a variant of DeLP, therefore supports defeasible facts and rules. Arguments attack each other on conclusions, or premises. Conflict is resolved via preference levels. Furthermore, dialectical trees are used to check the acceptability of arguments.

Take for example Figure 2.3.2. There, the dialectical tree consists of three arguments, A, B and C. Argument C is the root argument and is *out*, since it is attacked by A and B. Arguments A and B are attacking C since A and B's conclusion, $\neg q$, is a complement of C's conclusion, q.

As mentioned, a dialectical tree is used to check the acceptability of the root argument. A dialectical tree must, however, be built by exchanging arguments. Chapter 5 will explain how, in the dialogue system used in this thesis, this is done. The next chapter will explain how arguments are used in a dialogue.

Chapter 3

Dialogues

As mentioned in the introduction, arguments can be used by agents in dialogues (so-called argumentation dialogues) to exchange information. Those dialogues may have different types of purposes. Agents may, for example, want to convince an opponent of the truth or a given statement, or may simply seek information. In other words, agents may have different types of goals with a dialogue.

This chapter will discuss types of dialogues that are applicable to intakes, and discuss what kind of dialogue an intake is. However, first is discussed how dialogues are defined (i.e. using dialogue systems).

3.1 Dialogue systems

Dialogues are formed when agents respond to each others moves. However, dialogues are bound by rules. Not every move is therefore always a valid response to another move. For example, it is nonsense to ask someone the reason why something is the case, when that person just asked you the same question. Moreover, moves have got side effects. Asserting an argument, for example, typically commits you to that argument. Also, it is useful to know when a dialogue is done, and how turn taking takes place.

Dialogue systems define sets of valid dialogues. As such, they define *how* agents are allowed to exhange information. It is assumed that dialogue systems contain turn taking rules, rules about moves, termination rules and effect rules. The turn taking rules define at what points in the dialogue agents are allowed to utter moves (i.e. who's turn it is). The termination rules define at what point a dialogue is done (i.e. terminated). The effect rules determine the side effects of the moves. Lastly, the rules about moves define per move whenever a move is legal to be uttered.

Those rules define a set of dialogues that are valid for one particular dialogue system. Dialogue systems are therefore designed with specific goals in mind. For example, the dialogue systems as described by Black and Hunter (2009) say that only dialogues in which either an argument is sought or an argument's validity is checked are valid.

Note that dialogue systems can be *used* to generate dialogues, although they do not specify how. Instead, a 'strategy' does this. In Chapter 4 strategies are discussed.

The next section will discuss which purposes dialogues may serve. The focus lies on purposes that may be relevant to intakes. The last section will investigate what types of dialogue an intake contains.

3.2 Types of dialogue

Dialogues can be of different types, which is visible in the different types of interactions agents can have. Agents may, for example, communicate in order to share information, or to allocate resources. This section gives a short overview of the influential typology set out by Walton and Krabbe (1995) which was expanded upon in Walton and Macagno (2007).

Walton and Krabbe (1995) divide agent interaction in six types, and specify for each dialogue type the initial situation, the goal of the interaction and the participant's goals. Of the six types defined by Walton and Krabbe only persuasion, information-seeking and inquiry dialogues are covered here. The other types are not relevant in case of the intake.

In persuasion dialogues one agent tries to persuade the opponent of some proposition. The opponent is convinced of a proposition opposite to the one the agent believes in. Both the agent and the opponent try, using arguments, to convince the other of the truth of their proposition.

In an information-seeking dialogue an agent lacks information, and is talking to an expert, or someone who has information. The agent's goal is to get the information he needs from the 'expert'. Note that information-seeking dialogues are asymmetrical, the agent knows less about the topic of the dialogue than the expert. A successful information-seeking dialogue therefore ends with the questioning agent knowing more about the topic of the dialogue. The goal for the agent is thus not to win, but to retrieve information.

Inquiry dialogues are a collaborative effort. The goal of both parties is to either prove a proposition or show the impossibility of proving it. A successful inquiry dialogue is therefore one in which both parties agree upon the same conclusion. Note that there is no conflict of opinion, both parties merely want to collaboratively determine whether a proposition is provable.

It is worth noting that few dialogues are of one type. Often dialogues incorporate a sub-dialogue of a different type to help the main dialogue further along. For example, if an agent wants to negotiate for a price, it may need to know first what the other agent offers. In that case, the negotiation dialogue will also contain an information-seeking sub-dialogue.

The next section will treat the intake and discuss what types of dialogues are typically found within.

3.3 The intake as a dialogue

As explained in the previous section, few dialogues are of one type of dialogue. Most dialogues contain elements of multiple dialogue types. So as what dialogue type can the intake be classified? Unfortunately, no literature was found on a possible classification for intakes.

Nevertheless, Walton (2003) investigated interrogation as a subclass of information seeking dialogues. Furthermore, Arioua and Croitoru (2015) formalized

explanatory dialogues, Mason (2016) and van Charldorp (2014) analysed what happens during and right after a police interrogation.[1] in short, there does not exist a clear answer as to what dialogue type an intake encompasses.

The police, however, clearly, *seeks information* from the complainant during an intake. Also, both parties engage in *an inquiry* to find out what the final report will be. Furthermore, it is reasonable to assume that a complainant can be stubborn. For example, he could be stubborn to accept that the crime he wants to report is merely a civil case, and not a criminal offense. In such a case the complainant first needs to be *persuaded* that no swindle took place. The intake therefore seems to be a mix of information-seeking, inquiry and persuasion dialogue.

Important to note here is that even though the intake is not an interrogation, it shares some of its goals with that of an interrogation. In both cases the wanted outcome is a clear fact-based chronology of what happened (a written story). Or, as observed by Charldorp: *"The written story is a more factual, detailed, precise and intentional story on paper constructed according to the institutional perspective of the officer."* (van Charldorp, 2014).

An intake can therefore be assumed to be a persuasion, information-seeking and/or inquiry dialogue. This thesis assumes, however, that in principle both the complainant and the police officer want a complete report and that the complainant is willing to share information. In other words, the intake is assumed to be a dialogue in which both parties cooperatively share information and try to find out what the final report will be. Therefore, this thesis treats the intake as an inquiry dialogue. It is also assumed that the police officer tries to gain as much accurate information as possible, while keeping the dialogue as short as possible.

Note that the question is still open as to how an intake should be kept as short and accurate as possible. Furthermore, it is not specified what specific requirements an intake dialogue has got. This thesis investigates the first question, assuming that the intake is an inquiry dialogue.

3.4 Conclusion

Dialogues are therefore formed using sets of rules. A dialogue system contains those rules. Moreover, in dialogues agents respond to each others moves. A dialogue is therefore assumed to be a list of moves.

The intake is assumed to be an inquiry dialogue. In an inquiry dialogue agents cooperatively try to prove a proposition. In case of the intake, the goal is to first find an argument for the topic, and then check its acceptability.

Dialogue systems on their own, however, are not capable of generating dialogues. Instead, it needs to be specified how a dialogue is generated, or what strategy is followed for that to happen. The next chapter will explain how, given a dialogue system and a strategy, dialogues are generated.

[1] i.e. how a police officer extracts a factual account of what happened out of a spoken story.

Chapter 4

Strategies

Dialogue systems do not specify how to generate dialogues. They therefore also do not specify how agents can accomplish their goals as soon as possible within a dialogue. Instead, agents use strategies to generate dialogues from dialogue systems.

Strategies choose, from a list of currently valid moves, one move. A good strategy will try to reach an agent's goal as soon as possible.

How a strategy tries to fulfil the agent's goal differs. Some strategies only look at the history of the current dialogue, while others also look at earlier dialogues (experience). Often the agent not only keeps track of its own state, but also tries to predict the opponent's state. This information can also be used by a strategy. Due to these things a strategy is (per definition) tied to a dialogue system.

This chapter describes papers with strategies that are either using a dialogue of a similar type to that of the intake, or which show a general approach which may be easy to adapt to an inquiry dialogue. The focus lies on describing the aim and inner workings of the techniques, and in what capacity they achieve their goal. The goal is to discuss possible approaches for a 'good' strategy for intakes.

Note that most strategies described here are defined upon persuasion dialogues. As mentioned in the introduction, the goal of persuasion dialogues differs from inquiry dialogues. Therefore, even if a strategy is very successful in one dialogue system, it does not have to be successful in another. The strategies that are defined upon persuasion dialogues may thus turn out to be ineffective on inquiry dialogues.

4.1 Exhaustiveness and planning

This section treats two approaches as described by Black and Hunter (2009) and Black et al. (2017). The first paper tries to generate dialogues which only generate valid answers, and which give an answer if one exist. Black et al. (2017) try to plan ahead, and use a propositional planner to come up with an optimal strategy.

Black and Hunter (2009) sought to find a strategy to generate dialogues for the medical domain. They discuss two subtypes of inquiry dialogue, namely

argument inquiry and warrant inquiry dialogues. In argument inquiry dialogues agents share beliefs to construct an argument for a given proposition. Warrant inquiry dialogues go further, since agents try to see whether a given proposition is acceptable. Black and Hunter (2009) use for their knowledge representation a defeasible variant of Defeasible Logic Programming (DeLP).[1] The variant used is explained in detail in Chapter 2. As in the medical domain a wrong decision can have huge consequences, Black and Hunter (2009) sought a sound and complete strategy. Such a strategy is guaranteed to give only arguments derivable from the agents' joint knowledge base (soundness) and to have generated an exhaustive dialogue (one in which each existing argument is uttered, i.e. completeness). Their 'exhaustive strategy' fulfils both properties, and amounts to first asserting all relevant beliefs, then open for each belief a sub-dialogue and lastly close the current dialogue.[2]

For persuasion dialogues Black et al. (2017) used propositional planning as a strategy. They only assume abstract argumentation as their knowledge representation and leave the contents of the arguments unspecified. Arguments are evaluated using grounded semantics. The agent's goal in the argumentation is to have its goal arguments to be acceptable at the end of the dialogue, under the grounded semantics. Black et al.'s aim was to find, using a propositional planner, a strategy that would be "*successful no matter which arguments the opponent asserts.*"(Black et al., 2017). The propositional planner is used to find an optimal 'simple strategy', i.e. one in which each successive move changes the acceptability of the root argument of the dialogue. This is done with an uncertain opponent model (ϵ, p) where ϵ is a set of opponent models with each model representing a possible knowledge base, and with p denoting the perceived likelihood that the opponent can be represented by the opponent model. As, however, the success of a strategy also depends on the actions of the opponent, few strategies succeed 100% of the time. Black et al. therefore searched for a strategy which had got the highest probability of success. By iteratively setting a higher probability as goal for the planner, a strategy can be found with the highest possible chance of succeeding.

4.2 Machine learning

Besides using a propositional planner, machine learning can also be used to find 'good' strategies. An idea is to let the planning take place before the agent is engaged in a dialogue, via Q-learning.

Alahmari et al. (2017) have taken such an approach. They tried to make an autonomous learning agent, which can find an optimal strategy on its own, using Q-learning. To keep the knowledge representation simple, they adopted abstract argumentation as their knowledge representation, and grounded semantics to determine the acceptability status of arguments. Alahmari et al. use a persuasion dialogue. Agents are expected to take turns by moving arguments that attack the previous argument. The dialogue ends when no further move is possible, and the winner of the game is the agent that made the last move. Besides this, Alahmari et al. place no additional restrictions on the dialogue or agents. Q-learning is a form of reinforcement learning wherein an agent with a

[1] See for a detailed description of DeLP (Garcia & Simari, 2004).
[2] See for more detail Chapter 6.

certain chance either follows the learned policy or does a random action. This provides the agent with rewards with which it can find an optimal policy (a strategy such that in any situation the expected sum of rewards is maximised). Alahmari et al. let the agent play against a random agent, a max-probability utility agent and a min-probability utility agent. Rewards were given out at the end of the dialogue. The reward of a dialogue was set to the number of acceptable arguments of the learning agent. By letting the agent play against the other agents, Alahmari et al. tried to teach the agent an optimal policy. The agent, however, only consistently outperformed the min-probability utility agent, which is not impressive. Alahmari et al. suspected that their limited state representation was responsible, as the agent could not differentiate effectively between different instances of the same argument.

4.3 User models

An opponent, however, does not have to be rational.[3] Taking this into account when designing an agent may improve that agent's performance. By using a stochastic user model, this can be done. Therefore, instead of having deterministic association rules, that tell for example that the user will say B if the agent says A, there will be a chance that the user will say B or something completely different.

Hadoux et al. (2015) took this approach. For a persuasion dialogue, in which the agent wants to persuade its opponent of a goal argument, their goal was to devise a strategy which would find an optimal sequence of moves, for a user that behaves stochastically. Moreover, they did not assume that the user behaves rationally or that he has specific initial knowledge. They tested their approach on a dialogical argumentation problem, which is an abstract argumentation problem with the possibility to move attack relations as well. The possible moves an agent can make are then defined by a set of rules of the form $r_j : prem_j \Rightarrow act_j$ where the rule r_j is applicable if $prem_j$ is derivable from the knowledge base or commitment store of the agent, after which act_j is applied. act_j is here a list of modifications on the knowledge base and commitment store of the agent. Hadoux et al. found an optimal sequence of moves by converting 'Argumentation Problems with Probabilistic Strategies' (APS) into 'Mixed Observation Markov Decision Processes' (MOMDP). An APS is an abstract argumentation framework in which the rules (such as r_j) used have probabilities. For example, the rule $r_j : e(c, f) \Rightarrow 0.5/\boxplus a(b) \vee 0.5/\boxplus a(c)$ is applicable if the attack relation $e(c, f)$ is in the knowledge base or commitment store, after which with chance 0.5 either b is added to the commitment store (denoted by $\boxplus a(b)$), or c (denoted by $\boxplus a(c)$). A MOMDP is a Mixed Observation Markov Decision Process, in which some states are visible to the agent and some are not. By converting the APS to a MOMDP, an external MOMDP solver can find a policy, which directly translates to an optimal sequence of moves. Like Black et al. (2017)'s approach the problem is therefore converted into a form for which there are feasible ways to solve it.

User modelling can also be done via predictions. In particular, Hadjinikolis et al. (2013) look at learning association rules in the beliefs of users, i.e. if he knows A then he should also know with a certain probability B too. Similar

[3]Meaning that he will not always take the most optimal action.

to Black et al. (2017) the aim is to persuade the user in a persuasion dialogue. However, whereas Black et al. solved the problem by reducing it to a planning problem, Hadjinikolis et al. tried to learn association rules, and let the agent act thereupon. The association rules are modelled via a probabilistic model over the possible beliefs of the user (modelling, that the user believes A with x chance, given that he knows B and C). Calculating those probabilities is, however, of exponential complexity. Hadjinikolis et al. therefore used a Monte-Carlo simulation to sample the probabilities. It was found that with relatively few samples a low error rate was achievable. Whether the learned probabilities were effective in a persuasion dialogue, is not mentioned.

A related approach is taken by Rienstra et al. (2013), in which the user model is included recursively in the agent's own belief state. They used abstract argumentation as knowledge representation. The goal of the agent is to persuade the opponent about some goal argument, i.e. the agent's goal is to make the goal argument acceptable under the grounded semantics. The dialogue is therefore a persuasion dialogue, with the goal being for the agent to win. Rienstra et al. put the user model in the belief state of the agent in the following way: (B, u, E) where B is the set of arguments the agent is aware of, u the utility function and $E = (B', u', E')$ as the belief state of the user with $B' \subseteq B$. Note that theoretically the depth of this belief state can be infinite, even though the depth will usually be fixed or limited. Rienstra et al. give two more of such belief states, for example by extending the recursive belief state by allowing E to be a set of belief states with probabilities. And secondly, by extending that to support 'virtual arguments', arguments that the agent itself is not aware of, but which are assumed to exist. Rienstra et al.'s goal was to find out whether increased complexity in the user model equates to better performance. They showed that by adding uncertainty over opponent models, the performance of the agent can indeed be increased. Note that, unlike Hadoux et al. (2015)'s approach, the user model itself is not stochastic. What is stochastic is the chance of a *possible* user model being true. Within such a user model, the user still acts deterministic.

4.4 Relevancy

While not a strategy, Prakken (2005) did describe relevancy as a potentially important property of strategies, and dialogues. Prakken (2005) studied coherence and flexibility in dialogues, particularly persuasion dialogues. The goal was to find out how different notions of coherence and flexibility affect a dialogue. In particular, Prakken defined the notion of relevancy, the idea that arguments are only desirable if they contribute towards the goal of the dialogue.

Prakken defined two notions of relevance, weak and strong relevance. Strong relevance is defined such that "*An attacking move in a dialogue d is [strongly] relevant iff it changes the dialogical status of d's initial move*" (Prakken, 2005, p. 20). This means that an argument is strongly relevant when it changes the status of the root argument. Prakken notes, however, that strong relevance requires a dialogue system in which agents can immediately reply to each other (i.e. using an immediate reply turn taking rule). This may be a drawback in some situations, which is why he also defined weak relevancy.

Weak relevancy states that "*An attacking move in a dialogue d is* weakly

relevant *iff it creates a new winning part of d for the speaker or removes a winning part of the hearer.*" (Prakken, 2005, p. 21). There, the winning part of a dialogue is defined as follows (adopted from Prakken (2005, Definition 5.8)).

Definition 4.4.1. Let d be a dialogue and p a player. A winning part d^p of d is recursively defined as follows.

1. First include the root argument;
2. for each move m of p that is included, include all its attackers;
3. for each attacking move m of $\hat{p}$ that is included, include one attacker of m that is *in* in d.

Prakken also notes that those notions of relevance are not complete. They only look at the relevance of a move in a dialectical tree, not at its contents. As such, a nonsense move like "I like birds, therefore this report of a crime is valid." may still found to be relevant.

4.5 Conclusion

There are many different ways to define a strategy. One could simply look at the currently legal moves in a dialogue, or also keep track of additional state. Likewise, using machine learning or Q-learning an agent may also be able to learn an optimal strategy himself.

Most of the listed approaches use extra information to allow the agent to make better decisions. This thesis will, however, first attempt to improve upon strategies that do not require extra information. As such, the exhaustive strategy from Black and Hunter (2009), the only strategy *not* using extra information, is used as a baseline. The other approaches may be used as inspiration if this thesis's conclusion is that improving strategies without giving agents extra information is hard.

Furthermore, the idea of relevancy as defined by Prakken (2005) may also be useful, since it does not require the use of extra information. It also fits in the idea of the intake, wherein the goal is to have a dialogue that is as short and complete as possible.

Chapter 5

An inquiry dialogue system

This chapter will describe two dialogue systems that can be used to generate inquiry dialogues. The dialogue systems used are an adoption of Black and Hunter's dialogue systems (Black & Hunter, 2009). The adoption consists of some changes in notation. It will be noted whenever changes are made when compared to Black and Hunter.

Black and Hunter (2009)'s dialogue systems can be used to generate argument and warrant inquiry dialogues. In an argument inquiry dialogue, an argument for the topic (of that sub-dialogue) is sought. A warrant inquiry dialogue is used to first find an argument for the topic, and then check whether that argument is acceptable.

Of those dialogue systems, warrant inquiry dialogues share the most similarities with an intake. In an intake, the police officer and the complainant after all first try to find out whether a crime has taken place, and then whether an acceptable argument exist. Argument inquiry dialogues are used to find new arguments.

Therefore, Black and Hunter's dialogue systems provide a way to jointly discover new arguments, and a way to check the acceptability of one. They thus seem to be a good model of intakes. However, whether they are actually a proper model of an intake, is discussed in the discussion (Chapter 10). This chapter instead focusses on the inner workings of the dialogue system.

First is discussed what elements make up the dialogue systems, and in what way they are used. Next, the dialogue systems are defined in detail. The next chapter will explain the corresponding strategy as defined by Black and Hunter.

5.1 Order of operations

The dialogue systems use turn taking rules, a protocol and a commitment function. The turn taking rules determine which agent has got the turn, the protocol determines the legal moves, and the commitment function defines what effect moves have. The protocol contains rules on when moves are valid. Moves may be valid only at the beginning of the dialogue, or only under some other circumstances.

The commitment function is defined for the commitment stores of the agents, and the query store. The commitment stores, of which each agent has got one,

track the propositions agents are committed to. The query store contains the current topic of the dialogue.

The validity of a move depends on the history of the dialogue. It is therefore useful to explicitly define the order in which the elements of the dialogue system are used. The following order is assumed.

1. The turn taking rules are used to determine which agent has got the turn;
2. The protocol gives for that agent a set of legal moves;
3. The agent chooses, using its strategy, a move from the set of legal moves;
4. That move is added to the dialogue history;
5. The effect rules on commitment and the query store are used to update the commitment stores and query store;
6. It is checked whether the dialogue should terminate;
7. If the dialogue did not terminate, go to 1.

Note that during step 2 the last added move to the history is the one uttered by the opponent.

5.2 Moves

Before defining any of the rules needed for a dialogue system, first the available moves are defined. Only at the end of the chapter will be defined when these moves are legal.

The available moves are determined by the communication language. It defines not what agents are able to say, but rather in what context they are able to say things during the dialogue.

Definition 5.2.1. The *communication language* L_c for argument and warrant inquiry dialogues is a set of locutions. This set consists of $\{open, close, assert\}$, which respectively open a sub-dialogue, (attempt to) close the current dialogue, and assert an argument. *open* is of the form $open(\theta, \gamma)$, where $\theta \in \{wi, ai\}$ denotes the type of sub-dialogue that is being opened, and $\gamma \in L_t$ is a defeasible rule or fact. *close* is of the form $close(\theta, \gamma)$, with $\theta \in \{wi, ai\}$ and $\gamma \in L_t$. *assert* is of the form $assert(\langle \Phi, \phi \rangle)$, where $\langle \Phi, \phi \rangle$ is an argument.

In argument and warrant inquiry dialogues, (sub-)dialogues can be opened, arguments can be asserted, and the current dialogue can be closed. The notation is slightly different from Black and Hunter (2009), in that the locutions have parameters.

Locutions are used in moves, which are of the following form.

Definition 5.2.2. *Moves* are of the form $\{P, O\} \times \{ai, wi\} \times L_c$. A move is denoted by $\langle x, \theta, p(c) \rangle$, where $x \in \{P, O\}$ denotes the player that uttered the move, θ is the current dialogue type, and $p(c)$ is a locution.

Moves are of a different form than in Black and Hunter (2009), to explicitly show the dialogue type of the sub-dialogue the move was uttered in. Black and Hunter defined a move as being of the form $\langle x, p, c \rangle$, where x is the player that uttered the move, p a performative and c the 'parameters' of the locution $p(c)$.

5.3 Dialogues

As explained in Chapter 3 dialogues are seen as a history of moves, i.e. as a list of moves. The following is the notation used to denote this sequence.

Definition 5.3.1. A *dialogue* d of length n is a sequence of moves $[m_1, \ldots, m_n]$, such that:

- The i^{th} move is given by $d[i]$, and $d[1]$ is the first move of the sequence;
- d_t^r, such that $1 \leq r < t \leq n$, denotes the sequence $[m_r, \ldots, m_t]$;
- If, given d_t^r, r equals 1, then $d_t = d_t^r$;
- If d is assumed to be n moves long, then $d = d_n$.

For example, let the dialogue d be n moves long, and consist of the sequence $[m_1, m_2, \ldots, m_{n-1}, m_n]$. d_{n-1} then denotes dialogue d without the last move, e.g. the sequence $[m_1, m_2, \ldots, m_{n-1}]$. d_{n-1}^2 denotes dialogue d minus the first and the last move, e.g. the sequence $[m_2, \ldots, m_{n-1}]$.

Note that the above notation is the exact opposite from Black and Hunter (2009). Black and Hunter use, for example, D_r^t to denote the part of dialogue D which runs from the r^{th} to the t^{th} move. Here, the notation is reversed, such that a normal dialogue (of length n) is denoted by d_n rather than by the potentially confusing d^n.

As sub-dialogues will be used, some terminology for the relationship between dialogues is needed.

Definition 5.3.2. Let d_t^r and $d_{t_1}^{r_1}$ be two dialogues. $d_{t_1}^{r_1}$ is a *sub-dialogue* of d_t^r iff $d_{t_1}^{r_1}$ is a sub-sequence of d_t^r ($r < r_1 \leq t_1 \leq t$). d_t^r is a *top-level* dialogue if $r = 1$; the set of all top-level dialogues is denoted $\mathcal{D}_{top}$. d_t^1 is a *top-dialogue* of d_t^r iff either the sequence d_t^1 is the same as the sequence d_t^r or d_t^r is a sub-dialogue of d_t^1. If d_t^r is a sequence of n moves, $d_{t_2}^r$ *extends* d_t^r iff the first n moves of $d_{t_2}^r$ are the sequence d_t^r.

A sub-dialogue is therefore a dialogue which is contained in a larger dialogue. Moreover, a top(-level) dialogue is a dialogue which is no sub-dialogue. Further, $\mathcal{D}_{top}$ denotes the set of all top dialogues.

Table 5.3.1 shows an example dialogue. There, the top dialogue is the whole dialogue, denoted by d_n or d_6^1. The dialogue contains one sub-dialogue. As will be explained in Section 5.6, a (sub-)dialogue terminates when both agents after another want to close the dialogue. Therefore, the sub-dialogue of Table 5.3.1 is d_4^2.

For notation purposes $M^{<\infty}$ will be used to denote the set of all dialogues with a finite history, and M will denote a single move.

The *topic* of a dialogue is defeasible rule or defeasible fact (from L_t). The goal of a dialogue is to do something with that topic. For example, in warrant inquiry dialogues is the topic a defeasible fact, for which an acceptable argument is sought. The topic in Black and Hunter (2009)'s argument inquiry dialogues is a defeasible rule, for which an argument is jointly constructed.

Lastly, the effect of the moves on the dialogue need to be defined.

1	$\langle P, \theta, open(\theta, \gamma)\rangle$
2	$\quad\langle O, \theta', open(\theta', \gamma')\rangle$
3	$\quad\langle P, \theta', close(\theta', \gamma')\rangle$
4	$\quad\langle O, \theta', close(\theta', \gamma')\rangle$
5	$\langle P, \theta, close(\theta, \gamma)\rangle$
6	$\langle O, \theta, close(\theta, \gamma)\rangle$

Table 5.3.1: An example of a dialogue with one sub-dialogue. Here d_4^2 is a sub-dialogue of the whole dialogue, which is denoted by d_6^1 or d_n. θ and θ' denote the dialogue type, and γ and γ' topics.

5.4 Effect rules and query store

As moves are uttered and added to the dialogue, they also have side effects. Two kinds of side effects are used here. First is the usual effect of moves on the commitment of agents, as defined by the commitment function. Note that no subscripts are used to denote the commitment stores and query stores. This is in contrast with Black and Hunter, to minimise the number of subscripts used.

Definition 5.4.1. A *commitment function* is of the type $C : M^{<\infty} \times \{P, O\} \to \mathcal{P}(L_t)$, such that $C(\emptyset, p) = \emptyset$. $C(d, p)$ denotes the commitment store of player p in dialogue d.

Definition 5.4.2. The commitment function for argument and warrant inquiry dialogues is the following. Let d_n be the current dialogue, γ a topic, and $\theta \in \{ai, wi\}$:

$$C(d_n, x) = \begin{cases} C(d_{n-1}, x) \cup \Phi & d_n[n] = \langle x, \theta, assert(\langle \Phi, \phi\rangle)\rangle \\ C(d_{n-1}, x) & \text{otherwise} \\ \emptyset & d = \emptyset \end{cases}$$

Note that the ϕ is not added to the commitment store. It is already part of the query store, which is explained below. Furthermore, the commitment store grows monotonically over time, as no elements are ever removed.

The second kind of side effect is the one on the query store. The query store is used in argument and warrant inquiry dialogues to contain the literals of the current topic. Both agents try to find (acceptable) arguments for the literals that are a part of the query store.

Definition 5.4.3. Let d_n be the dialogue so far, $x \in \{P, O\}$ and $\theta, \theta' \in \{ai, wi\}$. The *query store*, denoted $QS : M^{<\infty} \to \mathcal{P}(L_t)$, is a finite set of literals, such that:

$$QS(d_n) = \begin{cases} \{\beta_1, \ldots, \beta_n, \alpha\} & \text{if } d_n[n] = \langle x, \theta, open(\theta', \beta_1 \wedge \cdots \wedge \beta_n \Rightarrow \alpha)\rangle \\ \emptyset & \text{otherwise} \end{cases}$$

The query store is set at the beginning of a (sub-)dialogue.

5.5 Turn taking

Now that is clear what can be said, and what the effects of the moves are, it needs to be defined when an agent may say something.

In general, strict turn taking is assumed. An agent has always got a chance with strict turn taking to respond to its opponent Thus, with d_n being the current dialogue of n moves long:

Definition 5.5.1. The *turn taking rule*, $T : M^{\leq\infty} \to \{P, O\}$, for argument and warrant inquiry dialogues is the following. Let $\gamma \in \mathcal{P}(L_t)$, $\theta \in \{ai, wi\}$, and $p(c)$ be a locution (where p is a performative and c any existing parameter):

$$T(d_n) = \begin{cases} P & \text{if } d_n[n] = \langle O, \theta, p(c)\rangle \\ O & \text{if } d_n[n] = \langle P, \theta, p(c)\rangle \end{cases} \tag{5.5.1}$$

5.6 Termination

But, when does the dialogue come to an end? Argument and warrant inquiry dialogues terminate when both agents want to close the current dialogue. This is realised via a notion of matched-close, when a dialogue extends another one, and a notion of sub-dialogues.

Definition 5.6.1. Let d_t^r be a dialogue of type θ, with topic γ and $x, \hat{x} \in \{P, O\}$. The move $d_t^r[s]$, such that $r < s \leq t$, is a *matched-close* iff $d_t^r[s-1] = \langle x, \theta, close(\theta, \gamma)\rangle$ and $d_t^r[s] = \langle \hat{x}, \theta, close(\theta, \gamma)\rangle$

Matched-close requires both agents to agree on the closure of the current dialogues. This way, if only one side wants to close the dialogue, then the other side can still make a move.

A (sub-)dialogue terminates if the following holds:

Definition 5.6.2. Let d_n be a (sub-)dialogue. d_n *terminates* at n iff the following conditions hold:

1. $d_n[n]$ is a matched-close for d_n;
2. $\nexists d_m$ such that d_m terminates at m and d_n extends d_m;
3. $\forall d_m^r$ $(1 \leq r \leq m \leq n)$ if d_m^r is a sub-dialogue of d_n,
   then $\exists d_{m_1}^r$ such that $d_{m_1}^r$ terminates at m_1;
   and either $d_{m_1}^r$ extends d_m^r or d_m^r extends $d_{m_1}^r$;
   and $d_{m_1}^r$ is a sub-dialogue of d_n.

A sub-dialogue has to be matched-close in order for it to terminate. Also, the current (sub-)dialogue must not already be terminated and all sub-dialogues must already be terminated.

5.7 Dealing with sub-dialogues

Sub-dialogues make it hard to say things about the current state of a dialogue. After all, if a sub-dialogue just terminated, then the last uttered move is no longer relevant for the current dialogue. The relevant move is in fact the one *before* the move that opened the last closed sub-dialogue.

It is therefore useful to have a way to refer to the current (sub-)dialogue.

Definition 5.7.1. Let d_n be a dialogue. The *current (sub-)dialogue* is given by $cDiag(d_n)$ such that $cDiag(d_n) = d_n^{r_1}$ $(1 \leq r_1 \leq n)$ where the following conditions hold:

1. $d_n[r_1] = \langle x, \theta, open(\theta, \gamma)\rangle$ for some $x \in \{P, O\}$, some $\gamma \in L_t$ and some $\theta \in \{wi, ai\}$;
2. $\forall d_{n_1}^{r_2}$ if $d_{n_1}^{r_2}$ is a sub-dialogue of $d_n^{r_1}$;
   then $\exists d_{n_2}^{r_2}$ s.t. either $d_{n_2}^{r_2}$ extends $d_{n_1}^{r_2}$ or $d_{n_1}^{r_2}$ extends $d_{n_2}^{r_2}$;
   and $d_{n_2}^{r_2}$ is a sub-dialogue of $d_n^{r_1}$;
   and $d_{n_2}^{r_2}$ terminates at n_2;
3. $\nexists d_{n_3}^{r_1}$ s.t. $d_n^{r_1}$ extends $d_{n_3}^{r_1}$ and $d_{n_3}^{r_1}$ terminates at n_3.

The current (sub-)dialogue is therefore a subsequence of the entire dialogue, such that;

1. The first move of the sequence opens that dialogue;
2. For all sub-dialogues of the sequence, it holds that they are terminated;
3. The (sub-)dialogue itself has not yet terminated.

Given the current dialogue, the current topic and current query store can easily be returned.

Definition 5.7.2. Given a dialogue d_n, $cTopic$ returns *the topic of the current dialogue*. If $d_n^r = cDiag(d_n)$ with $d_n^r[r] = \langle x, \theta, open(\theta, \gamma)\rangle$, then $cTopic(d_n) = \gamma$.

Definition 5.7.3. Let d_n be a dialogue, and $d_n^r = cDiag(d_n)$ the current dialogue. The query store of the current dialogue, denoted $cQS(d_n)$, is then given by $QS(d_n^r)$.

5.8 Knowledge base

The following section describes the propositions the agent has access to during the dialogue. Besides his own and his opponent's commitment store, he namely also has got access to his private space, or knowledge base.

Definition 5.8.1. The *knowledge base* of agent x, denoted Σ_x, is a set of defeasible rules and facts ($\Sigma_x \subseteq L_t$). The knowledge base of agent x is set before the start of the dialogue.

Definition 5.8.2. Let d_n be the dialogue so far, and player x have the turn ($T(d_n) = x$). Agent x then has access to the following *knowledge*, denoted by $K : M^{<\infty} \times \{P, O\} \to \mathcal{P}(L_t)$, such that $K(d_n, x) = \Sigma_x \cup C(d_n, x) \cup C(d_n, \hat{x})$.

An agent therefore reasons over his own knowledge base, and any commitment stores.[1]

5.9 Argument inquiry

Next, the protocol for argument inquiry dialogues is defined. In argument inquiry dialogues, agents try to find an argument for a given defeasible rule. An agent can always try and close the current dialogue. However, he can only open sub-dialogues or assert arguments if they relate to the current query store, and if the move was not uttered before.

A protocol is a function which, given the current player and the dialogue so far, returns a list of legal moves.

Definition 5.9.1. The *argument inquiry protocol* is a function $Pr_{ai} : M^{<\infty} \times \{P, O\} \to \mathcal{P}(M)$. Let d_n be a dialogue of n moves, or $\emptyset$ at the beginning of the dialogue, such that $d_n[n]$ was uttered by $\hat{x}$, and $cTopic(d_n) = \gamma$. Then $Pr_{ai}(d_n, x)$ is

$$Pr_{ai}^{assert}(d_n, x) \cup Pr_{ai}^{open}(d_n, x) \cup \{\langle x, ai, close(ai, \gamma)\rangle\}$$

where

$$\begin{aligned} Pr_{ai}^{assert}(d_n, x) = \{\langle x, ai, assert(\langle \Phi, \phi\rangle)\rangle | \\ \Phi \subseteq K(d_n, x), \\ \phi \in cQS(d_n), \\ \Phi \mathrel{|\!\sim} \phi, \\ \forall x' \in \{P, O\} : \langle x', ai, assert(\langle \Phi, \phi\rangle)\rangle \notin d_n\} \end{aligned}$$

and

$$\begin{aligned} Pr_{ai}^{open}(d_n, x) = \{\langle x, ai, open(ai, \beta_1 \wedge \cdots \wedge \beta_n \Rightarrow \alpha)\rangle | \\ \alpha \in cQS(d_n) \text{ or } cQS(d_n) = \emptyset, \\ (\beta_1 \wedge \cdots \wedge \beta_n \Rightarrow \alpha) \in K(d_n, x), \\ \forall x' \in \{P, O\} : \langle x', ai, open(ai, \beta_1 \wedge \cdots \wedge \beta_n \Rightarrow \alpha)\rangle \notin d_n\} \end{aligned}$$

An agent can therefore always try to close the current argument inquiry dialogue. He can assert any argument which has not been asserted before, and of which the support is a subset of his knowledge and the conclusion a member is of the current query store. The agent can open a sub-dialogue using any defeasible rule of which the conclusion is in the current query store, and which has not been uttered before.

Table 5.9.1 shows an example of an argument inquiry dialogue. The proponent starts with a knowledge base of $\{(c \Rightarrow d, 1), (b \Rightarrow c, 1), (a \Rightarrow b, 1)\}$, and the

[1] The above definition assumes a dialogue between two agents, with two commitment stores.

opponent with $\{(a,1),(b,1)\}$. Meaning that all rules and facts have a preference level of one. The dialogue starts with the proponent wanting to find an argument for the defeasible rule $c \Rightarrow d$. To this end, two sub-dialogues are opened, one for $b \Rightarrow c$, and the other one for $a \Rightarrow b$.

First, the proponent opens an argument inquiry dialogue to find an argument for $c \Rightarrow d$. The opponent reacts by saying that he does not know what to do, and that he wants to close the dialogue. Instead, the proponent opens a sub-dialogue for $b \Rightarrow c$. After all, an argument for $b \Rightarrow c$ can be used in an argument for $c \Rightarrow d$. The opponent reacts by asserting a, after which the proponent knows enough to assert the argument $\langle\{a, a \Rightarrow b\}, b\rangle$. Both parties then have nothing to say on the subject of $a \Rightarrow b$, so they close the dialogue. Then, in the sub-dialogue with topic $b \Rightarrow c$, is the opponent able to construct and assert an argument for c. Afterwards, the sub-dialogue is closed. Next, the proponent and opponent assert in order two different arguments for d using $c \Rightarrow d$. Since after those moves neither have anything to assert, or any rule to open a sub-dialogue over, the dialogue is closed.

	$\mathbf{CS_P}$	**History**	$\mathbf{CS_O}$	**QS**
1		$\langle P, ai, open(ai, c \Rightarrow d)\rangle$		$\{c, d\}$
2		$\langle O, ai, close(ai, c \Rightarrow d)\rangle$		
3		$\langle P, ai, open(ai, b \Rightarrow c)\rangle$		$\{b, c\}$
4		$\langle O, ai, assert(\langle\{b\}, b\rangle)\rangle$	b	
5	b,$b \Rightarrow c$	$\langle P, ai, assert(\langle\{b, b \Rightarrow c\}, c\rangle)\rangle$		
6		$\langle O, ai, close(ai, b \Rightarrow c)\rangle$		
7		$\langle P, ai, open(ai, a \Rightarrow b)\rangle$		$\{a, b\}$
8		$\langle O, ai, assert(\langle\{a\}, a\rangle)\rangle$	a	
9	a,$a \Rightarrow b$	$\langle P, ai, assert(\langle\{a, a \Rightarrow b\}, b\rangle)\rangle$		
10		$\langle O, ai, close(ai, a \Rightarrow b)\rangle$		
11		$\langle P, ai, close(ai, a \Rightarrow b)\rangle$		
12		$\langle O, ai, assert(\langle\{a, a \Rightarrow b, b \Rightarrow c\}, c\rangle)\rangle$	$a \Rightarrow b$,$b \Rightarrow c$	
13		$\langle P, ai, close(ai, b \Rightarrow c)\rangle$		
14		$\langle O, ai, close(ai, b \Rightarrow c)\rangle$		
15	$c \Rightarrow d$	$\langle P, ai, assert(\langle\{b, b \Rightarrow c, c \Rightarrow d\}, d\rangle)\rangle$		
16		$\langle O, ai, assert(\langle\{a, a \Rightarrow b, b \Rightarrow c, c \Rightarrow d\}, d\rangle)\rangle$	$c \Rightarrow d$	
17		$\langle P, ai, close(ai, c \Rightarrow d)\rangle$		
18		$\langle O, ai, close(ai, c \Rightarrow d)\rangle$		

Table 5.9.1: An example argument inquiry dialogue, from Black and Hunter (2009), with $\Sigma_P = \{(c \Rightarrow d, 1), (b \Rightarrow c, 1), (a \Rightarrow b, 1)\}$ and $\Sigma_O = \{(a,1),(b,1)\}$. As the commitment stores grow monotonically, only new entries are denoted.

5.10 Warrant inquiry

In warrant inquiry dialogues, an acceptable argument is sought for a defeasible fact. An argument is acceptable, given a set of uttered propositions if the

argument is *in* in the corresponding dialectical tree.[2]

Next, some definitions used in the protocol for warrant inquiry dialogues are defined. $defderiv$ returns all literals that can be derived from a set of propositions. $RootArg$ returns the first argument that was asserted in the current dialogue, if it exists.

Definition 5.10.1. The function $defderiv : \mathcal{P}(L_t) \to \mathcal{P}(L_t)$ returns, given a set of rules and literals $\Psi \subseteq L_t$, all literals that can be defeasible derived from it:

$$defderiv(\Psi) = \{\phi | \Phi \subseteq \Psi, \Phi \mathrel{|\!\sim} \phi\}$$

Definition 5.10.2. The function $RootArg : M^{<\infty} \to \langle \Phi, \phi \rangle$ returns the *root argument* of a warrant inquiry dialogue. Let d_t^r be a warrant dialogue, with players $\{P, O\}$:

$$RootArg(d_t^r) = \begin{cases} \langle \Phi, \phi \rangle & \text{if } \exists s \text{ s.t. } r < s \leq t, \\ & d_t^r[s] = \langle x, \theta, assert(\langle \Phi, \phi \rangle) \rangle, \\ & Topic(d_t^r) = \gamma, \\ & x \in \{P, O\}, \\ & \nexists s' \text{ such that } r < s' < s \text{ and} \\ & \quad \exists \langle \Phi', \phi' \rangle \text{ s.t. } d_t^r[s'] = \langle x', \theta, assert(\langle \Phi', \phi' \rangle) \rangle \\ & \quad \text{and } x' \in \{P, O\} \\ \emptyset & \text{otherwise} \end{cases}$$

The protocol for a warrant inquiry dialogue is given below. Like with argument inquiry dialogues can warrant inquiry dialogues always be closed. Moreover, an agent can assert arguments which have not been asserted before, if no other argument has been asserted before or if asserting the argument would change the status of the root argument in the dialectical tree. Furthermore, an agent can open an argument inquiry dialogue, if there is no root argument, or if the defeasible rule's negated conclusion can be derived from the commitment stores.

Definition 5.10.3. If d_n is the dialogue, with $x \in \{P, O\}$ and $topic(d_n) = \gamma$, then the *warrant inquiry protocol* Pr_{wi} is

$$Pr_{wi}^{assert} \cup Pr_{wi}^{open} \cup \langle x, wi, close(wi, \gamma) \rangle$$

where

$$\begin{aligned} Pr_{wi}^{assert} = \{ & \langle x, wi, assert(\langle \Phi, \phi \rangle) \rangle | \\ & (1)\ RootArg(d_n) = \emptyset \text{ and } \phi = \gamma \text{ or} \\ & \quad T\left(RootArg\left(d_n\right), K(d_n, x) \cup \Phi\right) \neq T(RootArg(d_n), K(d_n, x)) \\ & (2)\ \nexists t' \text{ s.t. } 1 < t' \leq n \text{ and} \\ & \quad d_n[t'] = \langle x', wi, assert(\langle \Phi, \phi \rangle) \rangle \text{ and } x' \in \{P, O\}\} \end{aligned}$$

[2] i.e. for an argument A and set of propositions Ψ, $status(A, \Psi) = in$.

$$Pr_{wi}^{open} = \{\langle x, wi, open(ai, \beta_1 \wedge \cdots \wedge \beta_n \Rightarrow \alpha)\rangle |$$
$$(1)\ RootArg(d_n) = \emptyset \text{ and } \alpha = \gamma \text{ or}$$
$$\neg\alpha \in defderiv\left(C(d_n, x) \cup C(d_n, \hat{x})\right), \text{ and}$$
$$(2)\ \nexists t' \text{ s.t. } 1 < t \leq n \text{ and}$$
$$d_n[t'] = \langle x', wi, open(ai, \beta_1 \wedge \cdots \wedge \beta_n \Rightarrow \alpha)\rangle \text{ and } x' \in \{P, O\}\}$$

In warrant inquiry dialogues one can always try to close the current dialogue. Asserting arguments can only be done if there is no root argument, or asserting the argument would change the dialectical tree. Furthermore, only argument inquiry sub-dialogues can be opened, and only if there is a root argument and the consequent is the same as the current topic, or if the defeasible rule's consequent is negated.

Table 5.10.1 shows an example of an warrant inquiry dialogue, adapted from the example given by Black and Hunter (2009). In the example, an acceptable argument is sought for the proposition b. Furthermore, four argument inquiry sub-dialogues are opened and closed. The commitment stores are depicted as containing beliefs instead of propositions, since arguments are compared on their preference levels.

5.11 Conclusion

The dialogue systems of Black and Hunter serve two purposes. Argument inquiry dialogues are used to find arguments and warrant inquiry dialogues check whether arguments are acceptable.

The dialogue systems do not specify, however, how the dialogues should be generated. Moreover, it needs to be specified whether the generated dialogues should adhere to some sort of criteria. The next chapter will define such a strategy.

	$\mathbf{CS_P}$	**History**	$\mathbf{CS_O}$	**QS**
1		$\langle P, wi, open(wi, b)\rangle$		b
2		$\langle O, wi, close(wi, b)\rangle$		
3	$(a, 4)$, $(a \Rightarrow b, 4)$	$\langle P, wi, assert(\langle\{(a, 4), (a \Rightarrow b, 4)\}, b\rangle)\rangle$		
4		$\langle O, wi, assert(\langle\{(d, 3), (d \Rightarrow \neg a, 3)\}, \neg a\rangle)\rangle$	$(d, 3)$, $(d \Rightarrow \neg a, 3)$	
5	$(c, 3)$, $(c \Rightarrow \neg b, 3)$	$\langle P, wi, assert(\langle\{(c, 3), (c \Rightarrow \neg b, 3)\}, \neg b\rangle)\rangle$		
6		$\langle O, wi, assert(\langle\{(\neg d, 1)\}, \neg d\rangle)\rangle$	$(\neg d, 1)$	
7		$\langle P, wi, open(ai, a \Rightarrow b)\rangle$		a, b
8		$\langle O, ai, close(ai, a \Rightarrow b)\rangle$		
9		$\langle P, ai, close(ai, a \Rightarrow b)\rangle$		
10		$\langle O, wi, open(ai, d \Rightarrow \neg a)\rangle$		$d, \neg a$
11		$\langle P, ai, close(ai, d \Rightarrow \neg a)\rangle$		
12		$\langle O, ai, close(ai, d \Rightarrow \neg a)\rangle$		
13		$\langle P, wi, open(ai, c \Rightarrow \neg b)\rangle$		$c, \neg b$
14		$\langle O, ai, close(ai, c \Rightarrow \neg b)\rangle$		
15		$\langle P, ai, close(ai, c \Rightarrow \neg b)\rangle$		
16		$\langle O, wi, open(ai, e \Rightarrow \neg d)\rangle$		$e, \neg d$
17	$(e, 2)$	$\langle P, ai, assert(\langle\{(e, 2)\}, e\rangle)\rangle$		
18		$\langle O, ai, assert(\langle\{(e, 2), (e \Rightarrow \neg d, 2)\}, \neg d\rangle)\rangle$	$(e, 2)$, $(e \Rightarrow \neg d, 2)$	
19		$\langle P, ai, close(ai, e \Rightarrow \neg d)\rangle$		
20		$\langle O, ai, close(ai, e \Rightarrow \neg d)\rangle$		
21		$\langle P, wi, close(wi, b)\rangle$		
22		$\langle O, wi, assert(\langle\{(\neg e, 1)\}, \neg e\rangle)\rangle$	$(\neg e, 1)$	
23		$\langle P, wi, close(wi, b)\rangle$		
24		$\langle P, wi, close(wi, b)\rangle$		

Table 5.10.1: An example warrant inquiry dialogue, from Black and Hunter (2009). $\Sigma_P = \{(a, 4), (a \Rightarrow b, 4), (c, 3), (c \Rightarrow \neg b, 3), (e, 2)\}$ and $\Sigma_O = \{(d, 3), (d \Rightarrow \neg a, 3), (\neg d, 1), (e \Rightarrow \neg d, 2), (\neg e, 1)\}$. As the commitment stores grow monotonically, only new entries are denoted. The commitment stores are depicted as containing beliefs instead of propositions.

Chapter 6

Soundness, completeness and the exhaustive strategy

The previous chapter did not specify how dialogues are generated from the argument and warrant inquiry dialogue systems. Before explaining how Black and Hunter (2009) generated dialogues, however, first will be explained what set of goals the generated dialogue should adhere to.

As explained below, Black and Hunter's inspiration is the safety-critical medical domain. In short, their goal is therefore to generate dialogues that never give a false answer, and that will always give an answer if one exists. These properties are, respectively, called soundness and completeness.

Using those goals, Black and Hunter devised a strategy that generates sound and complete dialogues. This strategy is called the exhaustive strategy.

In the following, firstly soundness and completeness is formally defined for both argument and warrant inquiry dialogues. Secondly, the exhaustive strategy is defined.

6.1 Soundness and completeness

Black and Hunter (2009)'s goal is to define a dialogue system that models the medical domain. They argue that, since the medical domain is safety-critical, completeness and soundness are important safety properties of any medical dialogue. After all, when dealing with a diagnosis, for example, you expect the doctor to have explored all options (completeness) and to be given a correct diagnosis (soundness). Black and Hunter's aim is therefore to define a strategy that generates sound and complete dialogues. Before defining a strategy, however, first needs to be clear what exactly Black and Hunter mean with completeness and soundness. As they define each notion separately for the argument and warrant inquiry, that distinction is followed here as well.

Note that dialogues are assumed to be 'well-formed'. A well-formed dialogue is a dialogue of which all moves are legal according to its dialogue system.

For both dialogue systems, first the notion of outcome is defined, then the notion of soundness, and lastly completeness.

6.1.1 Argument inquiry

The outcome of an argument inquiry dialogue is defined as a set of all arguments derivable from the union of the commitment stores whose conclusion is in the query store.

Definition 6.1.1. Let d_t^r be an argument inquiry dialogue with the participants P and O. The function $outcome_{ai} : \mathcal{D} \rightarrow \mathcal{P}\left(AT_{L_t}\right)$ returns the outcome of an argument inquiry dialogue, with

$$outcome_{ai} = \{\langle \Phi, \phi \rangle \in AT_{C(d_t^r, O) \cup C(d_t^r, P)} | \phi \in QS(d_t^r)\}$$

Therefore, the outcome is empty if in an argument inquiry dialogue no argument is found for any of the propositions in the query store. Otherwise, the outcome consists of all arguments constructible from the commitment stores that support a proposition from the query store.

Black and Hunter say that an argument inquiry dialogue is sound if and only if all arguments in the outcome of the dialogue are constructible from the union of the agents' beliefs.

Definition 6.1.2. Let d_t^r be an argument inquiry dialogue. d_t^r is *sound* iff $\forall \langle \Phi, \phi \rangle \in outcome_{ai}(d_t^r)$ it holds that $\langle \Phi, \phi \rangle \in AT_{\Sigma_O \cup \Sigma_P}$.

Similarly, an argument inquiry dialogue is complete if the dialogue is terminated and if all arguments constructible from the union of the agents' beliefs with a conclusion that is in the query store, are also in the outcome of the dialogue. In other words, a terminated dialogue is complete if all possible arguments for the propositions in the query store have been found.

Definition 6.1.3. Let d_t^r be an argument inquiry dialogue. d_t^r is *complete* if it is terminated at t and $\forall \langle \Phi, \phi \rangle \in \{\langle \Psi, \psi \rangle \in AT_{\Sigma_O \cup \Sigma_P} | \psi \in QS(d_t^r)\}$ it holds that $\langle \Phi, \phi \rangle \in outcome_{ai}(d_t^r)$.

6.1.2 Warrant inquiry

The outcome of a warrant inquiry dialogue is the root argument of the dialectical tree iff the root argument is undefeated (*out*), and otherwise empty.

Definition 6.1.4. The outcome of a warrant inquiry dialogue is given by the function $outcome_{wi} : \mathcal{D} \rightarrow AT_{L_t}$. Let d_t^r be a warrant inquiry dialogue, with the participants P and O, and $CS(d_t^r) = C(d_t^r, P) \cup C(d_t^r, O)$:

$$outcome_{wi}(d_r^t) = \begin{cases} RootArg(d_r^t) & status(RootArg(d_t^r), CS(d_t^r)) = in \\ \emptyset & RootArg(d_t^r) = \emptyset \\ \emptyset & \text{otherwise} \end{cases}$$

The soundness of a warrant inquiry dialogue hinges on the status in the dialectical tree. A dialogue is sound only if, given that the outcome is the argument $\langle \Phi, \phi \rangle$, and that $\langle \Phi, \phi \rangle$ is the root of the dialectical tree, then the status of $\langle \Phi, \phi \rangle$ in the tree is *in*.

Definition 6.1.5. Let d_t^r be a warrant inquiry dialogue. d_t^r is *sound* iff, if d_t^r terminates at t and $outcome_{wi}(d_t^r) = \langle \Phi, \phi \rangle$, then $status(\langle \Phi, \phi \rangle, \Sigma_O \cup \Sigma_P) = in$.

Like with argument inquiry dialogues, is the definition of completeness practically the reverse of the definition of soundness. A warrant inquiry dialogue is, therefore, complete if and only if there exists an argument in the union of the agents' beliefs that is *in* as the root in the dialectical tree, and if the outcome of the dialogue is that argument.

Definition 6.1.6. The warrant inquiry dialogue d_t^r is *complete* iff, if it terminates at t, $RootArg(d_t^r) = \langle \Phi, \phi \rangle$ and $status(\langle \Phi, \phi \rangle, \Sigma_O \cup \Sigma_P) = in$, then $outcome_{wi}(d_t^r) = \langle \Phi, \phi \rangle$.

6.2 Exhaustive strategy

As already mentioned, the goal of Black and Hunter (2009) is to define a dialogue system that models the medical domain. Therefore, their aim is to define a strategy that generates sound and complete dialogues. Here, a strategy is a function which, given a list of legal moves, selects a single one. This section details the proposed strategy of Black and Hunter, called the exhaustive strategy. To this end, functions are defined which return a legal *open* move and a legal *assert* move, if they exist.

Definition 6.2.1. Let $\Xi = \{\langle x, \theta, open(\theta_1, \gamma_1) \rangle, \ldots, \langle x, \theta, open(\theta_k, \gamma_k) \rangle\}$ be a set of legal *open* moves that agent x could make. The function $pick_o$ returns the selected *open* move to make. $pick_o(\Xi) = \langle x, \theta, open(\theta_j, \gamma_j) \rangle$ such that $1 \leq j \leq k$ and $\nexists j' : j' < j$.

Definition 6.2.2. Let $\Xi = \{\langle x, \theta, assert(\langle \Phi_1, \phi_1 \rangle) \rangle, \ldots, \langle x, \theta, assert(\langle \Phi_k, \phi_k \rangle) \rangle\}$ be a set of legal *assert* moves that agent x could make. The function $pick_a$ returns the selected *assert* move to make. $pick_a(\Xi) = \langle x, \theta, assert(\langle \Phi_j, \phi_j \rangle) \rangle$ such that $1 \leq j \leq k$ and $\nexists j' : j' < j$.

Definition 6.2.3. The *exhaustive strategy* for an agent x in a dialogue d_n is a function $S_{exh} : \mathcal{D}_{top} \to M$. Let $T(d_n) = x$, $ctopic(d_n) = \gamma$ and $\theta \in \{ai, wi\}$:

$$S_{exh}(d_n) = \begin{cases} pick_a(asserts(d_n, x)) & \text{if } asserts(d_n, x) \neq \emptyset \\ pick_o(opens(d_n, x)) & \text{if } asserts(d_n, x) = \emptyset \text{ and } opens(d_n, x) \neq \emptyset \\ \langle x, \theta, close(\theta, \gamma) \rangle & \text{if } asserts(d_n, x) = \emptyset \text{ and } opens(d_n, x) = \emptyset \end{cases}$$

where $asserts(d_n, x)$ returns all legal *assert* moves for agent x in dialogue d_n, and $opens(d_n, x)$ likewise returns all legal *open* moves.

The dialogues that are generated by the exhaustive strategy are called 'exhaustive dialogues'. A dialogue is 'generated' by the exhaustive strategy iff every move in the dialogue is the move that the exhaustive strategy would have picked on that moment.

Example 6.2.4. *The argument inquiry dialogue shown in Table 5.9.1 (on page 28) is also an exhaustive argument inquiry dialogue (i.e. one in which the exhaustive strategy was used). Likewise, the warrant inquiry dialogue shown in Table 5.10.1 (on page 31) is also an exhaustive warrant inquiry dialogue.*

The exhaustive strategy therefore returns a legal assert move if one exist, otherwise it returns a legal open move. If no legal open move exists, it returns the close move.

Note that dialogues generated by the exhaustive strategy have outcomes that can also be reached by reasoning over the union of the agents' beliefs. Black and Hunter note this as well: "*As the dialogue outcome we are aiming for is the same as the outcome we would arrive at if reasoning with the union of the agents' beliefs, a natural question to ask is why not simply pool the agents' beliefs and then reason with this set?*" (Black & Hunter, 2009). They give as answer that in many real world scenarios this may not desirable/legal due to privacy issues. In those cases it may be required to reason with separate knowledge bases.

In other cases, it may simply be impossible to construct a union of the knowledge bases, for example when dealing with the intake. Assuming that the police officer knows the rules of the law and the complainant the facts, it is not known beforehand by the police officer which facts are applicable. In other words, only through a dialogue with the complainant is the police officer able to ascertain the facts. Since the complainant has no intricate knowledge of the law, he is assumed to be unable to give all relevant facts himself. Therefore, it is reasonable to assume that for the intake it is impractical, if not impossible, to construct the union of the agents' beliefs.

6.3 Conclusion

Inspired by the medical domain, Black and Hunter specified that dialogues should be complete and sound. In other words, dialogues should, respectively, always produce an answer if one exists, and should only generate dialogues that give valid answers. Their exhaustive strategy serves this purpose. By asserting all known arguments, opening all possible sub-dialogues and closing the dialogue otherwise, sound and complete dialogues are generated.

Note that dialogues generated using this strategy are typically large. While this may be acceptable in the medical domain, it may not be for intakes. The next chapter will thus show that the exhaustive strategy is not always ideal, and define strategies that try to mitigate some of these problems.

Chapter 7

Modified exhaustive strategies

The strategy as defined by Black and Hunter is not ideal for intakes. For one, it generates long dialogues. This chapter will modify that strategy (the exhaustive strategy from Black and Hunter (2009)) in two ways. Both times, an extra condition is added for assert moves to be uttered. The exhaustive strategy is modified in two ways, since it possibly contains two kinds of redundancies. The modified strategies both aim to reduce one of these kinds of duplicity, as inspired by the following scenario.

A complainant arrives at a police station to report online fraud. The police officer handling the intake does not have the goal to make the intake as long as possible. Rather, he firstly tries to determine whether fraud actually took place. He uses his knowledge of the law to determine if it was fraud that took place, and stops the intake otherwise. If he determines that fraud took place, then he stops trying to ascertain that. Instead, he moves on to gather further details of the case. In other words, the police officer stops seeking an argument for a proposition if he already has got an argument for it, or, more importantly, no further argument can change the status of the proposition.

As mentioned, there are thus two kinds of redundancy observed. The first kind of redundant information is, therefore, the observation that during a dialogue multiple arguments for the same conclusion are found. The second kind of unnecessary information is that the exhaustive strategy also asserts arguments which do not change the status of the root argument. While only applicable to warrant inquiry dialogues, these arguments seem to serve no purpose.

In other words, it seems that dialogues generated by the exhaustive strategy can be made more relevant. Prakken (2005) also discussed relevancy, but for persuasion dialogues. As discussed in Section 4.4, he defined two notions of relevancy in order to get more coherent persuasion dialogues. Of those two notions, strong relevancy might help to make inquiry dialogues more relevant.

The strategies defined in Section 7.2.2 in fact enforce that *assert* moves either change the status of the root argument, or not modify the dialectical tree. This is an application of strong relevance, with the caveat that agents are allowed to 'build' arguments.

Note that the modifications proposed here may seem trivial, but are not.

This means that while the formal definitions might be trivial, it is not guaranteed that those definitions are effective in making inquiry dialogues more relevant.

7.1 Commitment store versus dialogue

The goal is to make the exhaustive strategy generate smaller dialogues and dialectical trees. However, maybe the protocol itself can also be made more coherent, by changing where the strategy looks to see which arguments already have been asserted. Therefore, the question needs to be answered, when does an argument for a specific proposition exist? Should the argument have been asserted in the dialogue? Or should the argument be constructible from the union of the commitment stores?

In general, strategies often reason upon the union of the commitment stores and the current agent's knowledge base. It may be interesting, however, to see what happens when a strategy instead reasons on a dialogue (and a knowledge base). What 'reasoning on the dialogue' exactly means, will depend on the kind of redundancy that this thesis tries to reduce.

When trying to reduce the number of arguments asserted in support of the same proposition, for example, you can only look at the asserted arguments, or look at the argumentation theory of the commitment stores. Therefore, only the strategy itself has to be changed. On the other hand, when trying to assert less irrelevant arguments, it may be useful to have a 'more relevant' dialectical tree, by constructing it from the dialogue. In that case, first the notion of a dialectical tree constructed on the dialogue is defined.

7.1.1 One argument per proposition

The following definitions closely follow the definition of $pick_a$ (Definition 6.2.2). The first $pick_a$ modification checks whether the commitment stores not already contain an argument for the given conclusion.

Definition 7.1.1. Let $\Xi = \{\langle x, \theta, assert(\langle \Phi_1, \phi_1 \rangle)\rangle, \ldots, \langle x, \theta, assert(\langle \Phi_k, \phi_k \rangle)\rangle\}$ be a set of legal assert moves that agent x could make. The function $pick_{a,com}$ returns the selected assert move to make. $pick_{a,com}(\Xi) = \langle x, \theta, assert(\langle \Phi_j, \phi_j \rangle)\rangle$ such that $1 \leq j \leq k$ and $\nexists \Phi' : \langle \Phi', \phi_j \rangle \in AT_{C(d_n,x) \cup C(d_n,\hat{x})}$.

$pick_{a,com}$ therefore picks the first legal assert move with a conclusion for which no argument is constructible from the union of the commitment stores.

Next, instead of looking at the argumentation theory of the union of the commitment stores, a $pick_a$ is defined that looks at the dialogue.

Definition 7.1.2. Let $\Xi = \{\langle x, \theta, assert(\langle \Phi_1, \phi_1 \rangle)\rangle, \ldots, \langle x, \theta, assert(\langle \Phi_k, \phi_k \rangle)\rangle\}$ be a set of legal assert moves that agent x could make. Also, let d_t^r be the current dialogue of length t, and $\theta \in \{ai, wi\}$. The function $pick_{a,diag}$ returns the selected assert move to make. $pick_{a,diag}(\Xi) = \langle x, \theta, assert(\langle \Phi_j, \phi_j \rangle)\rangle$ such that $1 \leq j \leq k$. Furthermore, $\nexists i$ such that $r \leq i \leq t$ and $\forall x \in \{P, O\} : \nexists \Phi' : d_t^r[i] = \langle x, \theta, assert(\langle \Phi', \phi_j \rangle)\rangle$.

$pick_{a.diag}$ therefore picks the first assert move with a conclusion for which no argument was asserted earlier in the dialogue.

7.1.2 Checking the status

For the second kind of duplicity, requiring that moves change the status of the root argument, merely looking at the dialogue does not make sense. Since the to-be-defined strategy will use the dialectical tree, it instead makes sense to define a dialectical tree built upon a dialogue.

Such a tree is interesting for two reasons. Firstly, it introduces a dependency on the order of moves in the dialogue. Secondly, the resulting tree should not differ much when compared to a 'proper' tree, since every asserted argument is also constructible from the commitment stores. It might even result in smaller dialectical trees and/or dialogues, since it only looks at the *assert* moves in the dialogue.

That notion of a dialectical tree is inspired by the dialectical graph used in Prakken (2005). The dialectical graph has also got a root argument, and is also based on the order of moves in a dialogue.

First, the modified notion of a tree is defined. Afterwards, the $pick_a$ functions are defined.

Another kind of tree

The dialectical tree *constructed from a dialogue* is therefore structured as follows. The first argument in the dialogue that supports the topic, is the root argument. Further on, if and only if an argument attacks another argument that is asserted at an *earlier* point in the dialogue, then the attacker is a child of the attacked argument.

Definition 7.1.3. Let d_t^r be the current dialogue, $x \in \{P, O\}$, $\theta \in \{ai, wi\}$, and A_0 an argument such that $d_t^r[k] = \langle x, \theta, assert(A_0)\rangle$ $(r \leq k \leq t)$. A *dialectical tree* **constructed from a dialogue** for A_0, constructed from d_t^r, denoted $T(A_0, d_t^r)$, is defined as follows

1. The root of the tree is labelled with A_0;
2. The argument A_0 appears at j in d_t^r such that $d_t^r[j] = \langle x, \theta, assert(A_0)\rangle$;
3. Let N be a node of the tree labelled A_n, and let $\Delta_i = [A_0, \ldots, A_n]$ be the sequence of labels on the path from the root to node N. Let B be a set of arguments defeating A_n such that $B = \{B_l | \exists l : 1 < l \leq t : d_t^r[l] = \langle x, \theta, assert(B_l)\rangle, B_l \text{ defeats } A_n\}$. Also, let A_n be asserted at m such that $d_t^r[m] = \langle x, \theta, assert(A_n)\rangle$ for some $x \in \{P, O\}$ and $\theta \in \{ai, wi\}$.

   For each defeater $C \in B$, if the argumentation line $\Delta_i' = [A_0, \ldots, A_n, C]$ is an acceptable argumentation line and C is asserted earlier in the dialogue than A_n $(\exists k : 1 \leq k < m : d_t^r[k] = \langle x, \theta, assert(C)\rangle$ for some $x \in \{P, O\}$ and $\theta \in \{ai, wi\})$, then the node N has a child N_j that is labelled C. If there is no defeater for A_n or there is no C such that Δ_j' is acceptable, then N is a leaf node.

Next, the status of an argument in a dialectical tree *constructed from a dialogue* is defined. This definition is similar to Definition 2.3.6. The difference being that the dialectical tree is constructed from a dialogue.

Definition 7.1.4. Let d_n be a dialogue wherein the dialectical tree is constructed from a dialogue. The *status* of an argument A given a set of rules and facts $\Psi \subseteq L_t$, the corresponding argumentation theory AT_Ψ, and the dialogue d_n is returned by the function $status : AT_\Psi \times \mathcal{D}_{top} \to \{in, out\}$. $status(A, d_n) = in$ iff A is marked in in $T(A, d_n)$. Otherwise, $status(A, d_n) = out$ (i.e. if A is marked out in $T(A, d_n)$).

The $pick_a$ functions

First, the $pick_a$ which uses Black and Hunter's notion of a dialectical tree is defined.

It will require assert moves to either change the status of the root argument, or leave the dialectical tree unchanged. The second condition is needed to allow agents to build up arguments. If, for example, in the argumentation theory of the joint knowledge bases, the argument $\langle\{a, a \Rightarrow b\}, b\rangle$ occurs, and if $(a, 1) \in \Sigma_O$ and $(a \Rightarrow b, 1) \in \Sigma_P$, then that argument can only be asserted if the agents are allowed to assert $\langle\{a\}, a\rangle$, even though it does not change the tree.

Definition 7.1.5. Let $\Xi = \{\langle x, \theta, assert(\langle\Phi_1, \phi_1\rangle)\rangle, \ldots, \langle x, \theta, assert(\langle\Phi_k, \phi_k\rangle)\rangle\}$ be a set of legal assert moves that agent x could make. The function $pick_{a', bhtree}$ returns the selected assert move to make. Let $CS(d_n) = C(d_n, x) \cup C(d_n, \hat{x})$ for the current and the other player. $pick_{a', bhtree}(\Xi) = \langle x, \theta, assert(\langle\Phi_j, \phi_j\rangle)\rangle$ such that, given d_n, $1 \leq j \leq k$, either

- $status(RootArg(d_n), CS(d_n) \cup \Phi_j) \neq status(RootArg(d_n), CS(d_n))$, or;
- $T(RootArg(d_n), CS(d_n) \cup \Phi_j) = T(RootArg(d_n), CS(d_n))$.

Second, the $pick_{a'}$ which uses a tree constructed from the dialogue is defined. It is similar to the above definition, but uses the modified dialectical tree.

Definition 7.1.6. Let $\Xi = \{\langle x, \theta, assert(\langle\Phi_1, \phi_1\rangle)\rangle, \ldots, \langle x, \theta, assert(\langle\Phi_k, \phi_k\rangle)\rangle\}$ be a set of legal assert moves that agent x could make. The function $pick_{a', diagtree}$ returns the selected assert move to make. $pick_{a', diagtree}(\Xi) = \langle x, \theta, assert(\langle\Phi_j, \phi_j\rangle)\rangle$ such that, given d_n and $1 \leq j \leq k$, either

- $status(RootArg(d_n), d_n \cup \langle x, \theta, assert(\langle\Phi_j, \phi_j\rangle)\rangle) \neq status(RootArg(d_n), d_n)$, or;
- $T(RootArg(d_n), d_n \cup \langle x, \theta, assert(\langle\Phi_j, \phi_j\rangle)\rangle) = T(RootArg(d_n), d_n)$.[1]

7.2 The strategies

Lastly, the modified strategies themselves are defined. The definitions are trivial, only using the modified $pick_a$ functions as defined above. Four strategies are defined, each using a different $pick_a$ function.

The strategies that limit that arguments are only asserted if no argument for that proposition was already asserted, are called the limited strategies. The strategies that 'smartly' check whether arguments change the root argument's status, are called the smart strategies.

First, the limited strategies are defined. Next, the smart strategies are defined.

[1] The union of the dialogue with the move represents the dialogue with that move added.

7.2.1 The limited strategies

The limited strategy that checks the commitment store, using $pick_{a,com}$, is called the limited commitment strategy.

Definition 7.2.1. The *limited commitment strategy* that **checks the commitment stores** for an agent x in a dialogue d_n is a function $S_{limCom} : \mathcal{D}_{top} \rightarrow M$. Let $T(d_n) = x$, $ctopic(d_n) = \gamma$ and $\theta \in \{ai, wi\}$:

$$S_{limCom}(d_n) = \begin{cases} pick_{a,com}(asserts(d_n, x)) & \text{if } asserts(d_n, x) \neq \emptyset \text{ and} \\ & pick_{a,com}(asserts(d_n, x)) \neq \emptyset \\ pick_o(opens(d_n, x)) & \text{if } asserts(d_n, x) = \emptyset \text{ and} \\ & opens(d_n, x) \neq \emptyset \\ \langle x, \theta, close(\theta, \gamma) \rangle & \text{if } asserts(d_n, x) = \emptyset \text{ and} \\ & opens(d_n, x) = \emptyset \end{cases}$$

where $asserts(d_n, x)$ returns all legal *assert* moves for agent x in dialogue d_n, and $opens(d_n, x)$ likewise returns all legal *open* moves.

Likewise, the limited strategy that checks the dialogue, using $pick_{a,diag}$, is called the limited dialogue strategy.

Definition 7.2.2. The *limited dialogue strategy* that **checks the dialogue** for an agent x in a dialogue d_n is a function $S_{limDiag} : \mathcal{D}_{top} \rightarrow M$. Let $T(d_n) = x$, $ctopic(d_n) = \gamma$ and $\theta \in \{ai, wi\}$:

$$S_{limDiag}(d_n) = \begin{cases} pick_{a,diag}(asserts(d_n, x)) & \text{if } asserts(d_n, x) \neq \emptyset \text{ and} \\ & pick_{a,diag}(asserts(d_n, x)) \neq \emptyset \\ pick_o(opens(d_n, x)) & \text{if } asserts(d_n, x) = \emptyset \text{ and} \\ & opens(d_n, x) \neq \emptyset \\ \langle x, \theta, close(\theta, \gamma) \rangle & \text{if } asserts(d_n, x) = \emptyset \text{ and} \\ & opens(d_n, x) = \emptyset \end{cases}$$

where $asserts(d_n, x)$ returns all legal *assert* moves for agent x in dialogue d_n, and $opens(d_n, x)$ likewise returns all legal *open* moves.

7.2.2 Smart strategies

Here, the strategies that attempt to reduce the second kind of redundancy (irrelevant arguments) are defined. They attempt to do so by requiring moves to be either strongly relevant (see Section 4.4), or not modify the dialectical tree. Strong relevance is used by Prakken (2005) to define more coherent persuasion dialogues. Here, it is used in an attempt to reduce the number of asserted irrelevant arguments. Arguments that do not change the dialectical tree are always allowed to allow agents to 'build' arguments.

The smart strategy using Black and Hunter's dialectical tree, is called the smart original strategy.

Definition 7.2.3. The *smart original strategy* **with Black and Hunter's notion** of a dialectical tree, for an agent x in a dialogue d_n is a function

$S_{smart,bhtree} : \mathcal{D}_{top} \to M$. Let $T(d_n) = x$, $ctopic(d_n) = \gamma$ and $\theta \in \{ai, wi\}$:

$$S_{smart,bhtree}(d_n) = \begin{cases} pick_{a',bhtree}(asserts(d_n, x)) & \text{if } asserts(d_n, x) \neq \emptyset \text{ and} \\ & pick_{a',bhtree}(asserts(d_n, x)) \neq \emptyset \\ pick_o(opens(d_n, x)) & \text{if } asserts(d_n, x) = \emptyset \text{ and} \\ & opens(d_n, x) \neq \emptyset \\ \langle x, \theta, close(\theta, \gamma) \rangle & \text{if } asserts(d_n, x) = \emptyset \text{ and} \\ & opens(d_n, x) = \emptyset \end{cases}$$

where $asserts(d_n, x)$ returns all legal *assert* moves for agent x in dialogue d_n, and $opens(d_n, x)$ likewise returns all legal *open* moves.

Secondly, the smart exhaustive strategy using the new dialectical tree (and $pick_{a',diagtree}$) is defined. It is called the smart dialogue strategy.

Definition 7.2.4. The *smart dialogue strategy* **using the dialectical tree based on a dialogue** (Definition 7.1.3), for an agent x in a dialogue d_n is a function $S_{smart,diagtree} : \mathcal{D}_{top} \to M$. Let $T(d_n) = x$, $ctopic(d_n) = \gamma$ and $\theta \in \{ai, wi\}$:

$$S_{smart,diagtree}(d_n) = \begin{cases} pick_{a',diagtree}(asserts(d_n, x)) & \text{if } asserts(d_n, x) \neq \emptyset \text{ and} \\ & pick_{a',diagtree}(asserts(d_n, x)) \neq \emptyset \\ pick_o(opens(d_n, x)) & \text{if } asserts(d_n, x) = \emptyset \text{ and} \\ & opens(d_n, x) \neq \emptyset \\ \langle x, \theta, close(\theta, \gamma) \rangle & \text{if } asserts(d_n, x) = \emptyset \text{ and} \\ & opens(d_n, x) = \emptyset \end{cases}$$

where $asserts(d_n, x)$ returns all legal *assert* moves for agent x in dialogue d_n, and $opens(d_n, x)$ likewise returns all legal *open* moves.

7.3 Conclusion

The modifications of the exhaustive strategy are inspired by the intake. The limited strategies will not assert 'duplicate' arguments, to avoid having multiple arguments supporting the same conclusion. The smart strategies check whether an argument changes the root argument's status, in order to avoid asserting irrelevant arguments.

In short, the **exhaustive strategy** tries to do the following things in order.

1. assert an argument if possible
2. open any sub-dialogue
3. (try to) close the current sub-dialogue

The other strategies are a modification of the exhaustive strategy. They replace the first item with a stricter condition on asserting arguments. The modified strategies therefore require the following conditions to be met before an argument can be asserted.

The **limited dialogue** strategy only asserts an argument if there exists an argument that supports a proposition for which no argument has been asserted yet.

The **limited commitment** strategy only asserts an argument if there exists an argument that supports a proposition for which no argument can be derived from the union of the commitment stores.

The **smart dialogue** strategy uses the dialectical tree based on the dialogue. Moreover, it only asserts an argument if that argument either changes the status of the root argument of the dialectical tree, or if it does not change the dialectical tree.

The **smart original** strategy uses Black and Hunter (2009) definition of a dialectical tree. It also only asserts if an argument either changes the status of the root argument of the dialectical tree, or if it does not change the dialectical tree.

Whether these strategies are sound and complete is investigated in the next chapter.

Chapter 8

Proving soundness and completeness

This chapter will determine whether the modified strategies are sound and complete in the sense of Section 6.1 (as defined by Black and Hunter (2009)).

Important to note is that a strategy is sound and/or complete if and only if all dialogues it generates are either sound or complete.

8.1 Limited strategies

Here, first will soundness and completeness be checked for argument inquiry, and then for warrant inquiry.

8.1.1 Argument inquiry

Both limited strategies are sound for argument inquiry dialogues. First lemmas adopted from Black and Hunter (2009) are used to say that whatever is asserted in an argument inquiry dialogue, comes from the union of the agents' beliefs. (i.e. the commitment stores are a subset of the agents' beliefs)

Lemma 8.1.1. If d_t^r is an argument inquiry dialogue, then $C(d_t^r, x) \cup C(d_t^r, \hat{x}) \subseteq \Sigma_P \cup \Sigma_O$.

Proof. The only time the commitment store changes, is when an agent x utters $\langle x, ai, assert(\langle \Phi, \phi \rangle) \rangle$. Following the protocol for argument inquiry dialogues (Definition 5.9.1) and the update function for commitment stores (Definition 5.4.2), after uttering that move it must hold that $\Phi \in C(d_t^r, x) \cup C(d_t^r, \hat{x})$. Since at the beginning of the dialogue the commitment stores are empty, any member of the union of the commitment stores must also be a member of the union of the agents' beliefs. Therefore, $C(d_t^r, x) \cup C(d_t^r, \hat{x}) \subseteq \Sigma_P \cup \Sigma_O$. □

Lemma 8.1.2. Let $\Phi \subseteq L_t$ and $\Psi \in L_t$ be two sets. If $\Phi \subseteq \Psi$, then $AT_\Phi \subseteq AT_\Psi$.

Proof. Assume that $\Phi \subseteq \Psi$ and that $\langle \Pi, \pi \rangle$ is an argument such that $\langle \Pi, \pi \rangle \in AT_\Phi$. From the definition of an argument (Definition 2.2.1), it holds that $\Pi \subseteq \Phi$.

As $\Pi \subseteq \Phi$ and $\Phi \subseteq \Psi$, $\Pi \subseteq \Psi$ holds. Therefore, $\langle \Pi, \pi \rangle \in AT_\Psi$ (Definition 2.2.1). Therefore, if $\Phi \subseteq \Psi$ and $A \in AT_\Phi$ then $A \in AT_\Psi$, and $AT_\Phi \subseteq AT_\Psi$. □

Using these lemmas, it is proven that the limited strategies, the limited commitment as well as the limited dialogue strategy, are sound.

Proposition 8.1.1. The limited dialogue and limited commitment strategy are sound.

Proof. Let d_t^r be an argument inquiry dialogue formed by either the limited dialogue, or limited commitment strategy. Neither strategy changes the definition of the outcome or the protocol. Let $\langle \Phi, \phi \rangle \in outcome_{ai}(d_t^r)$. From the definition of the outcome (Definition 6.1.1), $\langle \Phi, \phi \rangle \in AT_{C(d_t^r,x) \cup C(d_t^r,\hat{x})}$. From Lemma 8.1.1, $C(d_t^r, x) \cup C(d_t^r, \hat{x}) \subseteq \Sigma_P \cup \Sigma_O$. Therefore, following Lemma 8.1.2, $\langle \Phi, \phi \rangle \in AT_{\Sigma_P \cup \Sigma_O}$. Therefore, the argument inquiry dialogues formed by the limited dialogue strategy, or the limited commitment strategy are sound. □

This proof closely follows the proof of soundness of Black and Hunter for the exhaustive strategy (Black & Hunter, 2009, p. 199). Since neither limited strategy changes any condition for an argument to be in the commitment store, that proof still holds. The limited strategies are therefore sound because any argument in the union of the commitment stores is also derivable from the union of the agents' beliefs.

As for completeness (in the sense of Definition 6.1.3), neither strategy is complete.

Proposition 8.1.2. The limited dialogue and limited commitment strategy are not complete for argument inquiry dialogues.

Proof. Let d_t^r be a dialogue, terminated at t, such that $\langle \{a, a \Rightarrow b\}, b \rangle, \langle \{c, c \Rightarrow b\}, b \rangle \in AT_{\Sigma_O \cup \Sigma_P}$. Also, assume that the dialogue was generated by either the limited dialogue strategy, or the limited commitment strategy. Then it cannot be that $\langle \{a, a \Rightarrow b\}, b \rangle \in AT_{C(d_t^r,x) \cup C(d_t^r,\hat{x})}$ ánd $\langle \{c, c \Rightarrow b\}, b \rangle \in AT_{C(d_t^r,x) \cup C(d_t^r,\hat{x})}$.

Assume that $\langle \{a, a \Rightarrow b\}, b \rangle$ was asserted at some point in the dialogue. Then, $\langle \{c, c \Rightarrow b\}, b \rangle$ cannot have been asserted at some point afterwards. Why, depends on the strategy. If the limited dialogue strategy generated d_t^r, then dialogue already contained an assert supporting b. Similarly, with the limited commitment strategy the union of the commitment stores already contained an argument for b. Assuming that $\langle \{c, c \Rightarrow b\}, b \rangle$ was asserted first, the same story holds. Therefore, neither the limited commitment or limited dialogue strategy are complete for argument inquiry dialogues. □

Both strategies are not complete. the condition that only one argument per proposition may exist, adds the possibility for the strategy to keep argumentation lines out of the dialectical that would otherwise have changed the status of the root argument.

8.1.2 Warrant inquiry

Just like for argument inquiry dialogues, the limited strategies do, however, not generate sound dialogues for warrant inquiry dialogues. Take for example the knowledge bases as described in Figure 8.1.1. It shows the dialectical tree as

constructed when using the exhaustive strategy. The problem is, that using the limited dialogue or commitment strategy, two different trees can be generated. Those trees each assign a different status to the root argument. The reason is that the full tree (Figure 8.1.1) has got two arguments for the same proposition $\neg b$. Using the limited strategies only one of them makes it way into the tree. The problem is that $(\{c, c \Rightarrow \neg b\}, \neg b)$ is a leaf, and therefore *in*, while $(\{d, d \Rightarrow \neg b\}, \neg b)$ is *out* (see figure 8.1.2). Which status is assigned to the root argument therefore depends on which of those two arguments is asserted first. After all, after $(\{c, c \Rightarrow \neg b\}, \neg b)$ is asserted, $(\{d, d \Rightarrow \neg b\}, \neg b)$ can never be asserted, as there already exists an argument for $\neg b$.

Note that this example, and the following proof, both hold for the limited dialogue and limited commitment strategy. That is because both strategies prevent one to assert multiple arguments that support the same proposition.

Proposition 8.1.3. The limited dialogue and limited commitment strategy are not sound for warrant inquiry dialogues.

Proof. Let $\Sigma_P = \{(a \Rightarrow b, 4), (c \Rightarrow \neg b, 3), (d \Rightarrow \neg b, 3)\}$. Furthermore, let $\Sigma_O = \{(a, 4), (c, 3), (d, 3), (\neg d, 2)\}$. The limited strategies generate the dialogue shown in Table 8.1.1. Irrespectively of the specific limited strategy, after $\langle \{f, f \Rightarrow \neg b\}, \neg b \rangle$ is asserted, no other argument for $\neg b$ will be asserted. The corresponding dialectical tree is shown on the right in Figure 8.1.2. There, $(\{a, a \Rightarrow b\}, b)$ is marked *in*, while $status((\{a, a \Rightarrow b\}, b), AT_{\Sigma_O \cup \Sigma_P}) = out$. The marking of the root argument in Table 8.1.1 differs from its status in the union of the knowledge bases. The dialogue is, therefore, *not* sound. The limited exhaustive strategy therefore is not sound. □

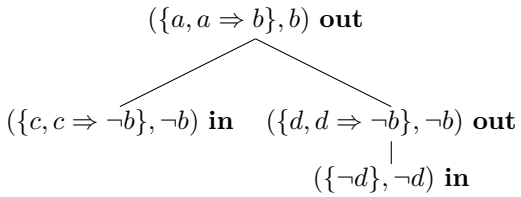


Figure 8.1.1: The (marked) dialectical tree generated by using the exhaustive strategy with $\Sigma_P = \{(a \Rightarrow b, 4), (c \Rightarrow \neg b, 3), (d \Rightarrow \neg b, 3)\}$, $\Sigma_O = \{(a, 4), (c, 3), (d, 3), (\neg d, 2)\}$ and a warrant inquiry dialogue about b.

Figure 8.1.2: The two possible (marked) dialectical trees generated by using a limited strategy with $\Sigma_P = \{(a \Rightarrow b, 4), (c \Rightarrow \neg b, 3), (d \Rightarrow \neg b, 3)\}$, $\Sigma_O = \{(a, 4), (c, 3), (d, 3), (\neg d, 2)\}$ and a warrant inquiry dialogue about b.

The limited strategies are, likewise, not complete. This is because leaving out an argument can make the root argument a different status.

$\langle P, wi, open(wi, (\{b\}, b))\rangle$
$\langle O, wi, close(b)\rangle$
$\langle P, wi, open(ai, a \Rightarrow b)\rangle$
$\quad \langle O, ai, assert(\langle\{a\}, a\rangle)\rangle$
$\quad \langle P, ai, assert(\langle\{a, a \Rightarrow b\}, b\rangle)\rangle$
$\quad \langle O, ai, close(a, b)\rangle$
$\quad \langle P, ai, close(a, b)\rangle$
$\langle O, wi, close(b)\rangle$
$\langle P, wi, open(ai, f \Rightarrow \neg b)\rangle$
$\quad \langle O, ai, assert(\langle\{f\}, f\rangle)\rangle$
$\quad \langle P, ai, assert(\langle\{f, f \Rightarrow \neg b\}, \neg b\rangle)\rangle$
$\quad \langle O, ai, close(\neg b, f)\rangle$
$\quad \langle P, ai, open(ai, c \Rightarrow \neg b)\rangle$
$\quad\quad \langle O, ai, assert(\langle\{c\}, c\rangle)\rangle$
$\quad\quad \langle P, ai, close(\neg b, c)\rangle$
$\quad\quad \langle O, ai, close(\neg b, c)\rangle$
$\quad \langle P, ai, close(\neg b, f)\rangle$
$\quad \langle O, ai, close(\neg b, f)\rangle$
$\langle P, wi, close(b)\rangle$
$\langle O, wi, assert(\langle\{\neg f\}, \neg f\rangle)\rangle$
$\langle P, wi, close(b)\rangle$
$\langle O, wi, close(b)\rangle$

Table 8.1.1: A possible dialogue using a limited strategy and the knowledge bases shown in figure 8.1.1

Proposition 8.1.4. The limited dialogue and limited commitment strategy are both *not* complete for warrant inquiry dialogues.

Proof. Let d_t^r be a warrant inquiry dialogue terminated at t, generated by one of the limited strategies[1], with $\Sigma_P = \{p \Rightarrow q, r \Rightarrow \neg p, s \Rightarrow \neg r, t \Rightarrow \neg r, u \Rightarrow \neg t\}$ and $\Sigma_O = \{p, r, s, t, u\}$. The goal of the dialogue is to find an acceptable argument for q. Figure 8.1.3 shows the dialectical tree generated from the union of the agents' beliefs. At some point j ($r \leq j \leq t$) in the dialogue, the dialectical tree shown in Figure 8.1.4 will have been generated. Following the limited strategies (Definitions 7.2.1 and 7.2.2), either $\langle\{t, t \Rightarrow \neg r\}, \neg r\rangle$ or $\langle\{s, s \Rightarrow \neg r\}, \neg r\rangle$ will then be asserted at some point after j. Assume, however, $\langle\{t, t \Rightarrow \neg r\}, \neg r\rangle$ is asserted in d_t^r. Following the limited strategies, next $\langle\{u, u \Rightarrow \neg t\}, \neg t\rangle$ is asserted, after which the dialogue is terminated. Note that the root argument in the dialectical tree constructed from the agents' beliefs is *in*. Therefore, $status(\langle\{p, p \Rightarrow q\}, q\rangle, \Sigma_P \cup \Sigma_O) = in$. However, $status(\langle\{p, p \Rightarrow q\}, q\rangle, C(d_t^r, x) \cup C(d_t^r, \hat{x})) = out$, therefore $outcome_{wi}(d_t^r) = \emptyset$. The limited strategies are therefore not complete for warrant inquiry dialogues, since $status(\langle\{p, p \Rightarrow q\}, q\rangle, \Sigma_P \cup \Sigma_O) = in$, but $outcome_{wi}(d_t^r) = \emptyset$. □

[1] In this case the specific variant does not matter.

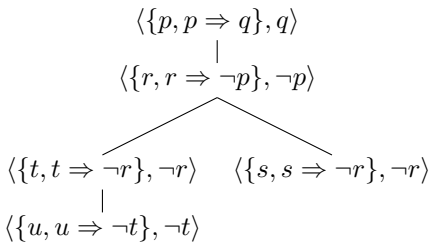


Figure 8.1.3: The dialectical tree generated by the exhaustive strategy in a warrant inquiry dialogue, with $\Sigma_P = \{p \Rightarrow q, r \Rightarrow \neg p, s \Rightarrow \neg r, t \Rightarrow \neg r, u \Rightarrow \neg t\}$ and $\Sigma_O = \{p, r, s, t, u\}$.

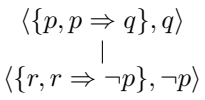


Figure 8.1.4: The dialectical tree generated by a limited strategy at some point in the dialogue, with $\Sigma_P = \{p \Rightarrow q, r \Rightarrow \neg p, s \Rightarrow \neg r, t \Rightarrow \neg r, u \Rightarrow \neg t\}$ and $\Sigma_O = \{p, r, s, t, u\}$.

8.2 Smart strategy

This section will prove for both smart strategies whether they are sound and/or complete for warrant inquiry dialogues. Here, only warrant inquiry dialogues are considered since a dialectical tree is used in the strategy. Therefore, it only operates on warrant inquiry dialogues. Like with the limited strategies, first is examined whether the strategy is sound, then whether it is complete.

8.2.1 Soundness

The smart dialogue strategy is first investigated to see whether it is sound. Afterwards, the smart original strategy is investigated for the same thing.

The smart dialogue strategy is not sound. There exists a situation in which the dialectical tree of the generated dialogue gives a different status to its root argument than the dialectical tree of the union of the agents' beliefs.

Proposition 8.2.1. The smart dialogue strategy is not sound for warrant inquiry dialogues.

Proof. Let $\Sigma_P = \{a \Rightarrow b, d \Rightarrow \neg b, e \Rightarrow \neg d, c \Rightarrow \neg b, f \Rightarrow \neg c\}$ and $\Sigma_O = \{a, c, d, e, f\}$. Let d_t^r be a warrant inquiry dialogue of t long, generated by the smart dialogue strategy, with b as its initial topic. Table 8.2.1 (page 51) shows a warrant inquiry dialogue generated using the previous settings. Figure 8.2.1 shows the dialectical tree generated by that dialogue, and Figure 8.2.2 shows the dialectical tree of the union of the agents' beliefs. The figures clearly show the root argument having a different status than the root argument in the other dialectical tree. The smart dialogue strategy therefore does is not sound. □

In the dialogue shown in Table 8.2.1 something interesting happens. While $\langle\{f, f \Rightarrow \neg c\}, \neg c\rangle$ is asserted, it does not make it into the dialectical graph. The reason being that $\langle\{c, c \Rightarrow \neg b\}, \neg b\rangle$ is asserted *after* $\langle\{f, f \Rightarrow \neg c\}, \neg c\rangle$.

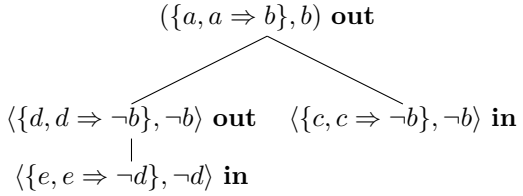


Figure 8.2.1: The dialectical tree as created by the dialogue of Table 8.2.1, with $\Sigma_P = \{a \Rightarrow b, d \Rightarrow \neg b, e \Rightarrow \neg d, c \Rightarrow \neg b, f \Rightarrow \neg c\}$, $\Sigma_O = \{a, c, d, e, f\}$ and a warrant inquiry dialogue about b.

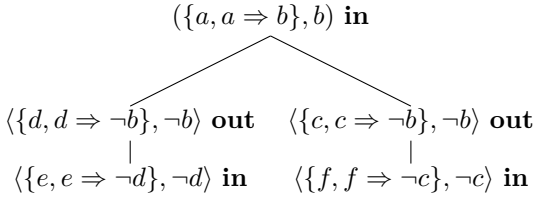


Figure 8.2.2: The dialectical tree as generated by the union of the agents' beliefs, using $\Sigma_P = \{a \Rightarrow b, d \Rightarrow \neg b, e \Rightarrow \neg d, c \Rightarrow \neg b, f \Rightarrow \neg c\}$ and $\Sigma_O = \{a, c, d, e, f\}$.

So, in this particular dialogue $\langle\{c, c \Rightarrow \neg b\}, \neg b\rangle$ was not asserted early enough. In fact, after the move $\langle O, ai, assert(\langle\{c\}, c\rangle)\rangle$ the argument was known by the proponent. If it wasn't for the strategy, the argument could have been asserted. Ultimately, the argument was not asserted, since it would change the dialectical tree, but not the status of the root argument (see Figure 8.2.3). For a sound dialogue, $\langle\{c, c \Rightarrow \neg b\}, \neg b\rangle$ should even have been asserted *before* $\langle\{d, d \Rightarrow \neg b\}, \neg b\rangle$. The smart strategy could have done this. After the argument inquiry sub-dialogue about $a \Rightarrow b$ it chose to open a sub-dialogue for $d \Rightarrow \neg b$, instead of $c \Rightarrow \neg b$. Choosing the right sub-dialogue would have generated a sound dialogue. The smart dialogue strategy therefore did not sufficiently take the consequences of opening a sub-dialogue into account.

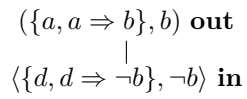


Figure 8.2.3: The dialectical tree of Table 8.2.1 just after the move $\langle O, ai, assert(\langle\{c\}, c\rangle)\rangle$.

On the other hand, the smart original strategy (which uses Black and Hunter's dialectical tree) *is* sound.

Proposition 8.2.2. The smart original strategy is sound for warrant inquiry dialogues.

Proof. Let d_t^r be a dialogue terminated at t, with $RootArg(d_t^r) = \langle\Phi, \phi\rangle$, such that $outcome_{wi}(d_t^r) = \langle\Phi, \phi\rangle$. Following Definition 6.1.4, $status(\langle\Phi, \phi\rangle, C(d_t^r, x) \cup C(d_t^r, \hat{x})) = in$. Following Definition 7.2.3, all arguments that would have changed the status of the root argument have been asserted.

Therefore, no argument in $\Sigma_P \cup \Sigma_O \setminus C(d_t^r, x) \cup C(d_t^r, \hat{x})$ can cause $\langle\Phi, \phi\rangle$ to be *out*. Therefore, $status(\langle\Phi, \phi\rangle, \Sigma_P \cup \Sigma_O) = in$. Hence, all dialogues generated

by the smart original strategy are necessarily sound, and therefore the strategy itself is sound. □

The smart original strategy is sound, since if an argument is found to be acceptable in a dialogue, then that argument is also acceptable in the union of the agents' beliefs. After all, all arguments that could cause the root argument to be *out* are defeated by other arguments, since the smart original strategy asserts all arguments which change the status of the root argument.

8.2.2 Completeness

Like in the previous section first the smart dialogue strategy is investigated. The difference being that now is being checked whether the strategy is complete. Afterwards, the same is done for the smart original strategy.

The smart dialogue strategy is not complete, due to the same reason that it is not sound.

Proposition 8.2.3. The smart dialogue strategy is not complete for warrant inquiry dialogues.

Proof. Let d_t^r be a dialogue, terminated at t, with a root argument such that $status(RootArg(d_t^r), \Sigma_O \cup \Sigma_P) = in$. Table 8.2.1 shows a dialogue where that is the case. However, as explained in Proposition 8.2.1, the status of the root argument $\langle \{a, a \Rightarrow b\}, b \rangle$ is *out* in the dialogue, while its status in the full tree (Figure 8.2.2) is *in*. Therefore, the smart dialogue strategy is not complete. □

The smart original strategy does not suffer from this flaw, and is complete. The crux being that if an argument is not asserted by the smart original strategy, then no argumentation line is created which would make the root argument *out*.

Proposition 8.2.4. The smart original strategy is complete (following Definition 6.1.6) for warrant inquiry dialogues.

Proof. Let d_t^r be a dialogue terminated at t, with $RootArg(d_t^r) = \langle \Phi, \phi \rangle$ and $status(\langle \Phi, \phi \rangle, \Sigma_O \cup \Sigma_P) = in$. It places, when compared to the exhaustive strategy, a restriction upon *assert* moves. In order for the smart original strategy to assert an argument, it should (1) either change the root argument's status, or (2) not change the dialectical tree.

The smart original strategy therefore asserts all arguments that have an influence on the status of the root argument. It also allows the agents to build up all of these arguments. Following the protocol for warrant inquiry dialogues and the smart original strategy, argument inquiry sub-dialogues are opened for every rule that might end up being an attacking argument. During these argument inquiry sub-dialogues, the premises supporting an argument containing the topic are still asserted. These arguments do not change the dialectical tree, therefore the smart original strategy still asserts them. The only arguments not asserted, are arguments that do not change the root argument's status, but would change the dialectical tree.

Therefore, all arguments that might influence the root argument's status will be at some point in the dialogue legal *assert* moves. All of these arguments are asserted by the smart original strategy. Therefore, if in the union of the agents' beliefs the root argument is *in*, the outcome of the warrant inquiry dialogue

generated by the smart original strategy will be that root argument. The smart original strategy is therefore complete. □

8.3 Conclusion

Of all modified strategies, only the smart original strategy is both sound *and* complete. The limited strategies are only sound in argument inquiry dialogues. For warrant inquiry dialogues they are not sound and complete because it is possible that an argument that would change the root argument's status is not asserted. Table 8.3.1 shows an overview over what was proven in this chapter.

Nevertheless, the next chapter will investigate the usefulness of the modified strategies.

$\langle P, wi, open(b)\rangle$
$\langle O, wi, close(b)\rangle$
$\langle P, wi, open(ai, a \Rightarrow b)\rangle$
    $\langle O, ai, assert(\langle\{a\}, a\rangle)\rangle$
    $\langle P, ai, assert(\langle\{a, a \Rightarrow b\}, b\rangle)\rangle$
    $\langle O, ai, close(a, b)\rangle$
    $\langle P, ai, close(a, b)\rangle$
$\langle O, wi, close(b)\rangle$
$\langle P, wi, open(ai, d \Rightarrow \neg b)\rangle$
    $\langle O, ai, assert(\langle\{d\}, d\rangle)\rangle$
    $\langle P, ai, assert(\langle\{d, d \Rightarrow \neg b\}, \neg b\rangle)\rangle$
    $\langle O, ai, close(\neg b, d)\rangle$
    $\langle P, ai, open(ai, c \Rightarrow \neg b)\rangle$
        $\langle O, ai, assert(\langle\{c\}, c\rangle)\rangle$
        $\langle P, ai, close(\neg b, c)\rangle$
        $\langle O, ai, close(\neg b, c)\rangle$
    $\langle P, ai, close(\neg b, d)\rangle$
    $\langle O, ai, close(\neg b, d)\rangle$
$\langle P, wi, open(ai, f \Rightarrow \neg c)\rangle$
    $\langle O, ai, assert(\langle\{f\}, f\rangle)\rangle$
    $\langle P, ai, assert(\langle\{f, f \Rightarrow \neg c\}, \neg c\rangle)\rangle$
    $\langle O, ai, close(\neg c, f)\rangle$
    $\langle P, ai, close(\neg c, f)\rangle$
$\langle O, wi, close(b)\rangle$
$\langle P, wi, open(ai, e \Rightarrow \neg d)\rangle$
    $\langle O, ai, assert(\langle\{e\}, e\rangle)\rangle$
    $\langle P, ai, assert(\langle\{e, e \Rightarrow \neg d\}, \neg d\rangle)\rangle$
    $\langle O, ai, close(e, \neg d)\rangle$
    $\langle P, ai, close(e, \neg d)\rangle$
$\langle O, wi, close(b)\rangle$
$\langle P, wi, assert(\langle\{c, c \Rightarrow \neg b\}, \neg b\rangle)\rangle$
$\langle O, wi, close(b)\rangle$
$\langle P, wi, close(b)\rangle$

Table 8.2.1: A counterexample to show that the smart dialogue strategy does not always generate sound dialogues. Uses the knowledge bases as defined in Figure 8.2.1.

	Argument inquiry		Warrant inquiry	
	Sound	Complete	Sound	Complete
Limited dialogue	Yes	No	No	No
Limited commitment	Yes	No	No	No
Smart dialogue	N/A	N/A	No	No
Smart original	N/A	N/A	Yes	Yes

Table 8.3.1: An overview of whether the modified strategies are sound and/or complete according to the definitions in Chapter 6.

Chapter 9

Experiments

In Chapter 7, modifications to the exhaustive strategy were proposed. Some of these new strategies were sound and/or complete. The practical effect of these strategies on warrant or argument inquiry dialogues is, however, not yet clear. This chapter therefore simulated dialogues in order to investigate the effect of those strategies in a setting that is as close as possible to intakes. It also simulated dialogues in a setting where the distribution of knowledge is random, to investigate the influence of the previously mentioned setting.

The strategies were compared using a method adopted from Hecham et al. (2018). Hecham et al. compared different tools for defeasible reasoning (e.g. ASPIC+, DeLP, etc.). They subjected the tools to different kinds of inference graphs. Their goal being to compare the tools on their performance and expressibility.

Here, the goal was to subject the strategies to different kinds of dialectical trees, in order to test them in different situations. For this, three inference graphs of Hecham et al. were taken and changed into equivalent dialectical trees. The rules and facts used to form these dialectical trees were then distributed over the participants, and a dialogue was simulated.

Those rules and facts were distributed in two ways. Firstly, they were distributed such that the proponent knows all the rules and the opponent all the facts. This distribution of knowledge tried to model the asymmetricity that is present in intakes. Secondly, the rules and facts were distributed randomly over all participants, such that each rule/fact had an equal chance of being in either the proponent's or opponent's knowledge. Per strategy, data set and a certain size of that data set, 30 dialogues with each a different random distribution of knowledge were simulated. This way, the effect of the first distribution of knowledge on the behaviour of the strategies was investigated.

Further on, the experiments that were done using a random distribution of knowledge are called the random tests, while the other tests were 'set in an intake setting'. Those tests will be called the 'normal' tests.

In the following, first the goal of the experiments will be clarified. Next, will be discussed using what means the experiments were conducted. Lastly, the setup of the experiments is explained, as well as the results of the experiments. The chapter ends with a discussion about the results.

9.1 Research questions

The goal of the modified strategies is ultimately to reduce the size of the dialogues and/or the dialectical trees when compared to the exhaustive strategy. The experiments therefore investigated whether those strategies actually accomplish that goal. For this, they were first tested in an intake setting.[1]

The research questions are twofold. Firstly, the goal was to find out what difference the modified strategies make in the intake setting within different data sets. Secondly, it was investigated how much of an influence the intake setting is on these strategies. The first goal was achieved by comparing, within the intake setting and within a certain data set, the modified strategies to the exhaustive strategy. The second goal was achieved by comparing each modified strategy within the intake setting to its counterpart in the random setting.

From this, the following research questions arose:

1. Within the same data set;
   (a) What difference do the modified strategies in the intake setting make in terms of dialogue and dialectical tree size, when compared to the exhaustive strategy?
   (b) Do the strategies in the random setting each show the same behaviour as the strategies in the intake setting? In other words, how much of an influence is the intake setting on the behaviour of the strategies?

9.2 Methods

As mentioned at the beginning of this chapter, dialectical trees based on the inference graphs used in Hecham et al. (2018) were used to test the modified strategies. The modified strategies were tested to investigate the practical effect of the modifications to the exhaustive strategy.

To test these dialectical trees (or data sets), an implementation of the dialogue system of Black and Hunter (2009) was developed. This implementation is one of the few open source dialogue engines. It can be found on https://gitlab.com/Kumeling/agentdialogues. The implementation also allows one to easily program new dialogue systems, or simulate dialogues using different knowledge bases.

Each combination of a strategy and data set was tested in two ways. The difference being whether the distribution of knowledge was generated randomly or not.

This section will first explain exactly what data sets were used, secondly how these were used and thirdly what implementation was used.

9.2.1 The data sets

In total, three inference graphs (data sets) from Hecham et al. (2018) were used to investigate the effect of the modified strategies in controversial situations. These graphs were used by Hecham et al. to investigate the semantics of several tools for defeasible logic.

[1] Where the proponent knows all of the rules, and the opponent all of the facts.

Here, the data sets were used to investigate the effect of the modified strategies on dialogues generated using these controversial situations. They were *not* used to test the semantics or expressibility of the underlying logic (a defeasible variant of DeLP).

The following data sets were used. It is also explained what situation the data sets try to test.

1. Ambiguity handling: How does the dialogue handle information derived from an argument that is attacked?

2. Team defeat: Whether an argument can be defended by another argument, or whether the argument must be more preferred than its attacking counterpart to defend itself.

3. Floating conclusions: If two arguments attack each other, but later reach the same conclusion, is that conclusion then valid, or contested?

The data sets were mainly chosen for the specific shape of their inference graph. Ambiguity handling, for example, was chosen in order to see what the effect of having series of attackers would be on the modified strategies. Team defeat was chosen to test how the modified strategies would behave in a situation where some arguments are attacked by two arguments.

The floating conclusion data set was, however, chosen for a different reason. It was chosen because its equivalent dialectical tree is always two arguments large. This will hopefully help show more clearly what effect the modified strategies have on the size of a dialogue.

The inference graphs

Figure 9.2.1 shows per data set the structure of the corresponding inference graph. The arrows represent defeasible implication, the $\top$'s represent a proposition that is always true, and dotted lines represent attacks. Note that the data sets as used by Hecham et al. assume the usage of first order logic.

Take for example the ambiguity data set. Here, $s_0(a)$, $q_0(a)$ and $p_0(a)$ follow necessarily from T. Also, from $q_0(a)$ defeasibly follows $q_1(a)$, and $q_2(a)$ follows defeasibly from $q_1(a)$, etc., until $\neg p_{2n}(a)$ follows defeasibly from $q_{2n}(a)$. The same holds for $p_0(a)$, which ends in $p_{2n}(a)$. The dotted line between $p_{2n}(a)$ and $\neg p_{2n}(a)$ shows that both propositions conflict.

Adoption

As already mentioned, the data sets used to run the simulations use dialectical trees similar in structure to the above data sets. In general, the dialectical trees are similar to the inference graphs shown in Figure 9.2.1.

Team defeat is, however, modelled differently. It is modelled such that the joint knowledge bases produce a dialectical tree in which the root argument is attacked by two arguments and defended by two arguments. Those defenders are in turn attacked and defended by four arguments (two attackers, two defenders). This results in n layers of arguments that are attacked and defended. Figure 9.2.2 shows the dialectical tree for team defeat for $n = 2$.

Therefore, in team defeat n layers of arguments exist. At the first layer one argument is attacked by two arguments. These arguments are in turn attacked

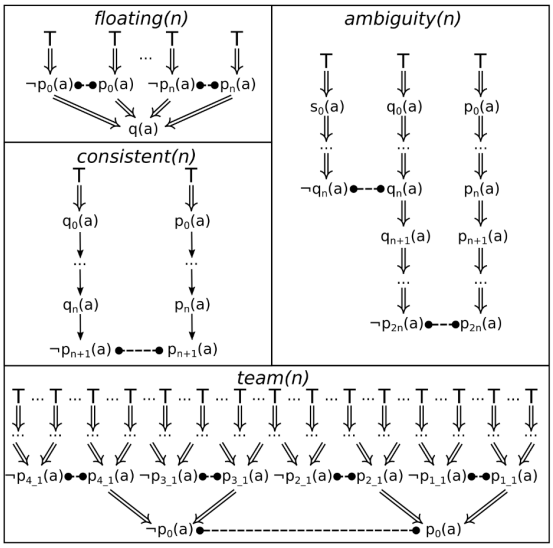


Figure 9.2.1: The different types of inference graphs as used by Hecham et al. (2018). The tests are parametrized by n to allow one to test with arbitrary-sized graphs. $floating(n)$ is the test used for floating conclusions, $ambiguity(n)$ is used for ambiguity handling, $team(n)$ is used for team defeat and $consistent(n)$ is not used. Note that first order logic is used. Adopted from Hecham et al. (2018).

each attacked by an argument, who therefore defend the original argument. These defenders (on the next layer) are in turn attacked and defended by arguments. This goes on n times, such that n 'layers' of arguments exist. In total, team defeat contains for a given n, $2^{n+2} - 3$ arguments.[2]

The floating conclusions data set contains $2n$ facts, and $2n$ rules. For each j such that $1 \leq j \leq n$, there exists p_j, $\neg p_j$, $p_j \Rightarrow q$ and $\neg p_j \Rightarrow q$. The topic of the dialogue is q. In total the data set contains $2n$ facts, $2n$ rules, and therefore $2n$ arguments. Note for an instance of this data set multiple dialectical trees are possible. The reason being that a dialectical tree can only have one root. Figure 9.2.3 shows, for example, all possible dialectical trees for when $n = 2$.

The ambiguity data set contains the rules $s_n \Rightarrow q$, $p_n \Rightarrow \neg q$, $r_{n/2} \Rightarrow \neg p_{n/2}$, and the facts s_n, p_n and $r_{n/2}$. It also contains n rules, where i s.t. $1 \leq i < n$, $s_i \Rightarrow \neg s_{i+1}$, s_i, $p_i \Rightarrow \neg p_{i+1}$ and p_i. Moreover, it also contains $n/2$ rules, with j s.t. $1 \leq j < n/2$, $r_i \Rightarrow r_{i+1}$ and r_i. The topic of the dialogue is q. In total, it contains $\lfloor 1\frac{1}{2}n \rfloor + 3$ facts, $\lfloor 1\frac{1}{2}n \rfloor + 3$ rules, and therefore $\lfloor 1\frac{1}{2}n \rfloor + 3$ arguments. Figure 9.2.4 shows, if $n = 5$, the corresponding dialectical tree.

9.2.2 Splitting the data sets

The only remaining issue is how to split the knowledge bases over the participants of the dialogue. Unfortunately, determining how to split the knowledge

[2]Each layer increases the number of arguments by 4 times the number of defenders added previously.

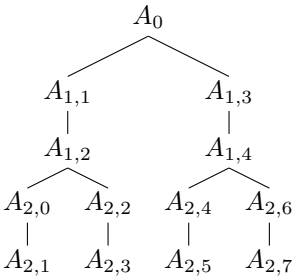


Figure 9.2.2: The dialectical tree for team defeat 2.

$$\langle\{p2, p2 \Rightarrow q\}, q\rangle \quad \langle\{\neg p2, \neg p2 \Rightarrow q\}, q\rangle \quad \langle\{p1, p1 \Rightarrow q\}, q\rangle \quad \langle\{\neg p1, \neg p1 \Rightarrow q\}, q\rangle$$
$$\langle\{\neg p2\}, \neg p2\rangle \qquad \langle\{p2\}, p2\rangle \qquad \langle\{\neg p1\}, \neg p1\rangle \qquad \langle\{p1\}, p1\rangle$$

Figure 9.2.3: The possible dialectical trees for the floating conclusions data set with $n = 2$. Assuming that the goal proposition is q. The arguments supporting q are the root arguments.

bases is not trivial. Especially since some modified strategies are not sound, the split may in part determine the outcome of the dialogue.

As mentioned at the beginning of the chapter, the data sets are used in two ways. Firstly the data sets are used to test the strategies in an 'intake setting'. The intake is one which tries to model the asymmetricity of the intake. Secondly, the data sets are distributed randomly over the agent's knowledge bases.

The intake setting is one in which the proponent knows all rules, and the opponent all the facts. This division is inspired by the intake scenario, in which the police officer is assumed to know the law (thus the rules), and the complainant the facts (or what happened).

The data sets are also used with a random distribution of knowledge. In this setting, each rule of fact exists with a 50% chance either in the proponent's or opponent's knowledge base.

That random setting is also tested differently. Instead of simulating 1 dia-

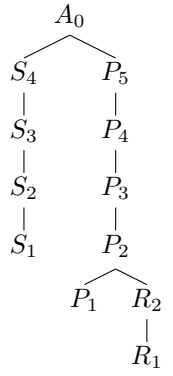


Figure 9.2.4: The exhaustive dialectical tree for the ambiguity data set for $n = 5$. A_0 is the root argument, supporting the proposition q.

logue per combination of a strategy, data set, and a certain size of that data set, 30 dialogues each with different knowledge bases are simulated.

9.2.3 The implementation

To run the strategies on different data sets, an implementation of Black and Hunter's dialogue system was developed.[3] It implements the argument inquiry and warrant inquiry dialogue systems, along with the exhaustive and all modified strategies.

First, is explained how the program can be used to simulate dialogues. Next, is discussed in more detail how the program is programmed.

Usage

The program as-is can be used to easily simulate an argument or warrant inquiry dialogue. Only the knowledge bases, topic, which dialogue system to start with, and the strategies the agents should use, have to be specified.

To specify the knowledge bases of the agents, three files are needed. The files `firstplayer.txt` and `secondplayer.txt` each respectively specify the knowledge bases of the proponent and opponent, and should contain propositions. The other file, `levels.txt`, specifies for each rule or fact in the union of the agents' knowledge bases the corresponding preference level. This file should therefore contain all beliefs (i.e. the propositions with their preference level). Due to a technical limitation, `firstplayer.txt` and `secondplayer.txt` cannot contain beliefs.

A fourth file (`initial.txt`) is used to specify the topic of the initial dialogue. The topic is specified as a defeasible fact. The type of the initial dialogue and the strategies that the agents should follow are specified when running the program.

Table 9.2.1 shows how the defeasible facts, rules, and beliefs are expected to appear in the aforementioned text files.

Note that defeasible facts are modelled as defeasible rules with an empty head. As seen in Table 9.2.1 for example, the defeasible fact a is modelled as `a -< TRUE.`, such that `TRUE` models the logical $\top$.

Because of this modelling, the knowledge bases, *not* `levels.txt`, must also contain "`TRUE.`". The logical $\top$ is namely modelled as a strict fact.

Formal representation	Expected input
Defeasible fact a	`a -< TRUE.`
Defeasible fact $\neg c$	`!c -< TRUE.`
Defeasible rule $a \Rightarrow b$	`b -< a.`
The belief $(a, 5)$	`a -< TRUE.; 5`
The belief $(a \Rightarrow b, 4)$	`b -< a.; 4`

Table 9.2.1: How the formal notation used in this paper maps to the input that is expected by the program.

Note that this syntax is a result of the way Tweety prints its arguments. There was unfortunately no time to change it.

[3] https://gitlab.com/Kumeling/agentdialogues/

As a result, the dialectical trees, and dialogues generated by the program also use the syntax that is shown in Table 9.2.1. Therefore, the dialogues and dialectical trees shown in this chapter look different than the representation as defined in the previous chapters.

Example dialogue

The argument inquiry dialogue shown in Table 5.9.1 on page 28 can therefore be simulated as follows.

The contents of `firstplayer.txt` should be as follows:

```
d -< c.
c -< b.
b -< a.
TRUE.
```

The contents of `secondplayer.txt` should be as follows:

```
a -< TRUE.
b -< TRUE.
TRUE.
```

`levels.txt` should contain the following:

```
d -< c.; 1
c -< b.; 1
b -< a.; 1
a -< TRUE.; 1
b -< TRUE.; 1
```

And, lastly, `initial.txt` should contain "`d -< TRUE.`".

These files should all be together in a single folder. In order to simulate an argument inquiry dialogue with both agents using the exhaustive strategy, the following command can then be used: `app diag -f exhaust -s exhaust black_ai <path_to_dataset_folder>` In order to simulate warrant inquiry dialogues, replace `black_ai` with `black_wi`. Using the command `app diag -h` will give more information about the available options.

Technical details

Here, it is discussed what libraries are used in the program to support the logic used. The high-level architecture is explained in the next section, as the program also makes it easy to write other dialogue systems.

Internally, the program uses the Tweety library (Thimm, 2017) for defeasible reasoning. More precisely, it uses its implementation of DeLP. That implementation, however, does not support defeasible facts, preference levels, and uses first order logic to represent its terms instead of propositional logic.

The implemented dialogue system works around the first two restrictions. The lack of support for defeasible facts is solved by modelling them as defeasible rules. Support for preference levels is implemented as a custom way of comparing arguments (via a custom `ComparisonCriterion`).

The use of first order logic is not a limitation, as it is a superset of propositional logic. The program is, however, slow on large data sets, in part due to the use of first order logic.

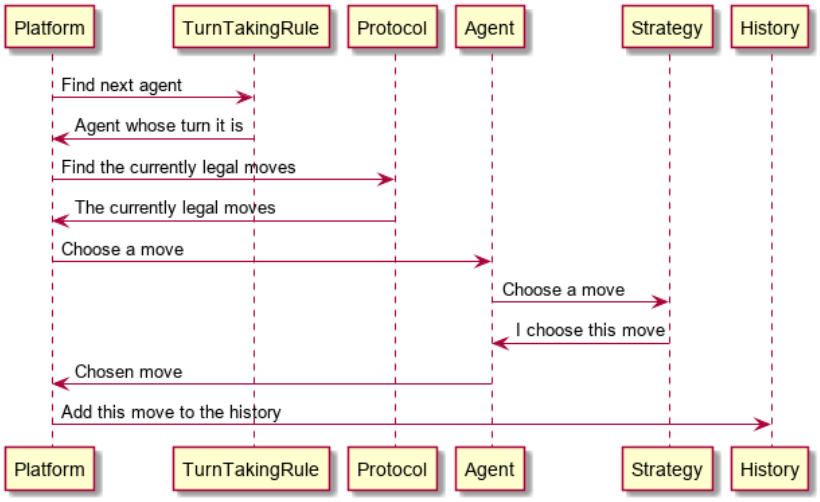


Figure 9.2.5: How the program determines for one turn of a dialogue what move is chosen. It shows from top to bottom all steps that are undertaken by the `Platform`.

High-level architecture

The driving force of the program is the `Platform`. The platform continuously seeks the agent who's turn it is, which moves are legal, which move the agent chooses to make and whether the dialogue should end. It communicates with a `TurnTakingRule` class which contains all participants, and which determines whose turn it is. Next, the `Protocol`, which contains the rules determining which moves are legal, will return which moves are legal for the agents who's turn it is. The `Strategy` of the agent whose turn it is, picks a move from this list. The picked move is added to the `History`, after which the effects of the move are applied. An *assert* move, for example, will add its argument to the commitment store of the agent that uttered it. After 'applying' the move, the cycle starts again. This continues until there are no legal moves, a strategy picked none of the legal moves, or if one of the dialogue system's termination rules are triggered. Figure 9.2.5 shows graphically how the `Platform` handles one turn of a dialogue.

Note that a dialogue system is here split into various parts. In the program the `Protocol`, `TurnTakingRule` and the way the history of the dialogue is recorded, determine the set of valid dialogues. The `Strategy` is then used to generate the dialogues. As already mentioned, the protocol contain rules on when certain moves are legal, and when the dialogue should end. The turn taking rule contains logic on how the turn order is determined.

Difficulties

Some difficulties were encountered in the development of the software. The main one being the dialectical tree implementation of Tweety. The implementation in Tweety has got a small flaw. When the `getDefeaters` method is called to compute the set of defeaters on a specific node in the dialectical tree, first all

of its already existing children are thrown away. It therefore does not provide a way to incrementally grow an existing dialectical tree.

Therefore, different dialectical trees, that are called `BHBetterDiacTree` and `BetterDiacTree`, were implemented. `BHBetterDiacTree` implements a dialectical tree that is constructed from the commitment stores. Moreover, the other tree, `BetterDiacTree`, implements the dialectical tree constructed from a dialogue as discussed in Section 7.1.2.

9.3 The setup

Two experiments are performed. In the first one, the modified strategies are per data set compared to the exhaustive strategy, in an 'intake' setting. The second experiment involves the same data sets and strategies, but with randomized knowledge bases. The goal of the second experiment is to check whether the behaviour of the strategies in the first experiment is a consequence of the intake setting.

Note that in both experiments, no argument inquiry dialogues are simulated. Instead, warrant inquiry dialogues are simulated. Argument inquiry dialogues are not tested for two reasons. Firstly, also testing argument inquiry dialogues would double the simulations required. Secondly, in case of the intake, an argument has to be checked on its validity, and not just found.

The size of the simulated dialogues and of the resulting dialectical trees are both measured. For the dialogues, the number of moves is measured. For the dialectical trees, the number of arguments is noted.

The first experiment is performed in what is called the 'intake setting'. Herein, the proponent knows all the rules, and the opponent all of the facts. Furthermore, per combination of a strategy, a data set, and a certain size of that data set one dialogue is simulated.

The second experiment is set in the 'random setting'. Here, knowledge is distributed such that each rule or fact is with a 50% chance in either the proponent's or opponent's knowledge base. Furthermore, per combination of a strategy, a data set, and a certain size of that data set thirty dialogues each with different, random, knowledge bases are simulated.

For each experiment, first the data sets whereupon the strategies are tested are generated. Then, the strategies are tested on those data sets. For a particular size of a given data set, all strategies are therefore tested on the same knowledge bases. For example, take ambiguity at $n = 1$. At $n = 1$, this data set contains the following rules and facts.

```
q -< s1.
!q -< p1.
!p0 -< r0.
s1 -< TRUE.
p1 -< TRUE.
r0 -< TRUE.
```

In the first experiment, these rules are divided such that the proponent knows the first three rules and the opponent the last three. Then, the modified strategies are all tested using that distribution of knowledge.

In the random setting (the second experiment), knowledge is divided randomly. For each size of a data set this is done thirty times. As such, for each size of each data set, thirty 'test cases' are generated. The modified strategies are then tested on all of these test cases. This ensures that the results of the modified strategies in the random setting are comparable with one another. Therefore, for ambiguity at $n = 1$, for example, there exist 30 'test cases' with different distributions of knowledge. The modified strategies are then tested on all of these thirty 'test cases'.

The experiments are also tested on smaller data sets than one might expect. The reason for this is that the software used is not particularly fast. For example, running the limited commitment strategy on the team defeat data set of size 5 would take around 5 hours. Tables 9.3.1 and 9.3.2 show up to which n dialogues were simulated.

	Team defeat	Floating conclusions	Ambiguity
Exhaustive strategy	4	35	20
Smart original	4	35	22
Smart dialogue	4	35	20
Limited commitment	4	35	25
Limited dialogue	4	35	25

Table 9.3.1: For the experiments in the **intake setting**, up to which size of the data set dialogues were simulated. If, for example, 35 is specified, then $n \in [1, 35]$.

	Team defeat	Floating conclusions	Ambiguity
Exhaustive strategy	3	15	12
Smart original	3	15	12
Smart dialogue	3	15	12
Limited commitment	3	15	12
Limited dialogue	3	15	12

Table 9.3.2: In the **random setting** up to which size of the data set dialogues were simulated. If, for example, 15 is specified, then $n \in [1, 15]$.

Therefore, 2 sets of experiments were performed, each with 3 data sets of varying sizes, and 5 strategies. In the first set 1 dialogue per combination of strategy, data set, and a particular size of that data set was simulated. In the second set of experiments, 30 dialogues were simulated per such a combination.

9.4 Results

This section will discuss the results of the experiments. It will do so by answering the research questions per data set.

The results are discussed per data set in an attempt to more clearly see their behaviours on a single type of dialectical tree. This is, however, not a realistic scenario. In the real world, knowledge bases are never clearly of one particular shape. As mentioned, however, such knowledge bases may also make it harder to clearly see how the strategies behave.

The modified strategies are compared to the exhaustive strategy as part of the experiments. It is therefore useful to be able to refer to the dialectical tree that strategy has generated. That dialectical tree will be called the *full tree*. The full tree is the same as the dialectical tree constructed from the union of the agents' beliefs (Black & Hunter, 2009). Likewise, the status of its root argument is the same as the root argument's status in the dialectical tree constructed from the union of the agents' beliefs. Further on, the exhaustive strategy's dialectical tree will therefore be called the full tree.

The data that is discussed, is included in the appendix. There, Appendix A (on page 87) shows per data set in the *intake setting* the size of the generated dialogues and dialectical trees. Appendix B (on page 93) shows the statuses of the root arguments, also in the *intake setting*. Lastly, Appendix C (on page 95) shows for the *random setting* the size of the generated dialogues and dialectical trees.

In the following, dialectical trees will be shown differently than in the rest of the thesis. This is because the dialectical trees shown are taken directly from the program. As a result, attack relations are shown using arrows, and the root argument is on the bottom. Furthermore, the dialectical trees will follow the syntax explained in Section 9.2.3. The defeasible fact $\neg a$ will therefore be represented as `!a -< TRUE.` and the defeasible rule $\neg a \Rightarrow b$ as `b -< !a.`. However, arguments are also represented differently. The argument $\langle\{\neg a, \neg a \Rightarrow b\}, b\rangle$ is represented as `<{b -< !a., !a -< TRUE.}, b>`.

For the same reasons as mentioned above, dialogues use the alternative syntax for the arguments. For example, the move $\langle P, ai, assert(\langle\{s1, s1 \Rightarrow q\}, q\rangle)\rangle$ will be shown as $\langle P, ai, assert($`<{s1 -< TRUE.,q -< s1.},q>`$)\rangle$. The change in syntax is inconvenient, but unfortunately a result of the way how Tweety prints arguments in DeLP.

9.4.1 Ambiguity handling

Here, the results of the experiments regarding the research questions posed in Section 9.1 for the ambiguity data set are discussed.

First, as part of the first research question, are the modified strategies within the intake setting compared to the exhaustive strategy. Secondly, it is investigated how much the intake setting influences those results.

Intake setting

Looking at the results for ambiguity handling in the intake setting, it is obvious (see Figure A.1.1 and A.1.4) that the modified strategies generally generate smaller dialectical trees, but not smaller dialogues. The reason why differs for the limited and for the smart strategies.

The smart strategies generate smaller trees but not smaller dialogues, even if the current dialectical tree allows them to be 'smart' (i.e. and there are no more arguments that change the status of the root argument). In such a case there will namely still be arguments that are asserted. This has got to do with the second condition of the smart strategies. The smart strategies require that arguments either change the status of the root argument, or do not modify the dialectical tree. Therefore, if there are no more arguments left to assert that would change the status of the root, all arguments that do not modify the

dialectical tree are still asserted. Look, for example, at $n = 4$. The full tree of that size is shown in Figure 9.4.1, while Figure 9.4.2 shows the tree generated by the smart strategies. The dialogue shown in Figure 9.4.4 is an excerpt from the dialogue generated by the smart dialogue strategy. In that dialogue, most of the *close* moves have been removed. The dialogue shows that even though the argument `<{p4-<TRUE., !q-<p4.}, !q>` was not asserted, the argument `<{p3 -< TRUE.,!p4 -< p3.},!p4>` *was* asserted.

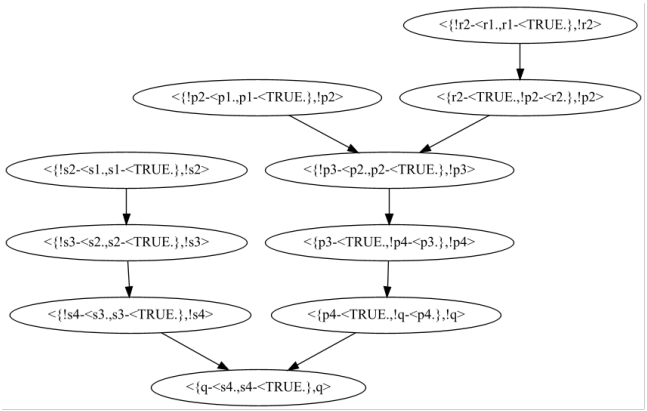


Figure 9.4.1: The full tree for ambiguity of size 4 in the intake setting.

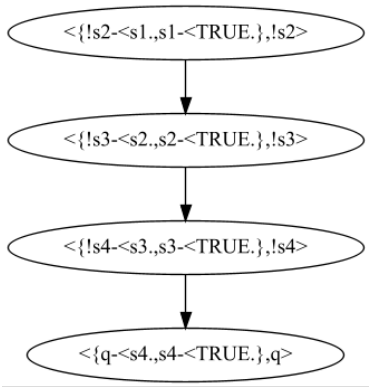


Figure 9.4.2: The dialectical tree generated by the smart strategies in the intake setting for ambiguity of size 4.

The limited strategies, on the other hand, show the same behaviour, even though an argument will not be asserted if there already exists an argument that supports the same proposition. Note, however, that ambiguity handling contains only one such instance of arguments supporting the same proposition. The limited strategies will therefore, when compared to the exhaustive strategy, assert only one move less. For example, continuing from the example in the previous paragraph for $n = 4$, Figure 9.4.3 shows the dialectical tree generated by both limited strategies. It shows that the argument `<{!p3 -< p2.,p2 -< TRUE.},!p3>` was never asserted, since there already existed an argument for

`!p3`. However, like with the smart strategy, the argument `<{!p2 -< p1.,p1 -< TRUE.},!p2>` *was* in fact asserted. Therefore, even though the dialectical tree was smaller through the extra condition placed upon asserts by the limited strategies, it did not prevent arguments that would never end up in the dialectical tree from being asserted.

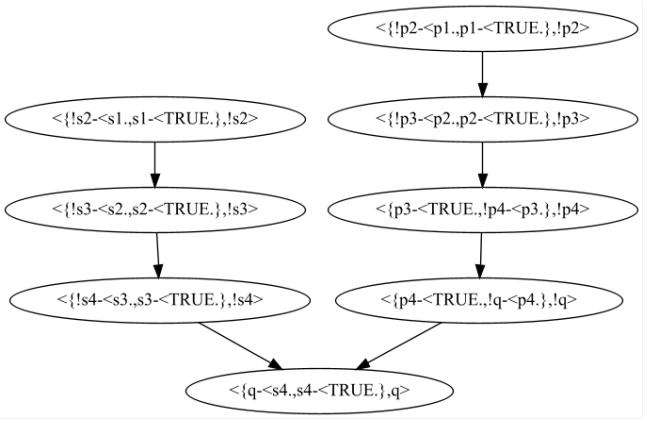


Figure 9.4.3: Dialectical tree generated by both limited strategies for ambiguity of size 4.

In short, when compared to the exhaustive strategy, all modified strategies generate smaller dialectical trees. However, the dialogues they produced were *not* significantly smaller. In fact, the biggest difference between dialogues generated by the exhaustive strategy and the modified strategy, was 2 moves. This difference is purely due to the single argument that is not asserted by the limited or smart strategy.

Influence of the intake setting

The influence of the intake setting differs for the smart and for the limited strategies. For the smart strategies the influence is far greater than on the limited strategies. However, this influence is, within ambiguity handling, only visible on the size of the dialectical trees. This section will show in what way the influence of the intake setting is visible on the smart and the limited strategies, when comparing the intake setting's results to the results from the experiments with random knowledge bases.

The smart strategies in the random setting, for example, show that the specific distribution of knowledge heavily influences the resulting size of the dialectical trees (see Section C.1.2 on page 97). This is shown by the huge differences in the maximum and minimum sizes of the generated dialectical trees. The intake setting, for example, causes the smart strategies to generate dialogues that have either very large or very small dialectical trees, which can be seen by comparing Figure A.1.1 with the plots in Section C.1.2. The reason being that it depends on the specific dialectical tree and the order of moves in the dialogue, whether the smart strategies end up in a situation wherein no argument is able to change the status of the root argument. At $n = 4$, for example, is the dialectical tree generated by the smart dialogue strategy quite

1. $\langle P, wi, open(wi, \texttt{q -< TRUE.})\rangle$
2. $\langle O, wi, close(\texttt{q})\rangle$
3. $\langle P, wi, open(ai, \texttt{q -< s4.})\rangle$
4. $\quad\langle O, ai, assert(\texttt{<\{s4 -< TRUE.\},s4>})\rangle$
5. $\quad\langle P, ai, assert(\texttt{<\{q -< s4.,s4 -< TRUE.\},q>})\rangle$
   ...
9. $\langle P, wi, open(ai, \texttt{!s4 -< s3.})\rangle$
10. $\quad\langle O, ai, assert(\texttt{<\{s3 -< TRUE.\},s3>})\rangle$
11. $\quad\langle P, ai, assert(\texttt{<\{!s4 -< s3.,s3 -< TRUE.\},!s4>})\rangle$
    ...
15. $\langle P, wi, open(ai, \texttt{!s3 -< s2.})\rangle$
16. $\quad\langle O, ai, assert(\texttt{<\{s2 -< TRUE.\},s2>})\rangle$
17. $\quad\langle P, ai, assert(\texttt{<\{!s3 -< s2.,s2 -< TRUE.\},!s3>})\rangle$
    ...
21. $\langle P, wi, open(ai, \texttt{!s2 -< s1.})\rangle$
22. $\quad\langle O, ai, assert(\texttt{<\{s1 -< TRUE.\},s1>})\rangle$
23. $\quad\langle P, ai, assert(\texttt{<\{!s2 -< s1.,s1 -< TRUE.\},!s2>})\rangle$
    ...
27. $\langle P, wi, open(ai, \texttt{!q -< p4.})\rangle$
28. $\quad\langle O, ai, assert(\texttt{<\{p4 -< TRUE.\},p4>})\rangle$
    ...
31. $\langle P, wi, open(ai, \texttt{!p4 -< p3.})\rangle$
32. $\quad\langle O, ai, assert(\texttt{<\{p3 -< TRUE.\},p3>})\rangle$
33. $\quad\langle P, ai, assert(\texttt{<\{p3 -< TRUE.,!p4 -< p3.\},!p4>})\rangle$
    ...
37. $\langle P, wi, open(ai, \texttt{!p3 -< p2.})\rangle$
38. $\quad\langle O, ai, assert(\texttt{<\{p2 -< TRUE.\},p2>})\rangle$
39. $\quad\langle P, ai, assert(\texttt{<\{!p3 -< p2.,p2 -< TRUE.\},!p3>})\rangle$
    ...
60. $\langle O, wi, close(\texttt{q})\rangle$
61. $\langle P, wi, close(\texttt{q})\rangle$

Figure 9.4.4: An excerpt from the dialogue generated by the smart dialogue strategy on ambiguity of size 4 in the intake setting. Most close moves, except the final two, have been omitted.

small (Figure 9.4.2). However, at $n = 5$ the generated dialectical tree is a lot bigger. As mentioned, whether this happens or not depends on the order of the moves in the dialogue. It is unclear why exactly this happens. Further analysis would be needed to find out why.

For the limited strategies, the same thing holds. It depends on the specific dialectical tree, and the order of the moves in the dialogue, whether the resulting dialectical tree is large, or small.

9.4.2 Team defeat

Here, the same things as with ambiguity are investigated, but only for the team defeat data set.

Note that the team defeat data set consists of n arguments that each are attacked by two arguments, and that those attackers each also have got one attacker. Figure 9.4.5 (from Section 9.2.1), shows an example for $n = 2$.

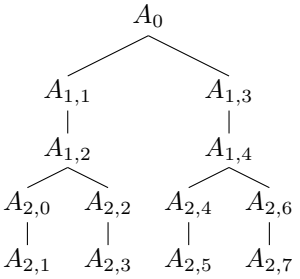


Figure 9.4.5: The dialectical tree for team defeat 2.

Intake setting

The modified strategies make in the intake setting little to no difference on the size of the dialogues (see Plot A.2.2). The biggest difference, that is seen on $n = 4$, is only 10 moves on a total of more than 300.

On the other hand, the limited strategies and the smart dialogue strategy do generate smaller dialectical trees, but the smart original strategy does not. Take, for example $n = 4$. Here, the dialectical tree generated by the smart original strategy is as large as the tree generated by the exhaustive strategy. However, the other modified strategies have generated smaller dialectical trees. Once again, the reason why this is the case differs for the limited strategies, and the smart dialogue strategy.

The limited strategies are quite interesting, since the trees generated by them grow linearly instead of exponentially. This has got to do with the structure of team defeat. Within team defeat, each argument has got up to 2 attackers. Those two attackers also always attack the same premise. Therefore, since the limited strategies only assert one argument per proposition, there will be in the dialectical trees generated by them at most one attacker per argument. The resulting tree always only contains one series of arguments. This series of arguments is also n arguments long. It is therefore no surprise that the dialectical trees generated by the limited strategies grow linearly.

The smart dialogue strategy, however, also generates small dialectical trees, even though it *is* able to assert multiple arguments for the same proposition. Furthermore, the dialectical tree that it generates at $n = 4$ (see Figure A.2.1 on page 90) is even smaller than the dialectical tree generated by both limited strategies. To see why the generated dialectical tree at $n = 4$ is so small, we compare the generated tree to full tree. The difference between those two dialectical trees is quite big. The tree generated by the smart dialogue strategy has got 4 arguments, while the full tree contains 61. The tree generated by the smart dialogue strategy is shown in Figure 9.4.7. The figure seems to show that the smart dialogue strategy ended up in a situation where no other argument could change the status of the root argument. That is, however, not the case. There *does* exist such an argument, namely `<{!p4 -< p5., p5 -< TRUE.}, !p4>`, which attacks the argument `<{p4 -< TRUE., !p0 -< p4.}, !p0>`. Why that argument was not included in the dialectical tree has got to do with why the dialogue generated by the smart dialogue strategy is not much smaller (see Figure A.2.2, page 90).

The dialogues generated by the smart dialogue strategy are, despite smaller trees, not much smaller for the same reason as that the dialogues within the ambiguity data set are not much smaller. Within the ambiguity data set, the reason was that there were, despite the smart dialogue strategy, still a lot of irrelevant arguments asserted. That is also the case here, since the smart dialogue strategy does not forbid one to assert arguments that do not end up in the tree. Note that this property is in fact desirable. It is needed in order to allow the agents to build arguments. However, this 'loophole' is also used by agents to assert irrelevant arguments, i.e. arguments that do not change the status of the root argument. Take, for example, an argument that is not being asserted because it does not change the status of the root. Attackers of that argument will still be asserted, since they do not change the dialectical tree. Those arguments are therefore not relevant, but still inserted in the dialogue. The excerpt shown in Figure 9.4.6 shows a couple of those arguments being asserted as the 33rd, 303rd and the 353rd move in the dialogue. Those arguments were not relevant (i.e. did not change the status of the root argument, see Figure 9.4.7), but were still asserted.

On the other hand, continuing the example of the smart dialogue strategy on $n = 4$, the argument `<{!p4 -< p5., p5 -< TRUE. }, !p4>` would change the root, was asserted, but is not present in the resulting dialectical graph (Figure 9.4.7). The reason being the same as to why the smart dialogue strategy is not sound on warrant inquiry dialogues. That reason was that the type of dialectical tree used has got a dependency on the order of moves in the dialogue. The argument `<{!p4 -< p5., p5 -< TRUE. }, !p4>` is, therefore, not included in the dialectical tree, since it was asserted *before* `<{p4 -< TRUE.,!p0 -< p4.},!p0>`.

Influence of the intake setting

From the results of the experiments in the random setting (Section C.2.1), it is immediately clear that the intake setting has little influence on the size of the dialogues, even when the corresponding dialectical trees (Section C.2.2) are small.

Furthermore, it is clear that the assertion of irrelevant arguments, as ex-

1. $\langle P, wi, open(wi, \texttt{p0 -< TRUE.})\rangle$
2. $\langle O, wi, assert(\texttt{<p0 -< TRUE.,p0>})\rangle$

...

5. $\langle P, ai, assert(\texttt{<\{!p0 -< p2.,p2 -< TRUE.\},!p0>})\rangle$

...

7. $\langle P, ai, open(ai, \texttt{!p0 -< p4.})\rangle$

...

15. $\langle P, ai, assert(\texttt{<\{!p4 -< p5.,p5 -< TRUE.\},!p4>})\rangle$

...

21. $\langle P, ai, assert(\texttt{<\{p12 -< TRUE.,!p5 -< p12.\},!p5>})\rangle$

...

25. $\langle P, ai, assert(\texttt{<\{p10 -< TRUE.,!p5 -< p10.\},!p5>})\rangle$

...

33. $\langle P, ai, assert(\texttt{<\{p11 -< TRUE.,!p10 -< p11.\},!p10>})\rangle$

...

43. $\langle P, wi, assert(\texttt{<\{p4 -< TRUE.,!p0 -< p4.\},!p0>})\rangle$

...

305. $\langle P, ai, assert(\texttt{<\{p23 -< TRUE.,!p22 -< p23.\},!p22>})\rangle$

...

353. $\langle P, ai, assert(\texttt{<\{!p50 -< p51.,p51 -< TRUE.\},!p50>})\rangle$

...

362. $\langle O, wi, close(\texttt{p0})\rangle$
363. $\langle P, wi, close(\texttt{p0})\rangle$

Figure 9.4.6: An excerpt from the dialogue generated with team defeat 4 in the intake setting using the smart dialogue strategy. It shows that key arguments are asserted in an order that causes the associated dialectical tree to not include certain arguments.

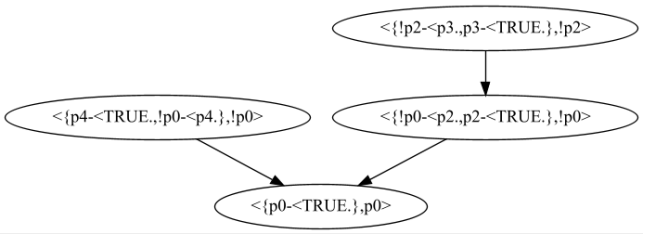


Figure 9.4.7: The dialectical tree generated using the smart dialogue strategy on the team defeat data set using an n of 4.

plained in the previous section, can only be fixed by changing the strategy used. After all, the specific distribution of knowledge has got, with team defeat, little to no influence on the final size of a dialogue.

When looking at the size of the generated dialectical trees within the random setting, a mostly similar picture as within the intake setting is painted. Firstly, the limited strategies' dialectical trees grow without exception linearly (Section C.2.2). Secondly, the smart dialogue strategy's generated dialectical trees within the intake setting (Figure A.2.1) are shown to mostly follow the same pattern as the dialectical trees within the random setting.

The same is true for the smart original strategy, except for an outlier in the random setting at $n = 3$.

The influence of the intake setting is therefore minimal in the team defeat data set.

9.4.3 Floating conclusions

Lastly, the research questions are investigated for the floating conclusions data set.

Intake setting

All generated dialectical trees are 2 arguments long. This is not a consequence of either the intake setting or any of the strategies. It is a consequence of the floating conclusions data set. As explained in Section 9.2.1, the floating conclusions data set contains $2n$ arguments for q, and $2n$ arguments that each attack one of the arguments that support q. Therefore, the maximum number of arguments within a single dialectical tree is, within the floating conclusions data set, 2 arguments.

There is, within the intake setting, a visible difference in the size of the dialogues, when using the limited strategies and comparing them to the exhaustive strategy. The smart strategies, however, do not generate smaller dialogues.

The smart strategies do not generate smaller dialogues, probably because there are no opportunities within this dataset for those strategies to be 'smart'. In other words, there is never a time in a dialogue wherein arguments cannot be asserted by the smart strategies. Take, for example, floating conclusions on $n = 2$, shown in Figure 9.4.8. Using that data set, the smart strategies will always act like the exhaustive strategy, therefore always asserting all arguments. Remember that using the smart strategies arguments cannot be asserted if they do not change the status of the root argument (assuming they would change the tree when asserted). However, during a dialogue with floating conclusions, no argument will ever fulfil that condition, and be un-assertable. Instead, every argument will at some point in the dialogue either change the status of the root, be an argument that supports q (iff there is no root argument), or not change the dialectical tree. Therefore, the smart strategies will always assert every argument in a dialogue. For example, when $n = 2$ and there is not yet a root argument, then there are four arguments that can be asserted. When one of those arguments is chosen as the root argument, then the other three arguments no longer change the tree, and can be asserted. The same holds for the attackers of those arguments.

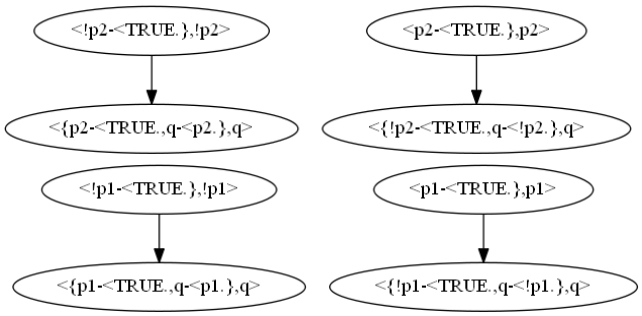


Figure 9.4.8: The four possible dialectical trees for the floating conclusions data set with $n = 2$. Assuming that the goal proposition is q. The arguments supporting q are the root arguments.

The limited strategies, on the other hand, do generate dialogues that are smaller. Those dialogues are smaller because the limited strategies do not assert a lot of arguments. For example, the floating conclusions data set contains $2n$ arguments supporting q. Of these, only one is asserted with the limited strategy. The rest ($2n-1$ arguments) will never be asserted, since there already was an argument asserted for q. This does not hold for the attackers of the arguments supporting q. Those attackers attack the arguments on their premise, not on q. Moreover, each argument supporting q has got different premises (see Figure 9.4.8 for $n = 2$). Therefore, all of those attackers are still asserted. Since the limited strategies therefore assert fewer arguments, the generated dialogues are also smaller.

However, the dialogues generated by the limited strategies are not that much smaller, even though roughly half of the arguments are not asserted. Within the intake setting, the reason for this is the number of argument inquiry sub-dialogues that are still opened, explored until an argument is nearly discovered, and then closed. An example is given in Figure 9.4.9. In that dialogue, on the eleventh move, an argument inquiry sub-dialogue is opened to search for an argument for the rule `q -< !p1.` (i.e. $\neg p1 \Rightarrow q$). In the next move, `!p1` is asserted. Next, the sub-dialogue is closed, and the argument `<{!p1 -< TRUE., q -< !p1.}, !p1>` was not asserted. That argument was not asserted because earlier in the dialogue another argument supporting q had already been asserted. However, there was still an argument inquiry sub-dialogue opened for it, resulting in four 'useless' moves.

The influence of the intake setting

As expected, the random setting shows that the intake setting has got, within floating conclusions, no influence on the size of the generated dialectical trees. This is no surprise, since, as explained earlier, the maximum size of a dialectical tree is 2 arguments long within this data set.

There is, however, a huge difference between the size of dialogues generated in the random setting, and those in the intake setting. Regardless of the strategy, in all dialogues in the random setting for $n \geq 6$, no dialogue is longer than six moves. For $1 \leq n \leq 5$, the maxima of the tests in the random setting are

1. $\langle P, wi, open(wi, \texttt{q -< TRUE.})\rangle$
2. $\langle O, wi, close(\texttt{q})\rangle$
3. $\langle P, wi, open(ai, \texttt{q -< p2.})\rangle$
4. $\quad\langle O, ai, assert(\texttt{<\{p2 -< TRUE.\},p2>})\rangle$
5. $\quad\langle P, ai, assert(\texttt{<\{q -< p2.,p2 -< TRUE.\},q>})\rangle$
6. $\quad\langle O, ai, close(\texttt{q, p2})\rangle$
7. $\quad\langle P, ai, open(ai, \texttt{q -< p1.})\rangle$
8. $\quad\quad\langle O, ai, assert(\texttt{<\{p1 -< TRUE.\},p1>})\rangle$
9. $\quad\quad\langle P, ai, open(ai, \texttt{q -< !p2.})\rangle$
10. $\quad\quad\quad\langle O, ai, assert(\texttt{<\{!p2 -< TRUE.\},!p2>})\rangle$
11. $\quad\quad\quad\langle P, ai, open(ai, \texttt{q -< !p1.})\rangle$
12. $\quad\quad\quad\quad\langle O, ai, assert(\texttt{<\{!p1 -< TRUE.\},!p1>})\rangle$
13. $\quad\quad\quad\quad\langle P, ai, close(\texttt{q, !p1})\rangle$
14. $\quad\quad\quad\quad\langle O, ai, close(\texttt{q, !p1})\rangle$
15. $\quad\quad\quad\langle P, ai, close(\texttt{!p2, q})\rangle$
16. $\quad\quad\quad\langle O, ai, close(\texttt{!p2, q})\rangle$
17. $\quad\quad\langle P, ai, close(\texttt{p1, q})\rangle$
18. $\quad\quad\langle O, ai, close(\texttt{p1, q})\rangle$
19. $\quad\langle P, ai, close(\texttt{q, p2})\rangle$
20. $\quad\langle O, ai, close(\texttt{q, p2})\rangle$
21. $\langle P, wi, close(\texttt{q})\rangle$
22. $\langle O, wi, close(\texttt{q})\rangle$

Figure 9.4.9: The dialogue generated in the intake setting, on floating conclusion with $n = 2$, by the limited dialogue strategy.

1. $\langle P, wi, open(wi, \texttt{q -< TRUE.})\rangle$
2. $\langle O, wi, assert(\texttt{<\{q -< !p3.,!p3 -< TRUE.\},q>})\rangle$
3. $\langle P, wi, assert(\texttt{<\{p3 -< TRUE.\},p3>})\rangle$
4. $\langle O, wi, close(\texttt{q})\rangle$
5. $\langle P, wi, close(\texttt{q})\rangle$

Figure 9.4.10: An example of a dialogue generated with floating conclusions for $n \geq 6$ within the random setting.

the same as the dialogues generated in the intake setting by the corresponding strategies.

Figure 9.4.10 shows the kind of dialogue encountered for $n \geq 6$. The difference with the dialogues in the intake dialogue, is that here no argument inquiry sub-dialogues are opened. Since this does not happen, there are, besides the two arguments shown, no other arguments asserted. Those other arguments (other supporters of q, or attackers of those arguments), will only be asserted in a sub-dialogue, since in warrant inquiry an argument needs to change the dialectical tree in order to be asserted. There are no argument inquiry sub-dialogues opened, because the proponent knew both a valid root argument and the attacker of that argument. However, within the intake setting, he only knows the rules, and is he forced to open a sub-dialogue. Therefore, the dialogue is short, because the proponent knew both a root argument and an attacker.

To make sure that a dialogue generated with the floating conclusion data set is 'long' (as was achieved within the random setting for $1 \leq n \leq 5$), therefore a specific distribution of knowledge is needed. The dialogue will namely only be cut short if one of the participants knows both a valid root argument, and an attacker. Long dialogues were, therefore, not achieved for $n \geq 6$ because the greater the data set, the greater the chance that a participant knows a valid root argument from the start of the dialogue.

The influence of the intake setting on the modified strategies in the floating conclusions data set is therefore a negative one. The specific distribution of knowledge that is assumed with the intake setting causes all strategies to generate long dialogues, which is not the goal.

9.5 Discussion of experiments

In this section the results of the simulations are discussed. First, the general setup of the experiments is explained and discussed. Secondly, the results for the first research question are discussed. Thirdly, the influence of the intake setting (the second research question) on the results of the experiments is discussed. Next, a general overview of the observed behaviour of the modified strategies is given. Lastly, the potential limitations of the experiments are discussed.

9.5.1 General setup

Here, the general setup of the experiments will be discussed. In both settings, the strategies were tested in isolation, i.e. on a single data set. First will be discussed whether this setup is realistic. Secondly, it will be discussed whether

the three data sets that were used are 'enough', i.e. whether the selection was not too limited.

It is not really realistic to test strategies in isolation. Real-world knowledge bases contain mixtures of different types of dialectical trees. The 'real' practical effect of the modified strategies is thus still unclear. However, by testing the strategies separately on different data sets, their behaviour can be observed more clearly.

The number of data sets used (three) is, moreover, also limited. The question therefore arises whether three data sets are sufficient, especially since the strategies were tested in isolation. The number of data sets is indeed quite limiting. Nevertheless, interesting results were obtained using 'just' those three data sets. Note the 'just'. Besides the three data sets, simulations were also performed with five strategies, the intake setting, the random setting, and various sizes of the three data sets. The total number of simulations was very high despite the total number of data sets. In addition, the results obtained were quite clear. Therefore, the number of data sets were, despite being limited, sufficient.

9.5.2 Effects of the modified strategies

The first research question was whether the modified strategies make a difference in the size of the generated dialogues and dialectical trees. Or, in other words, whether they reduce or completely eliminate irrelevant arguments. This section will, firstly, discuss that the modified strategies only reduce the number of irrelevant arguments, and not eliminate them. Secondly, it is discussed why the modified strategies are not particularly effective. Thirdly, it will be discussed that unnecessary sub-dialogues are also a reason for irrelevancy in warrant inquiry dialogues.

Lack of smaller dialogues

After the experiments in the intake setting, a common theme was found when investigating the effect of the modified strategies. Often, one of the modified strategies would generate a (relatively) small dialectical tree, but not a smaller dialogue. For example, the smart dialogue strategy in the ambiguity data set, in the intake setting, generated smaller dialectical trees, but not significantly smaller dialogues.

It was discovered that the main reason is that none of the modified strategies completely block irrelevant arguments (i.e. that do not contribute to the goal of the dialogue). These arguments do not cause the dialectical tree to grow, but do contribute to the size of the dialogue. The reason why the modified strategies still assert those arguments, differs per strategy.

The limited strategies

The limited strategies assume that irrelevant arguments are arguments that support a proposition that already has an argument supporting it. Therefore, they limit the number of arguments supporting the same proposition to one. This means that arguments are only asserted if they support an unsupported

proposition. This extra condition is, however, not enough to completely prevent irrelevant arguments from being asserted.

For example, while the generated dialectical trees within the team defeat data set were much smaller, the size of the dialogues was not (See Plot A.2.1 and A.2.2, and Section 9.4.1). The generated dialogues were, however, a little smaller. Therefore, the limited strategies did not prevent all irrelevant arguments from being asserted.

The smart strategies

The smart strategies, on the other hand, assume that only arguments that change the status of the root argument are relevant. Therefore, arguments have to either change the root argument's status or not change the tree before being asserted. However, the second condition is a 'loophole'.

The 'loophole' of the smart strategies is their second condition, namely that arguments that do not modify the dialectical tree may always be asserted. This condition is necessary to allow the agents to build arguments. However, as explained in Section 9.4.1, this condition also allows agents to assert most of the irrelevant arguments. Those arguments do not contribute to the size of the dialectical tree, but do contribute to the size of the dialogue.

Irrelevant sub-dialogues

The floating conclusions data set, on the other hand, revealed that irrelevant arguments are not the entire reason for the lack of smaller dialogues. Instead, the number of sub-dialogues also plays a role. For example, within the floating conclusions data set for $n \geq 6$ the main difference between dialogues generated in the random setting and the intake setting was that in the random setting no sub-dialogues were opened. In the intake setting often a sub-dialogue was opened, one argument (a premise of another argument) was asserted, and the sub-dialogue was closed. Such a sub-dialogue contributes nothing to the goal of the dialogue and is therefore generally useless.

In addition, it is possible that some irrelevant arguments that are still asserted, are tied to the irrelevant sub-dialogues that are opened. After all, most arguments are, before being asserted, discovered in an argument inquiry sub-dialogue. Reducing the number of irrelevant sub-dialogues would in that case also reduce the number of irrelevant arguments that are asserted.

Unfortunately, the modified strategies did not place extra conditions on the opening of sub-dialogues. It is therefore unknown if it is even doable to place such conditions cleverly. After all, assuming distributed knowledge, there is no way for the agents to know beforehand whether a sub-dialogue is going to be relevant. They simply do not know which arguments will be discovered in such a sub-dialogue.

9.5.3 Influence of the intake setting

The second research question will be discussed in this section. The question was in what way the intake setting influences the behaviours of the modified strategies. The first pair of simulations were done in the intake setting, wherein the proponent knew all the rules, and the opponent all the facts. The question

therefore arose, what is the effect of the intake setting? The second set of experiments were performed for this reason. By also simulating the modified strategies with randomized knowledge bases, the effect of the intake setting was investigated.

The precise influence of the intake setting differed per data set. Within the team defeat data set, the influence was limited. As mentioned in Section 9.4.2, the intake setting did not influence the behaviour of the modified strategies.

The ambiguity data set saw more influence. Especially the smart strategies' behaviour was heavily influenced. Moreover, the experiments in the random setting showed that, depending on the distribution of knowledge, the smart strategies generated small or large dialogues/dialectical trees.

Unsurprisingly, the floating conclusion data set saw the most influence of the intake setting. As mentioned in the results for the floating conclusions data set, the only time small dialogues were generated, was when one of the agents knew both a valid root argument, and its attacker. In the intake setting, no agent initially knows any argument, as the rules and facts are split over different agents. The dialogues are therefore large and grow linearly, instead of staying the same size as is the case for every strategy for $n \geq 6$ in the random setting.

9.5.4 Behaviour of the modified strategies

The limited strategies were generally observed to generate smaller dialectical trees and dialogues. This is no surprise since the limited strategies assert per proposition only one argument in support of it. In other words, they were therefore thorough in reducing the size of the dialectical trees. Whether the generated dialogues were also more relevant is, however, a more complicated question. One can say that they were more relevant. After all, in the generated dialogues there were fewer arguments asserted, and thus probably also fewer irrelevant arguments. On the other hand, neither one of the limited strategies was sound or complete. It is therefore unclear which of the smaller dialogues actually reached a 'correct' outcome (as defined by Black and Hunter (2009)). In other words, the limited strategies may have been too thorough.

Of the smart strategies, the smart original strategy (using the dialectical tree as defined by Black and Hunter (2009)) did indeed generate smaller, more relevant, dialogues. The catch is that the dialogues that are generated, were not that much smaller. Moreover, in most cases the smart original strategy generated the same dialogues as the exhaustive strategy. Its prime advantage is, however, that it is sound and complete. The smart dialogue strategy (using a dialectical tree based on the dialogue), on the other hand, is not sound and complete. The dialogues generated using it were, consequently, smaller than the one generated by the smart original strategy.

In general, the modified strategies did assert less irrelevant arguments. However, the limited strategies often asserted 'too few' arguments (as they are not sound or complete), and the smart strategies were only a marginal improvement over the exhaustive strategy. The general problem is that the strategies still not seem 'smart' enough. The limited strategies, for example, flat out refuse to assert duplicate arguments, while a duplicate argument may in fact be better than the one that was already asserted. Furthermore, the smart dialogue strategy is too dependent on the order of the moves in the dialogue, while the smart original strategy still asserts too many irrelevant arguments.

9.5.5 Limitations

Here, some limitations of the experiments are discussed. In general, two questions arise. Firstly, is the software that was used to simulate the dialogues good enough? Secondly, is the notion of relevancy represented well? In the following, first two of the main limiting factors of the software used are discussed. Secondly, the notion of relevancy that is used is discussed.

The software

The performance of the program was quite low, partly due to the internal use of predicate logic. For example, the ambiguity data set in the intake setting with $n = 22$ took around 2 hours to run using the smart original strategy. Therefore, fewer simulations than desired were run. Especially in the random setting, there were simulations performed up to smaller n than in the intake setting. Due to the internal use of Tweety's (Thimm, 2017) DeLP implementation, which uses predicate logic, the simulations spend a lot of time deriving all possible arguments from a knowledge base. Converting the implementation of DeLP to use propositional logic instead may alleviate this problem a bit. This was not attempted due to time constraints.

Secondly, the used software also does not support generating multiple dialogues given a single knowledge base. Most times, this is not a problem. The exhaustive strategy, for example, is specified to choose the first assert move from the list of all legal assert moves. The ordering of this list is unspecified, which is no problem for the exhaustive strategy, as the strategy is sound and complete. However, the limited commitment strategy is not sound and complete for warrant inquiry dialogues. Therefore, the ordering of the list of legal moves *does* matter when using that strategy. For example, if in the union of the agents' beliefs two arguments supporting q exist, then it matters which argument is asserted first. This means that in that case at least two different dialogues can be generated, depending on which argument is asserted first. The first dialogue may have a defeated root argument, while the second has an undefeated root argument.

Idea of relevancy

The modified strategies were created to generate 'more relevant' dialogues, by generating smaller dialectical trees and dialogues. However, is this a good implementation of the notion of relevancy? As mentioned in Section 4.4, Prakken defines strong relevance in persuasion dialogues such that: "*An attacking move in a dialogue d is [strongly] relevant iff it changes the dialogical status of d's initial move [i.e. root argument].*" (Prakken, 2005, Definition 6.1).

Note that this definition in principle only holds for persuasion dialogues, in which there exists a conflict between the two agents. In inquiry dialogues, no such conflict exist. Instead, both agents want to find out the acceptability of an argument. However, with this goal strong relevance is still applicable. After all, when trying to find out whether an argument is acceptable, only arguments that change the status of the root argument are relevant.

Note that the smart strategies require arguments to either not change the dialectical tree, or be strongly relevant. This thesis therefore tested whether the notion of strong relevancy is effective in inquiry dialogues. It seems that strong

relevance is somewhat effective in reducing the size of dialectical trees. However, it does not reduce the size of the dialogues significantly (See Section 9.5.4). After all, the smart original strategy did not generate significantly smaller dialogues. It took another notion of a dialectical tree (used in the smart dialogue strategy) to get significantly smaller dialogues.

The limited strategies, on the other hand, were not an implementation of strong or weak relevancy. Instead, they require that there are no duplicate arguments for the same proposition. The experiments show that the notion of relevancy that they do implement, does cause them to generate smaller dialogues and dialectical trees. Whether these dialogues are more relevant, however, is still up for debate, since none of the limited strategies are sound or complete.

Chapter 10

Discussion

The aim of this chapter is to discuss the findings that were presented in this thesis. Furthermore, this chapter also aims to discuss the chosen dialogue system and logic are a good model of an intake. Firstly, the findings of this thesis are discussed. Afterwards, it is discussed whether the logic and dialogue system used were a good choice for running the experiments on.

10.1 Discussion of results

In Chapter 8 it was investigated whether the modified strategies are sound and/or complete. Section 9.5 discussed whether the modified strategies generated dialogues that improved upon dialogues generated by the exhaustive strategy. Moreover, it was investigated in what way the intake setting influenced the behaviour of the strategies. Therefore, firstly, the observed behaviour of the strategies (i.e. a summary of the results) will be discussed. Then, some improvements for the modified strategies are suggested. Lastly, it is discussed whether the notions of soundness and completeness need refinement when dealing with the intake setting.

Summary of results

As mentioned in Section 9.5, only the smart dialogue and the limited strategies were able to generate smaller dialectical trees. The smart original strategy, on the other hand, only generated smaller dialectical trees in a handful of instances. The smaller dialectical trees did unfortunately not always result in smaller dialogues.

The influence of the intake setting on the modified strategies was also investigated. In general, the influence of the intake setting depends on the data set. Within the team data set, the intake setting had no influence on the modified strategies. The ambiguity data set showed that the intake setting's influence made the modified strategies generate 'extreme' dialogues. This means that the generated dialectical trees were either the largest or the smallest of the possible dialectical trees that could have been generated. Compare for example Figure A.1.1 with one of the plots in Section C.1.2. It is easily seen that the dialectical trees that are generated in the intake setting are either the largest or the smallest. Within the floating conclusions data set, the influence of the

intake setting is even greater. As explained in Section 9.4.3, the intake setting causes the modified strategies to generate dialogues that grow linearly. The reason for this is that the intake setting is a distribution of knowledge wherein the rules and facts are strictly separated.

Two reasons were identified for the lack of smaller dialogues. Firstly, there were a significant number of 'irrelevant' sub-dialogues observed. Secondly, no modified strategy completely blocked irrelevant arguments from being asserted. The smart strategies still allowed some of those arguments to be asserted, since irrelevant arguments often do not change the dialectical tree. Likewise, the limited strategies only blocked duplicate arguments for the same proposition, and not all irrelevant arguments.

Those 'irrelevant' sub-dialogues are sub-dialogues in which no relevant arguments are asserted. For example, Figure 9.4.9 on page 71 shows such a sub-dialogue at move 11 till 14. Such an 'irrelevant' sub-dialogue does not add arguments to the dialectical tree, but it does contribute to the size of the dialogue. Note that the modified strategies do not place extra conditions on the opening of sub-dialogues. They therefore do not reduce the number of irrelevant sub-dialogues.

The modified strategies, on the other hand, are designed to limit the number of irrelevant arguments that are asserted. Nevertheless, as mentioned in Section 9.5, they still allow some irrelevant arguments to be asserted. The modified strategies are therefore either still not smart enough, or use the wrong approach.

Improving the strategies

Here, three suggestions for improving the modified strategies are given. Firstly, different notions of soundness or completeness may be necessary. Secondly, using different notions of completeness, the number of sub-dialogues may be limited. Lastly, maybe the modified strategies need to be allowed to reason using extra data.

Chapter 8 proved which of the modified strategies were sound and complete. Except the smart original strategy, however, none of them were for warrant inquiry dialogues. Maybe the modified strategies therefore need to be judged on different criteria (i.e. using different notions of soundness/completeness). After all, besides the smart original strategy, no modified strategy was sound and complete (using Black and Hunter (2009)'s definitions). Even though the smart original strategy is sound/complete, it is the only modified strategy that was not a significant improvement over the exhaustive strategy. The results therefore show that it is hard to design sound and complete strategies that generate more relevant dialogues. Strategies that completely eliminate irrelevancy may thus need different notions of completeness and soundness.

Different notions of soundness and completeness may include omitting the requirement for a strategy to be complete. For example, while having a sound strategy (such that its dialogues produce no false answers) might be important, that strategy might not have to be complete (being able to find all acceptable arguments). After all, all intakes should be correct, but there does not necessarily have to be a report of every crime ever committed. Therefore, a strategy could be acceptable as long as it is sound.

Dropping the requirement that strategies need to be complete, opens the door to limiting the number of sub-dialogues. As explained in Section 9.5,

empty sub-dialogues are one reason for the fact that the modified strategies did not generate smaller dialogues. It is therefore worthwhile to design such a strategy. However, simply omitting sub-dialogues makes strategies automatically *not* complete. This is for the reason that agents do not know before opening a sub-dialogue whether a relevant argument is going to be found or not. The intake setting may provide a solution in that case. In the intake setting, the proponent is assumed to know all the rules. Assuming that this means that the proponent also knows the structure of the full dialectical tree, he can use this information by not opening sub-dialogues that would have found irrelevant arguments.

Likewise, one could require strategies to be sound and complete in *most* dialogues they generate. This means that there exist a certain chance that the generated dialogue is either not sound or complete. The goal would then be to create strategies where that chance is as low as possible.

On the other hand, maybe no different notion of soundness or completeness is needed, but instead the strategies need more data. For example, an agent could be allowed to reason using extra information. This is done in order to make a strategy that is noticeably 'better' in terms of dialogue and dialectical tree size. This extra information may be in the form of a user model (Section 4.3).

10.2 Applicability of DeLP and Black and Hunter

The results discussed above were obtained by simulating dialogues using a variant of DeLP, warrant inquiry dialogues and argument inquiry sub-dialogues. It was assumed that this combination is a good (enough) model of an intake. The question remains whether this is actually the case. This section will discuss whether the used logic and dialogue systems are a good model of an intake. This encompasses two things. Firstly, this encompasses the question whether the dialogue systems provides a good model of the intake. Secondly, this encompasses the question whether the chosen logic is strong enough to reflect everything that is typically said in an intake.

Warrant inquiry dialogues were designed with the medical domain in mind and not the intake. Therefore, the goal was to generate dialogues which only generate correct answers (soundness), and only generate answers if there are any (completeness). These goals are similar to the ones the police have with intakes. In an intake, a police officer first of all tries to ascertain that the crime being reported is a criminal offence, and not a civil matter. Secondly, he tries to find out what exactly happened. Both these things need to be done as accurately as possible. Otherwise, the police have an incorrect report of a crime. In other words, the police's goal with an intake is to only generate correct reports. Likewise, the police only want reports for crimes that actually took place (i.e. not erroneously create false reports). The police's goals with the intake are therefore comparable to Black and Hunter's medical domain.

However, as mentioned in Section 3.3, intakes probably encompass more than just inquiry dialogues. They also contain elements of both information-seeking and persuasion dialogues. Argument inquiry dialogues are partly used for information-seeking, although the information-seeking happens cooperatively. Warrant inquiry dialogues do contain elements of persuasion. After all, agents use counter-arguments in order to cooperatively find out the acceptability of an argument. Neither dialogue systems are, however, capable of dealing with

arguments that are off topic. As such, an angry or uncooperative complainant cannot be simulated.[1] Instead, it is assumed that the complainant is willing to cooperate. Neither warrant nor argument inquiry dialogues are therefore a complete model of intakes, although they probably come very close.

The used logic (a propositional, defeasible variant of DeLP) was chosen by Black and Hunter, since in the medical domain all knowledge is defeasible (Black & Hunter, 2009). Is it therefore still capable of encoding the information usually discussed in an intake? First it is discussed whether the defeasible logic itself is capable of encoding this information. Next, it is discussed whether the warrant procedure of the variant of DeLP is sufficient.

The main difference in the topic language of the used logic and DeLP is the absence of strict rules and predicate logic. It is assumed that all knowledge is defeasible. Even though some knowledge in law *is* strict, this knowledge can be modelled using defeasible rules that are not attacked. Even the absence of predicate logic is, using careful modelling of the knowledge reasoned upon, surmountable.

The warrant procedure of the used logic is, however, a weak point. Specifically, it is the dialectical trees that may pose a problem. The problem occurs whenever there are multiple arguments supporting the same proposition. Let, for example, q be that proposition, arguments A_0 and A_1 support q and argument A_2 attack A_0. Also, let the goal be to find an acceptable argument for q. The problem is that, using a warrant inquiry dialogue (using the exhaustive strategy), it depends on whether A_0 or A_1 is the root argument if an acceptable argument for q is found. If in such a warrant inquiry dialogue A_0 is the root argument, no acceptable argument is found. Depending on the root argument, it may therefore be falsely concluded that no argument for q has got a warrant. This is a problem, since in a dialogue it is unknown whether A_1 (which *is* acceptable) exists, as knowledge is distributed. The problem is therefore how warrant inquiry dialogues use dialectical trees (i.e. by using one root argument per dialogue).

The issue with having multiple arguments supporting the same topic can, however, be mitigated by making the dialogues more specialised. The police currently try to do a similar thing with certain types of intakes. For example, there exists a special telephone number to report cargo thefts from trucks.[2]

The dialogue system of Black and Hunter (2009) is therefore clearly not able to *fully* model intakes. After all, as explained in Section 3.3, it is safe to assume that a 'real' intake also has got elements of persuasion and information-seeking dialogues. The chosen dialogue system is, however, still a good simplified model of reality.

10.3 Contribution

Another contribution is the dialogue engine developed for the simulations. As explained in section 9.2.3, the developed dialogue engine can be used to more easily implement dialogue systems. The dialogue systems of Black and Hunter (2009) are, for example, implemented on top of this engine.

[1] Assuming that he/she also needs to be persuaded of the outcome of the dialogue.

[2] https://www.politie.nl/nieuws/2017/mei/16/11-politie-opent-speciaal-aangifteloket-ladingdiefstal.html

Chapter 11

Conclusion

This chapter will summarize the results as they have been presented in this thesis.

In an intake, an asymmetric conversation takes place. The complainant knows why he has come to file a report, but only the police officer knows whether what happened actually was in violation of the law. In other words, it is assumed that only the police officer knows the rules. Furthermore, it is the joint goal of both participants to exchange information, such that the police officer can file a report. The police officer therefore asks questions in order to gain information, while the complainant may also ask questions. Such a dialogue is an inquiry dialogue. An inquiry dialogue is, according to Walton and Krabbe (1995), a dialogue in which both participants try to prove a proposition. In case of the intake, both participants try to prove that the crime the complainant has come to report actually took place. It is assumed that they are cooperatively trying to find an acceptable argument.

The goal of this thesis was to find out how such dialogues can be made shorter. The idea being to take an existing dialogue system that generates inquiry dialogues, and improve on the corresponding strategy. In the literature several approaches are used in order to create smart strategies.

Firstly, the idea is to represent the state of the dialogue in such a way that a planner can find an 'optimal' strategy, which they define as a strategy using which the proponent has got the highest possible chance of making the dialogue succeed. In warrant inquiry dialogues, for example, the goal would be to find an acceptable argument as quickly as possible. Black et al. (2017) used propositional planning to find optimal strategies for persuasion dialogues. Alahmari et al. (2017), on the other hand, used Q-learning to let the agent find an optimal strategy on his own. The downside of such an approach is, however, that a training phase must be held before going into a dialogue.

Another approach is using a user model, such that an agent can make better decisions, using a model of his opponent. Such a user model can be build upon previous experiences, the history of the dialogue or other things. Hadoux et al. (2015) put this information into the knowledge base itself (using probabilistic rules), such that an external tool (after converting the dialogue into a MOMDP) could 'plan' an optimal strategy. Hadjinikolis et al. (2013) and Rienstra et al. (2013) both tried to predict the state of the opponent, by adding recursive user models. Hadjinikolis et al. learned the user model from previous experiences,

while Rienstra et al. kept track of the current state of the dialogue.

The work mentioned above assumes the existence of a persuasion dialogue. The goal of an inquiry dialogue, however, is completely different from a persuasion dialogue. Therefore, it is not trivial to convert the described strategies to work on inquiry dialogues.

Black and Hunter (2009) described two inquiry dialogue systems, along with a corresponding strategy (the exhaustive strategy). They described an argument inquiry dialogue system and a warrant inquiry dialogue system. Argument inquiry dialogues try to find an argument for the goal proposition, while warrant inquiry dialogues also check whether the argument is acceptable. Of these dialogue system, warrant inquiry is used in the simulations as it models the intake the best, as explained in the discussion.

The exhaustive strategy is a strategy designed to be sound and complete (hence exhaustive). Using this strategy, agents first try to assert an argument, then open a sub-dialogue, or otherwise try to close the current dialogue. The exhaustive strategy, however, generates long dialogues. It was therefore researched whether a modification of the exhaustive strategy could improve upon the dialogues generated by the exhaustive strategy. In other words, it was investigated whether those dialogues could be made more relevant.

Two modifications on the exhaustive strategy were devised and compared to the exhaustive strategy in warrant inquiry dialogues. The idea being to devise strategies that do not require extra information. The first modification only asserted arguments for which no earlier argument was asserted. The second only asserted 'relevant' arguments, i.e. arguments that change the status of the root argument in a dialectical tree. This is an implementation of strong relevancy as defined by Prakken (2005). Both modifications were formalised into two variants. The first one (the limited strategies) was split in a strategy which only looks at the dialogue to find out which arguments were uttered (limited dialogue), and another which looks at the union of the commitment stores (limited commitment). The second modification (the smart strategies) was split in a strategy that uses Black and Hunter's dialectical tree (smart original strategy), and another that uses a tree based on the dialogue (smart dialogue strategy). The smart dialogue strategy was inspired by the dialectical graph as used in Prakken (2005).

A setting inspired by the intake was used in the first set of experiments. In this 'intake setting' the proponent knows all the rules, and the opponent all the facts. This distribution models an intake wherein the police officer knows the law (the rules), and the complainant what happened (the facts).

A first set of experiments researched the effect of the modified strategies on the size of the generated dialogues and dialectical trees. In this intake setting, the modified strategies were tested using different sets of dialectical trees. Per data set, the exhaustive and modified strategies were tested on different sizes of the data set. The resulting status of the root argument, the number of moves uttered and the number of arguments in the dialectical tree were recorded.

The influence of the intake setting was investigated by a second set of experiments. This experiment was set in the 'random setting'. This means that dialogues were simulated using the same strategies and sets of dialectical trees (data sets), but with randomized distributions of knowledge. In this randomization, each fact or rule had a 50% chance of being in either of the agent's knowledge bases. Per combination of a strategy, data set, and a particular size

of that data set 30 dialogues were simulated, each with different distributions of knowledge.

In addition to the experiments, it was also proven whether the modifications were sound and/or complete.

In conclusion, none of the modified strategies were sound and complete, *and* generated significantly smaller dialogues and dialectical trees. Nevertheless, two strategies systematically reduced either the size of the dialogue or dialectical tree. The smart dialogue, limited commitment and limited dialogue strategies, were however not sound and complete for warrant inquiry dialogues. The smart original strategy was sound and complete but did not generate smaller dialogues or dialectical trees when compared to the exhaustive strategy.

It was also found that the influence of the intake setting heavily depends on the type of dialectical tree it is tested. The same holds true for the behaviour of the modified strategies.

Most importantly, empirical evidence was found that the modified strategies did not completely eradicate irrelevant arguments. To improve those strategies two things can be done. Firstly, strategies could be allowed to reason using extra information (using a user model). Secondly, different criteria may need to be set upon the strategies (using other definitions of soundness and completeness). In addition, work can be done to ensure that the irrelevant arguments that are otherwise asserted, are not found, by limiting the number of sub-dialogues.

11.1 Future work

It was found that improving existing strategies, without adding extra information to reason about, is hard. For example, the smart original strategy was only a marginal improvement. Therefore, it may be interesting to explore whether extra information can allow a strategy to be 'smarter'. Using a user model, for example, a strategy may be able to say earlier in a dialogue that certain propositions will never become *in* again. Then, it may skip those propositions.

Furthermore, in this thesis it is assumed that both participants use the same strategy for simplicity reasons. An intake, however, is an asymmetric dialogue. Therefore, dialogues wherein agents may have different strategies could be better models of the intake. For example, the goal of the complainant may be to complete the intake as soon as possible, while the police officer may be to be more thorough.

Different notions of soundness and/or completeness can also be researched. Using Black and Hunter's notion, the limited strategies are not interesting, even though they do reduce the size of dialogues and dialectical trees. Moreover, maybe weaker notions of soundness and completeness may also be acceptable for modelling intake dialogues.

References

Alahmari, S., Yuan, T., & Kudenko, D. (2017). Reinforcement learning for abstract argumentation: A Q-learning approach. In *Adaptive and Learning Agents workshop (at AAMAS '17)*.

Arioua, A., & Croitoru, M. (2015). Formalizing Explanatory Dialogues. In *Scalable Uncertainty Management* (pp. 282–297). Springer International Publishing.

Bex, F., Peters, J., & Testerink, B. (2016). *A.I. for online criminal complaints: From natural dialogues to structured scenarios.* ECAI 2016 workshop on Artificial Intelligence for Justice.

Black, E., Coles, A., & Hampson, C. (2017). Planning for Persuasion. In *Proceedings of the Sixteenth International Conference on Autonomous Agents and Multiagent Sytems* (pp. 933–942).

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

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.

Garcia, A. J., & Simari, G. R. (2004). Defeasible Logic Programming: An Argumentative Approach. *Theory and Practice of Logic Programming*, *4*(2).

Hadjinikolis, C., Siantos, Y., Modgil, S., Black, E., & McBurney, P. (2013). Opponent modelling in persuasion dialogues. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence* (pp. 164–170).

Hadoux, E., Beynier, A., Maudet, N., Weng, P., & Hunter, A. (2015). Optimization of probabilistic argumentation with Markov decision models. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence* (Vol. 2015-01, pp. 2004–2010).

Hecham, A., Croitoru, M., & Bisquert, P. (2018). A First Order Logic Benchmark for Defeasible Reasoning Tool Profiling. In *Rules and Reasoning* (pp. 81–97). Springer. Springer International Publishing.

Mason, M. (2016, May). The 'preparatory' and 'argumentation' stages of police interrogation: A linguistic analysis of a criminal investigation. *Language & Communication*, *48*, 79–87.

Prakken, H. (2005). Coherence and Flexibility in Dialogue Games for Argumentation. *Journal of Logic and Computation*, *15*(6), 1009–1040.

Prakken, H. (2017). Historical Overview of Formal Argumentation. *IfCoLog Journal of Logics and their Applications*, *4*(8), 2183–2262.

Rienstra, T., Thimm, M., & Oren, N. (2013). Opponent models with uncertainty for strategic argumentation. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence Opponent* (pp. 332–338).

Thimm, M. (2017). The Tweety Library Collection for Logical Aspects of Artificial Intelligence and Knowledge Representation. *Künstliche Intelligenz*, *31*(1), 93–97.

van Charldorp, T. C. (2014). What happened? From talk to text in police interrogations. *Language & Communication*, *36*(1), 7–24.

Walton, D. (2003). The interrogation as a type of dialogue. *Journal of Pragmatics*, *35*(12), 1771–1802.

Walton, D., & Krabbe, E. C. W. (1995). *Commitment in dialogue: Basic concepts of interpersonal reasoning*. SUNY press.

Walton, D., & Macagno, F. (2007). Types of Dialogue, Dialectical Relevance and Textual Congruity. *Anthropology and Philosophy*, *8*(1/2), 101–120.

Appendix A

Intake setting

This appendix contains plots for the experiments performed with a specific distribution of knowledge. See Chapter 9 for details. Per combination of a strategy, data set, and a particular size of that data set, the size of the generated dialogue and dialectical tree is shown.

A.1 Ambiguity

The first plot shows for all strategies the size of the generated dialectical trees. Plot A.1.2 and A.1.3 show the same strategies, only split over multiple plots for clarity.

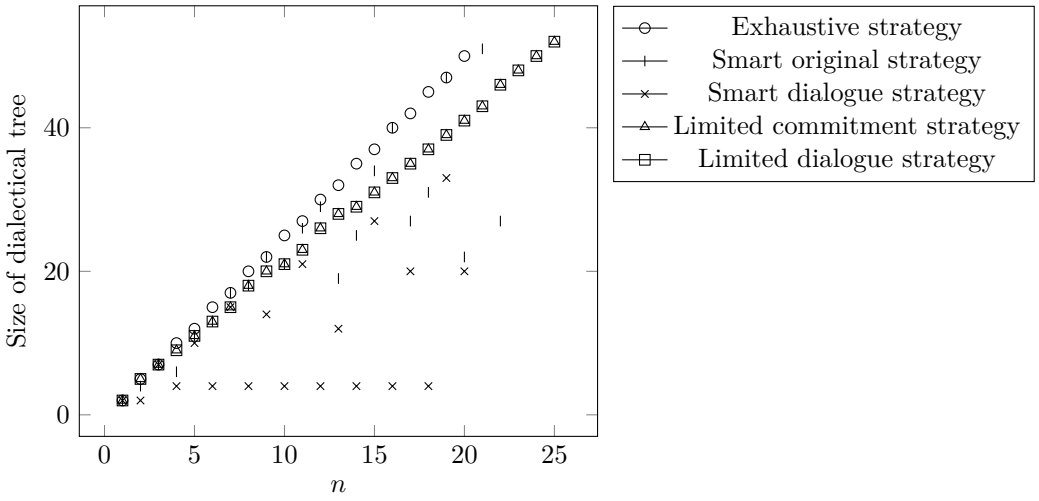


Figure A.1.1: The size of the dialectical trees constructed by all strategies. The strategies are plotted on top of each other.

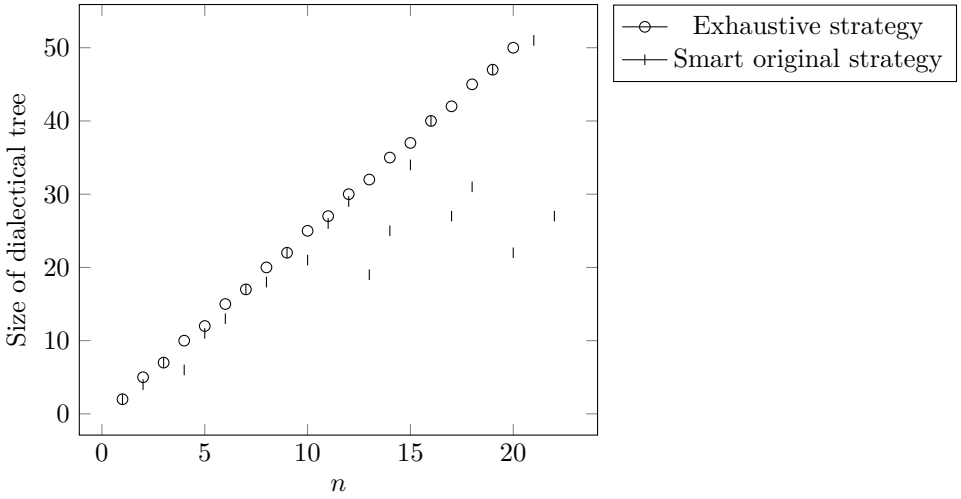


Figure A.1.2: The size of the dialectical trees constructed by the exhaustive and smart original. The strategies are plotted on top of each other.

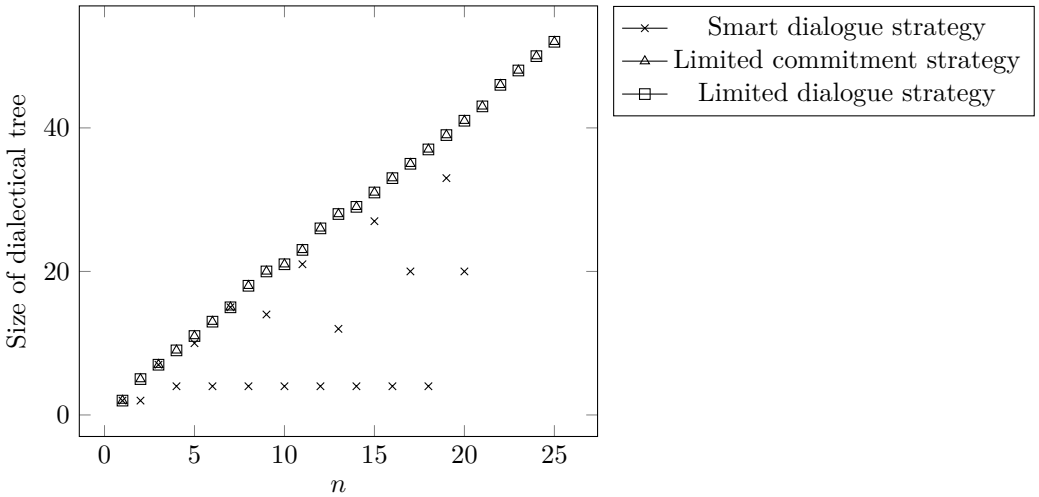


Figure A.1.3: The size of the dialectical trees constructed by the smart dialogue, limited commitment and limited dialogue strategy. The limited strategies are plotted on top of each other.

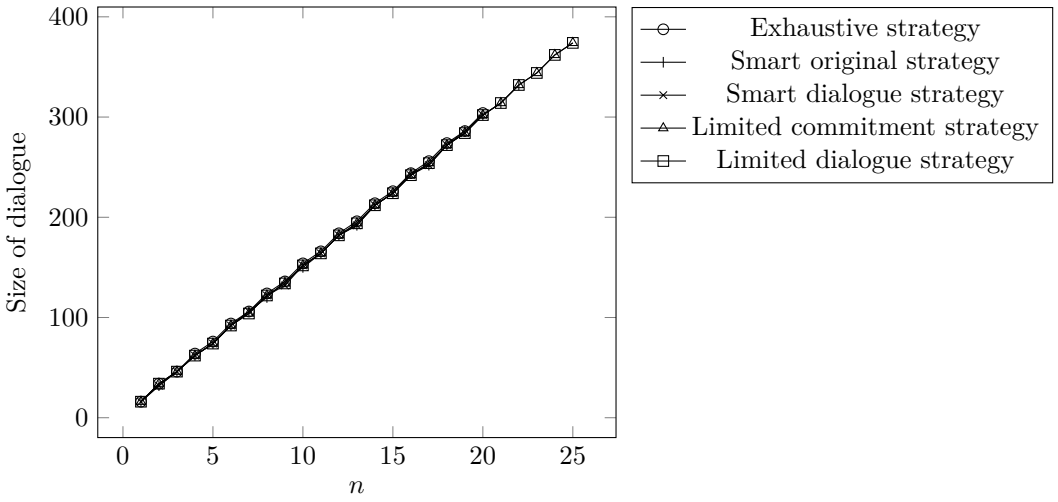


Figure A.1.4: The size of the dialogues generated on the ambiguity data set. All strategies generate the same length of dialogues.

A.2 Team defeat

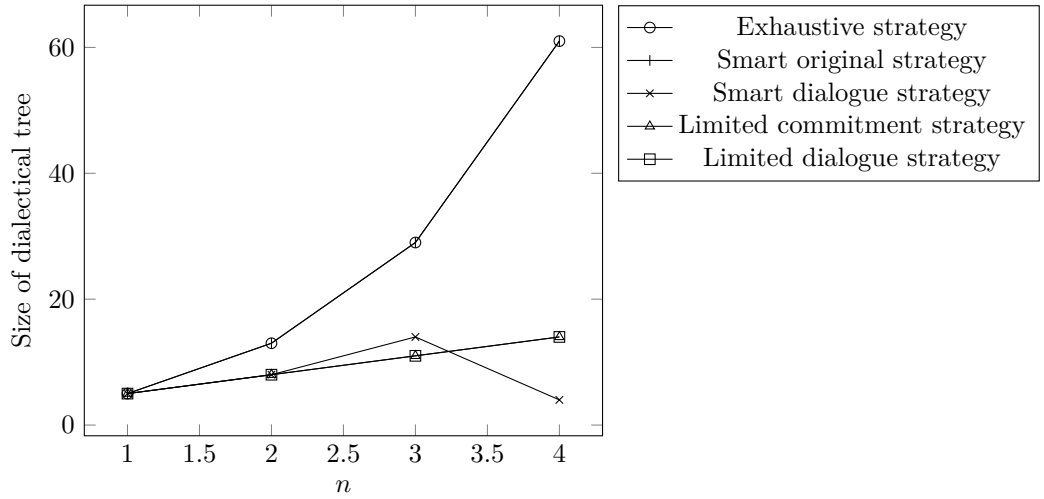


Figure A.2.1: This plot shows for team defeat the size of the dialectical tree. Note that the limited commitment and limited dialogue strategy are plotted on top of each other. Until $n = 3$ the smart dialogue strategy overlaps with the exhaustive and smart original strategy.

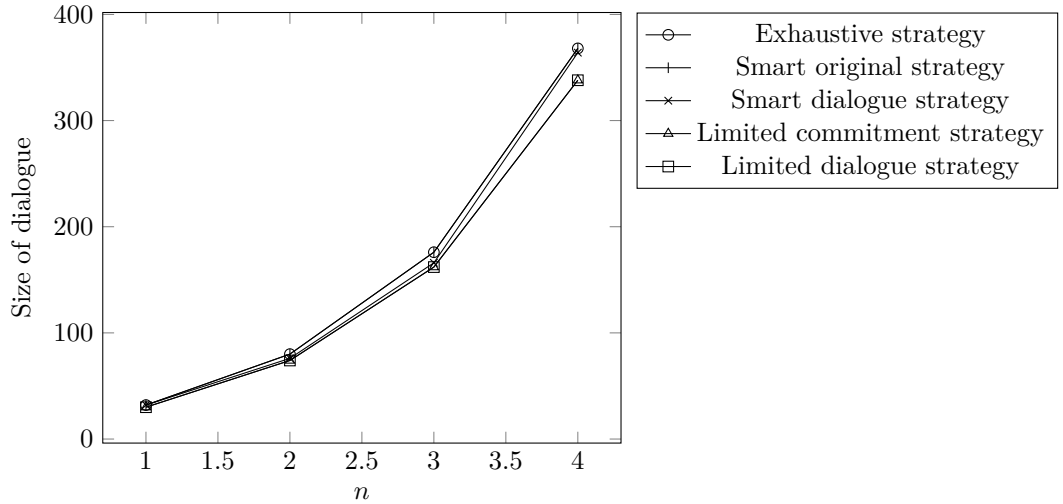


Figure A.2.2: The size of the generated dialogues, plotted per strategy. Here, the exhaustive, smart original and smart dialogue strategies are plotted on top of each other. Similarly, the limited strategies are also plotted on top of each other.

A.3 Floating conclusions

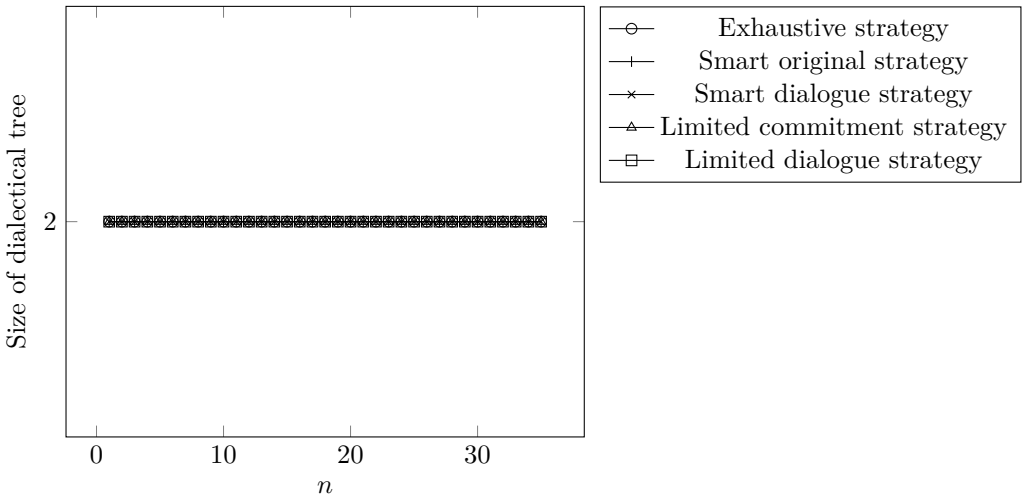


Figure A.3.1: The size of the dialectical trees constructed by the generated dialogues. The size of the dialectical tree is in all instances 2.

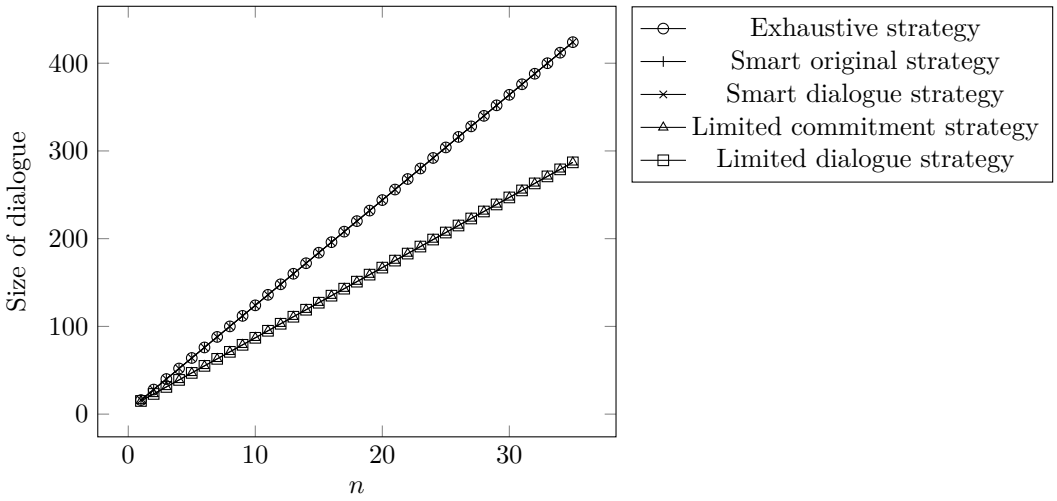


Figure A.3.2: The size of the dialogues generated on the floating conclusions data set by the specified strategies. Here, the exhaustive, smart original and smart dialogue strategies are plotted on top of each other. Similarly, the limited strategies are also plotted on the same line.

Appendix B

Root arguments in the intake setting

Here, per simulated dialogue in the intake setting is shown what the resulting status of the root argument was.

B.1 Ambiguity

n	exhaustive	smart original	smart dialogue	limited commitment	limited dialogue
1	Out	Out	Out	Out	Out
2	Out	Out	Out	Out	Out
3	In	In	In	In	In
4	Out	Out	Out	Out	Out
5	Out	Out	Out	Out	Out
6	Out	Out	Out	Out	Out
7	In	In	In	Out	Out
8	Out	Out	Out	Out	Out
9	Out	Out	Out	In	In
10	Out	Out	Out	Out	Out
11	In	In	In	Out	Out
12	Out	Out	Out	Out	Out
13	Out	Out	Out	In	In
14	Out	Out	Out	Out	Out
15	In	In	In	Out	Out
16	Out	Out	Out	Out	Out
17	Out	Out	Out	Out	Out
18	Out	Out	Out	Out	Out
19	In	In	In	Out	Out
20	Out	Out	Out	Out	Out
21		Out		Out	Out
22		Out		Out	Out
23				In	In
24				Out	Out
25				In	In

B.2 Team defeat

n	exhaustive	smart original	smart dialogue	limited commitment	limited dialogue
1	In	In	In	In	In
2	In	In	Out	In	In
3	In	In	Out	In	In
4	In	In	Out	In	In

B.3 Floating conclusions

n	exhaustive	smart original	smart dialogue	limited commitment	limited dialogue
1	Out	Out	Out	Out	Out
2	Out	Out	Out	Out	Out
3	Out	Out	Out	Out	Out
4	Out	Out	Out	Out	Out
5	Out	Out	Out	Out	Out
6	Out	Out	Out	Out	Out
7	Out	Out	Out	Out	Out
8	Out	Out	Out	Out	Out
9	Out	Out	Out	Out	Out
10	Out	Out	Out	Out	Out
11	Out	Out	Out	Out	Out
12	Out	Out	Out	Out	Out
13	Out	Out	Out	Out	Out
14	Out	Out	Out	Out	Out
15	Out	Out	Out	Out	Out
16	Out	Out	Out	Out	Out
17	Out	Out	Out	Out	Out
18	Out	Out	Out	Out	Out
19	Out	Out	Out	Out	Out
20	Out	Out	Out	Out	Out
21	Out	Out	Out	Out	Out
22	Out	Out	Out	Out	Out
23	Out	Out	Out	Out	Out
24	Out	Out	Out	Out	Out
25	Out	Out	Out	Out	Out
26	Out	Out	Out	Out	Out
27	Out	Out	Out	Out	Out
28	Out	Out	Out	Out	Out
29	Out	Out	Out	Out	Out
30	Out	Out	Out	Out	Out
31	Out	Out	Out	Out	Out
32	Out	Out	Out	Out	Out
33	Out	Out	Out	Out	Out
34	Out	Out	Out	Out	Out
35	Out	Out	Out	Out	Out

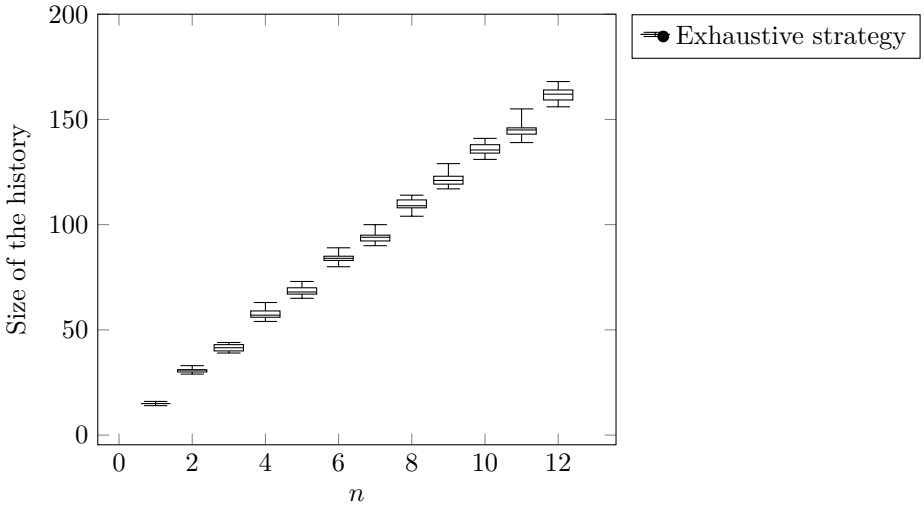
Appendix C

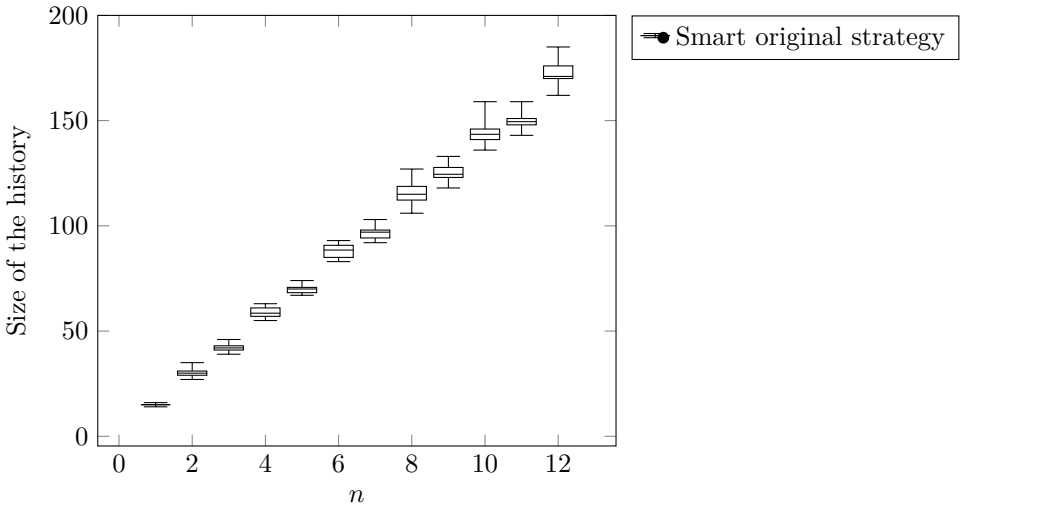
Random setting

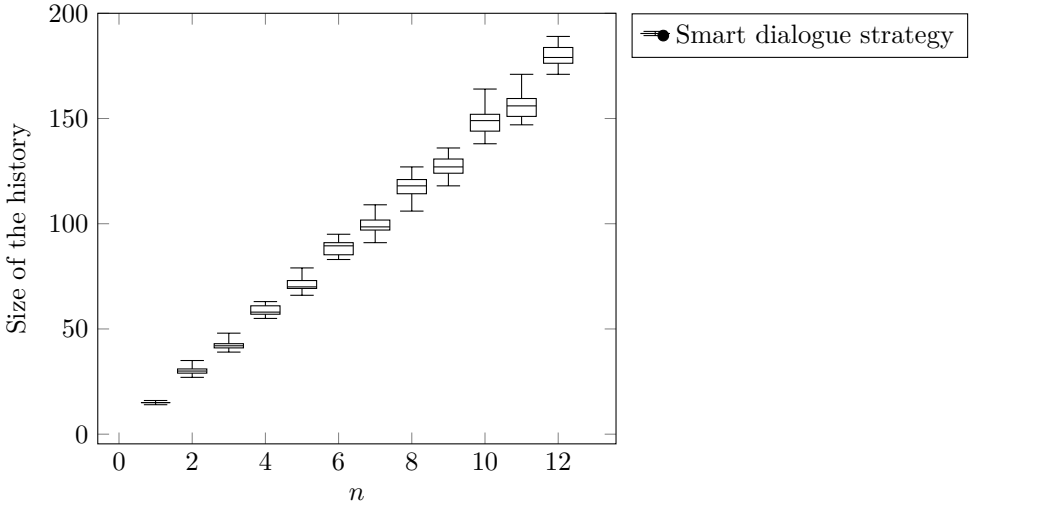
Per combination of strategy, data set and a particular size of that data set, 30 dialogues each with a random distribution of knowledge were simulated. Note that the tests done here are on a smaller n than in the 'normal' tests. This is because of time constraints.

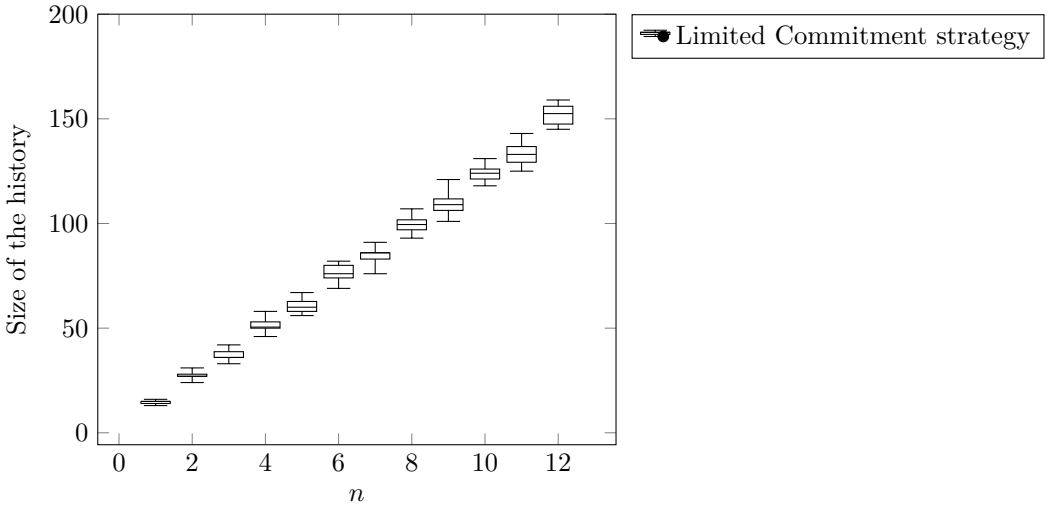
C.1 Ambiguity

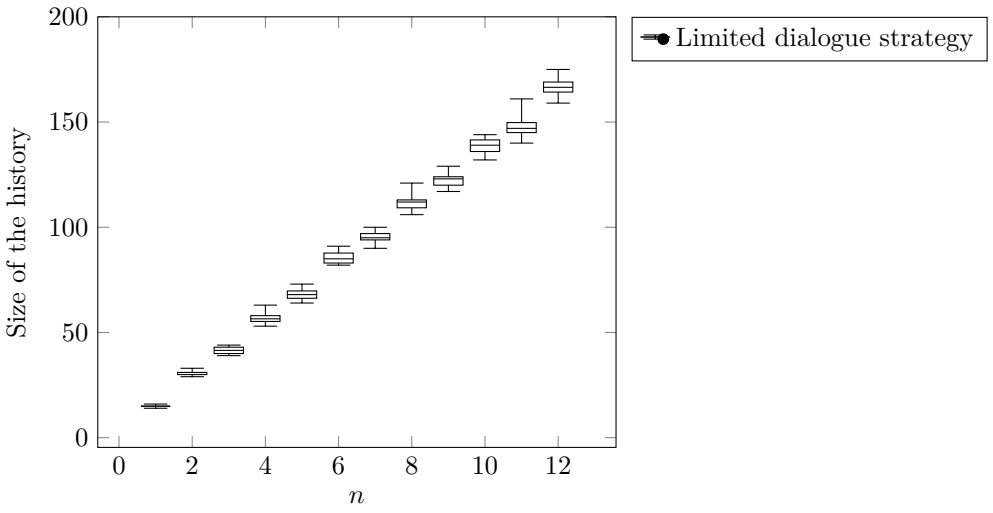
C.1.1 Size of dialogues



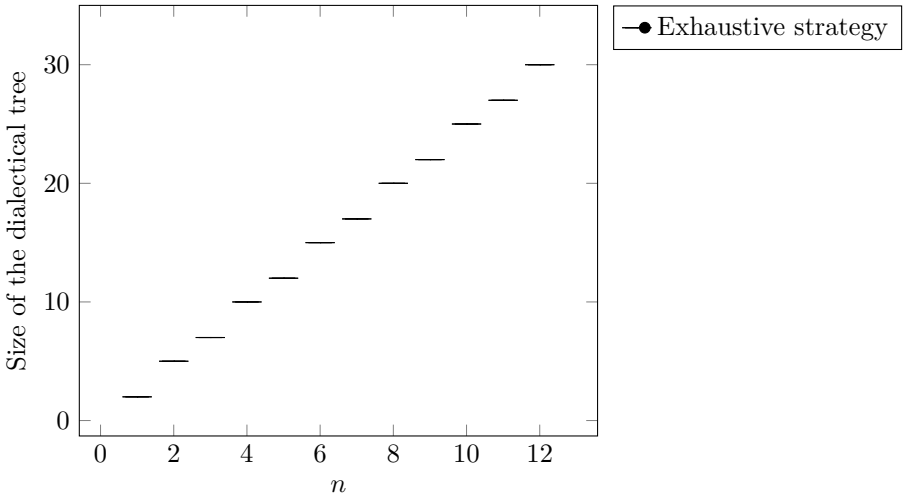

Smart original strategy
Size of the history
n
0
50
100
150
200
0
2
4
6
8
10
12

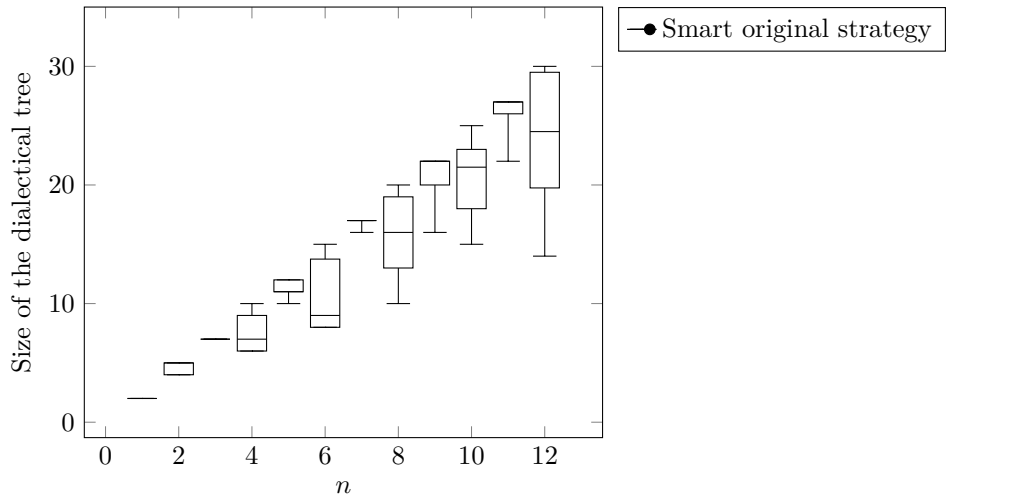

Smart dialogue strategy
Size of the history
n
0
50
100
150
200
0
2
4
6
8
10
12

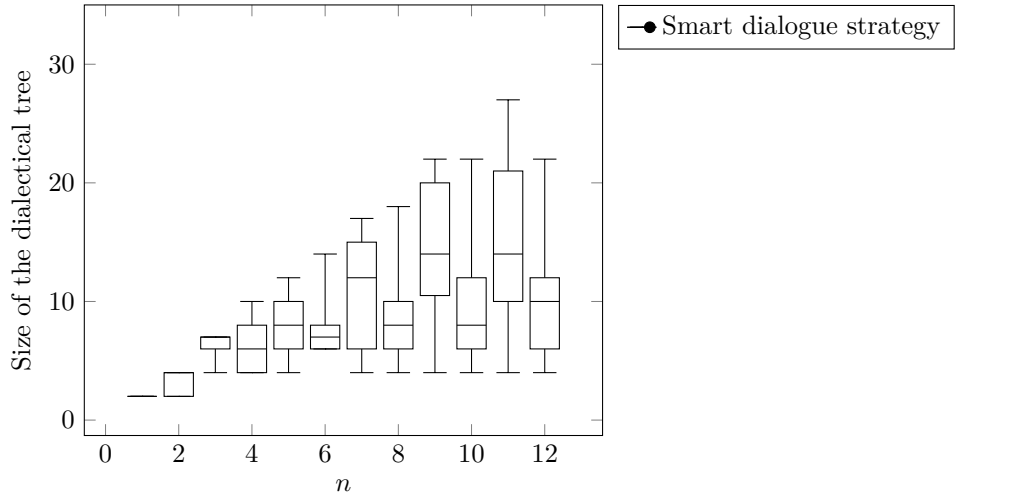

Limited Commitment strategy
Size of the history
n
0
50
100
150
200
0
2
4
6
8
10
12

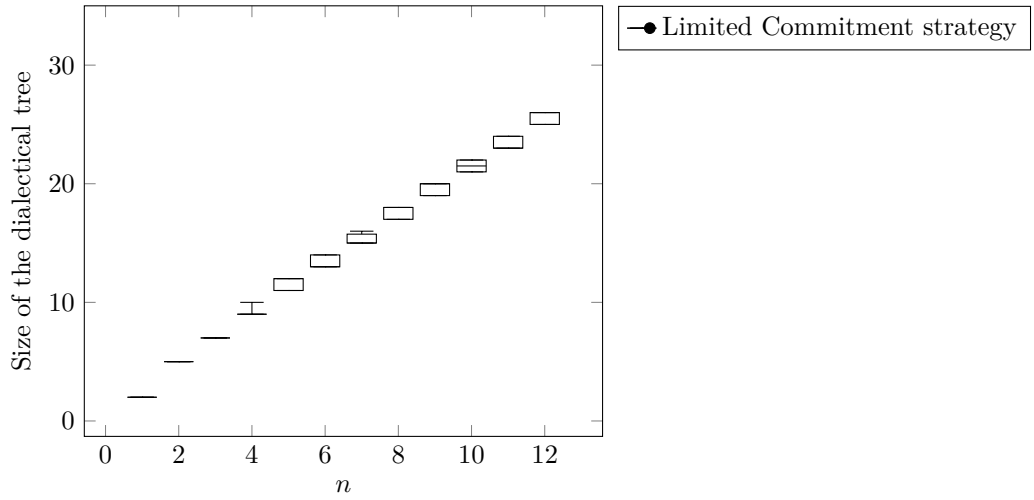


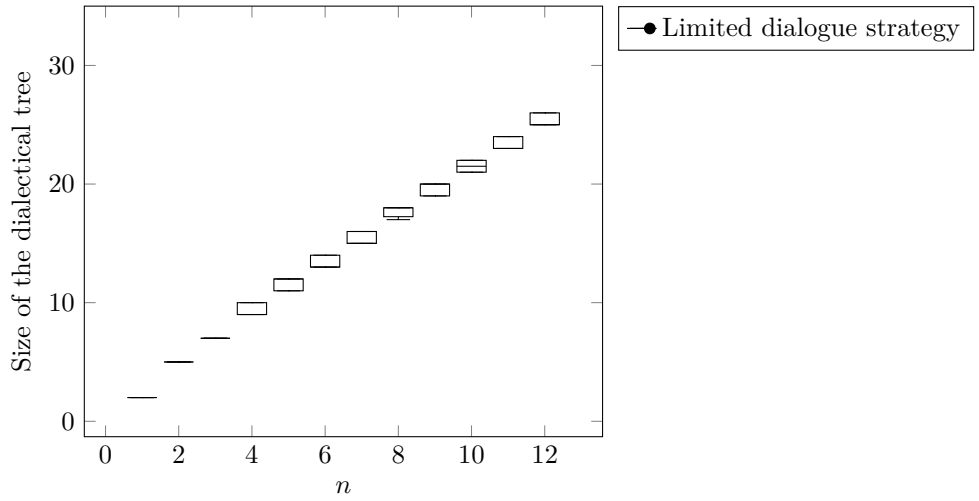
C.1.2 Size of dialectical trees




Smart original strategy
Size of the dialectical tree
0
10
20
30
0
2
4
6
8
10
12
n

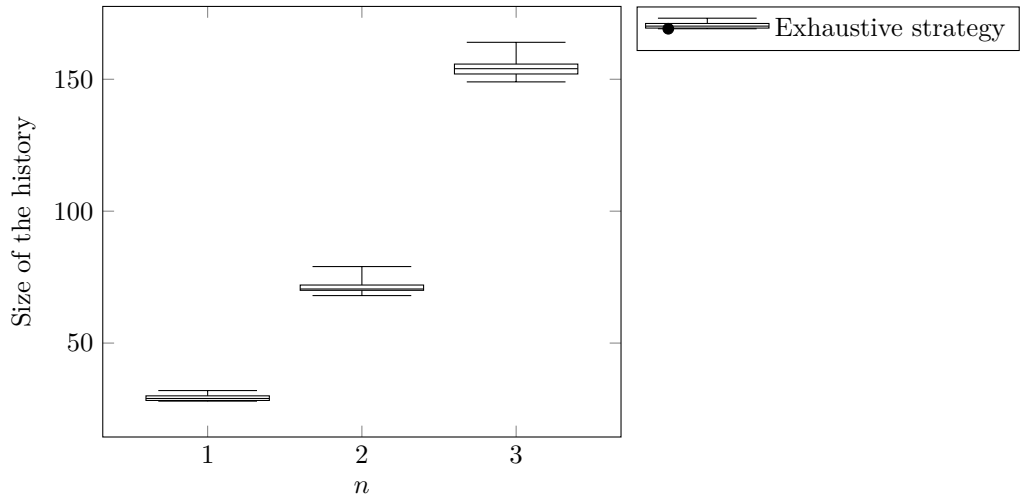

Smart dialogue strategy
Size of the dialectical tree
0
10
20
30
0
2
4
6
8
10
12
n

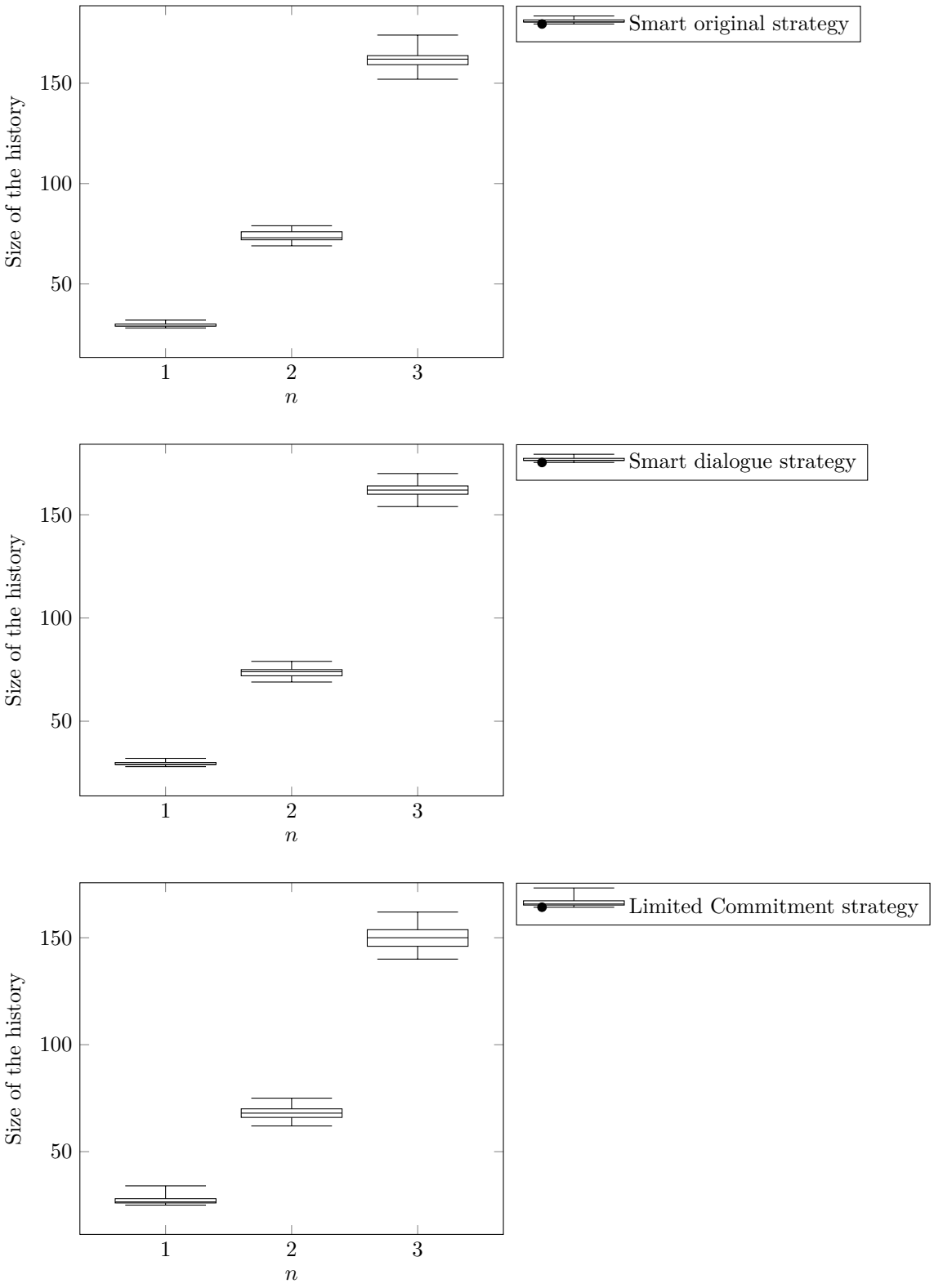

Limited Commitment strategy
Size of the dialectical tree
0
10
20
30
0
2
4
6
8
10
12
n

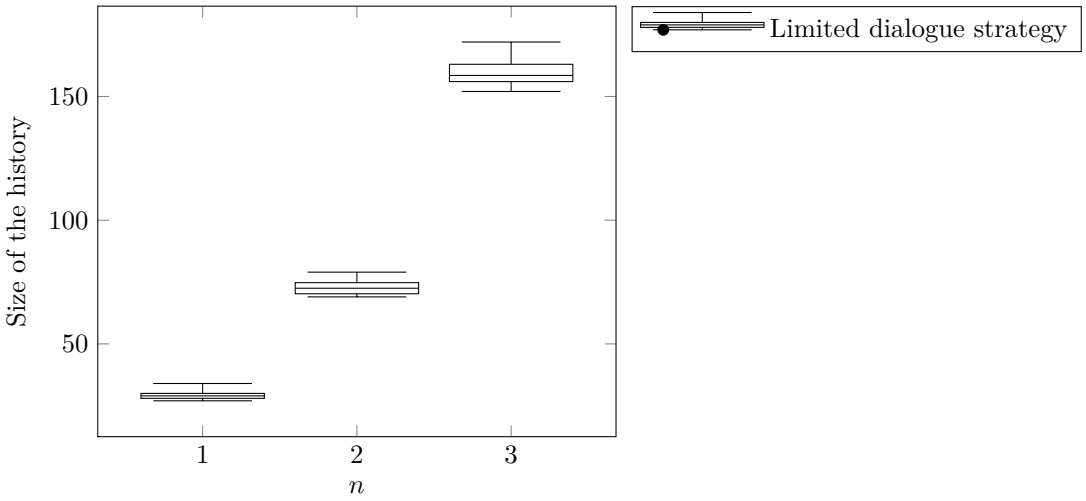


C.2 Team Defeat

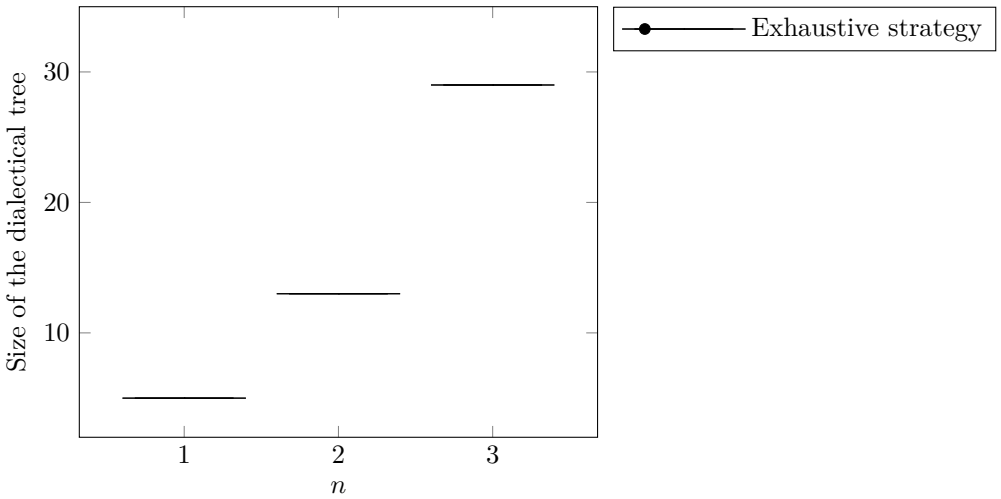
C.2.1 Size of dialogues

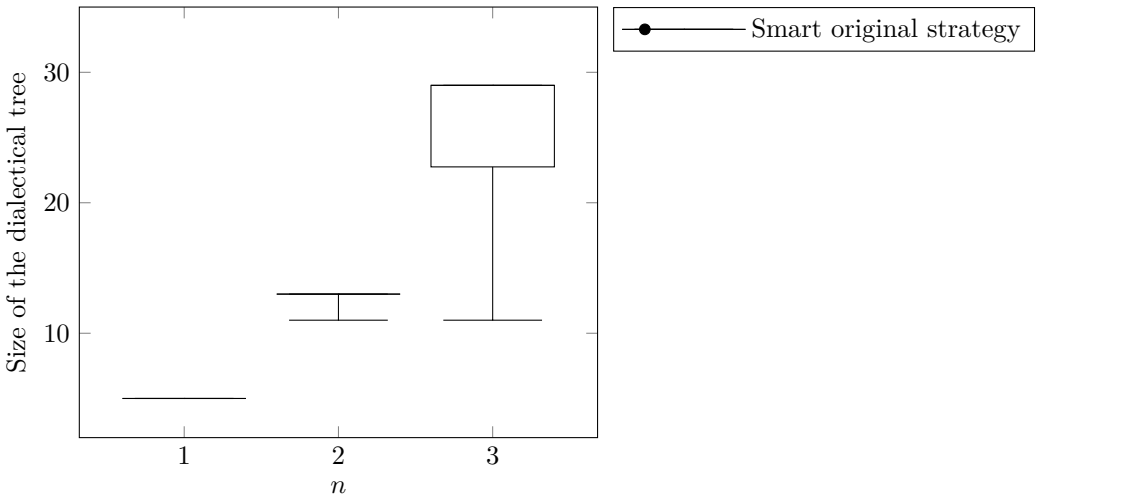


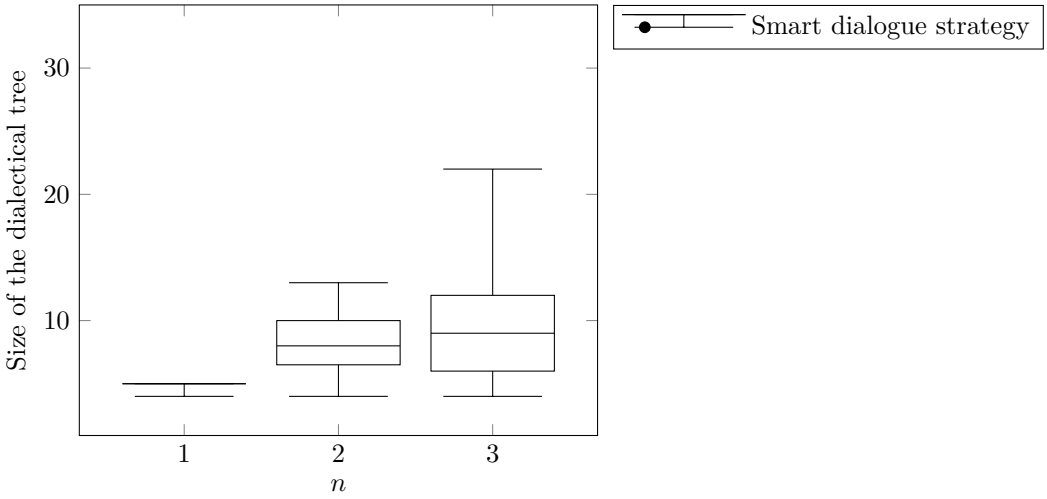

Smart original strategy
Size of the history
150
100
50
1
2
3
n
Smart dialogue strategy
Size of the history
150
100
50
1
2
3
n
Limited Commitment strategy
Size of the history
150
100
50
1
2
3
n

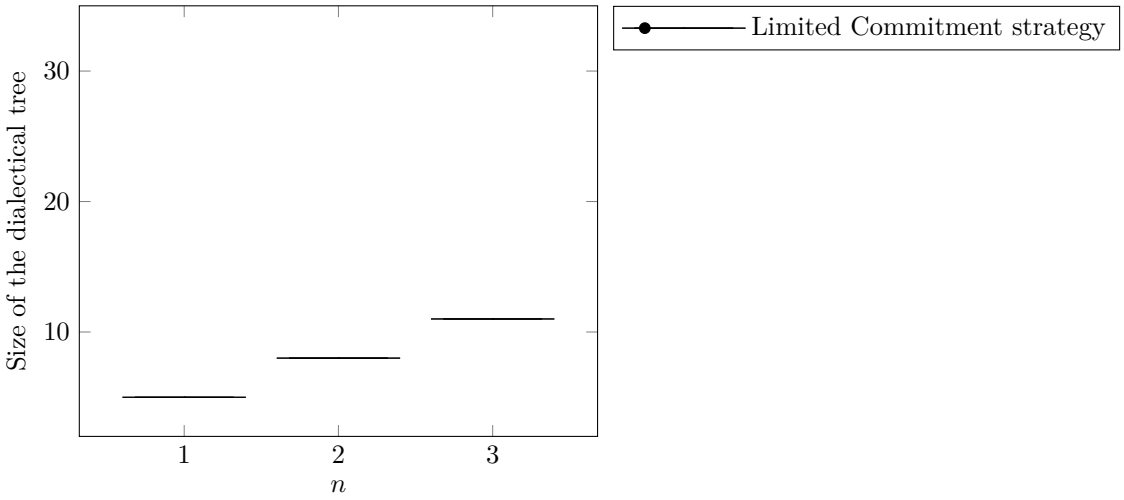


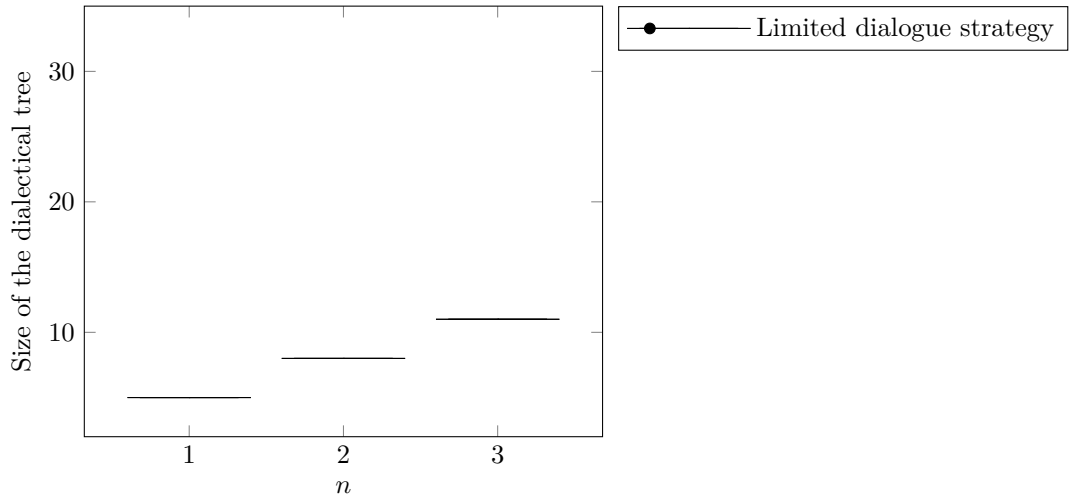
C.2.2 Size of dialectical trees




Smart original strategy
Size of the dialectical tree
30
20
10
1
2
3
n

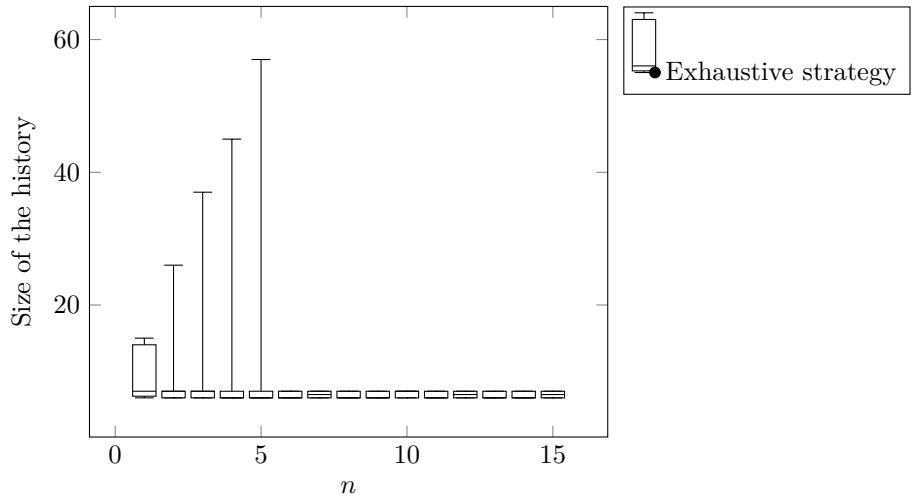

Smart dialogue strategy
Size of the dialectical tree
30
20
10
1
2
3
n

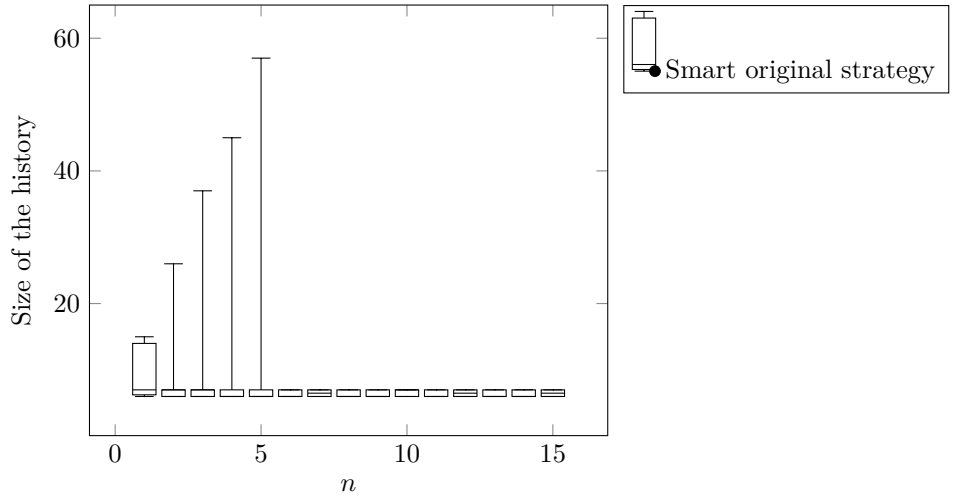

Limited Commitment strategy
Size of the dialectical tree
30
20
10
1
2
3
n

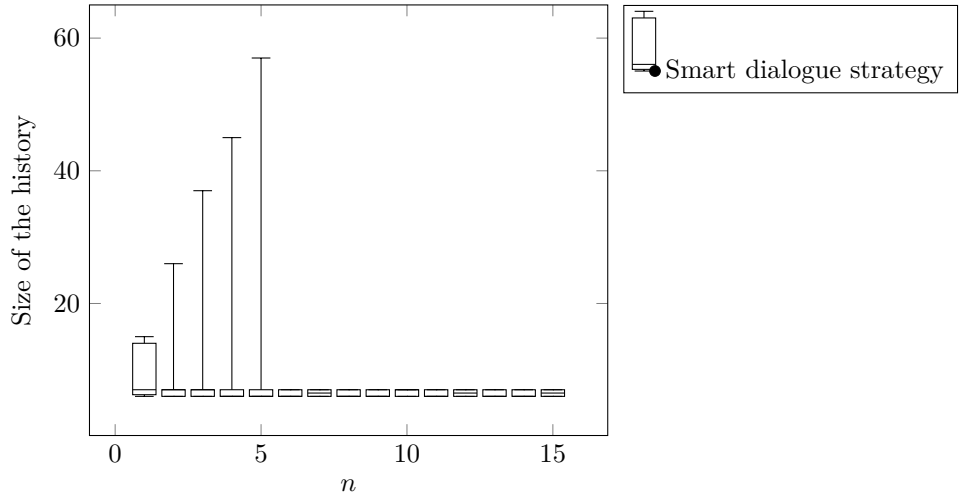


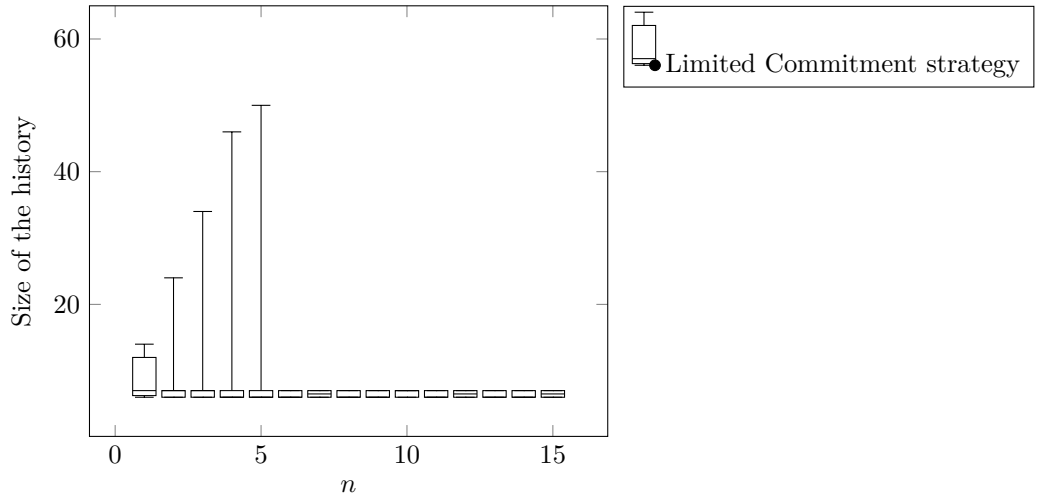
C.3 Floating conclusions

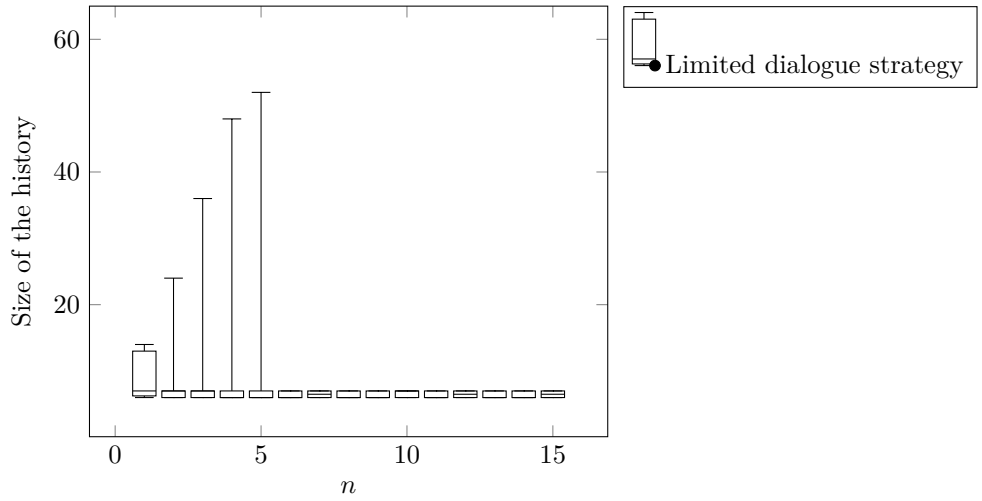
C.3.1 Size of dialogues



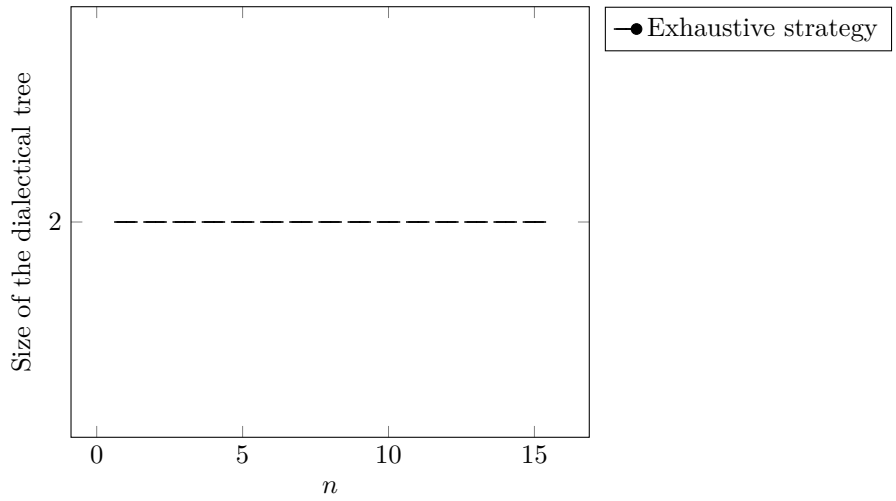

Size of the history
60
40
20
0
5
10
15
n
Smart original strategy

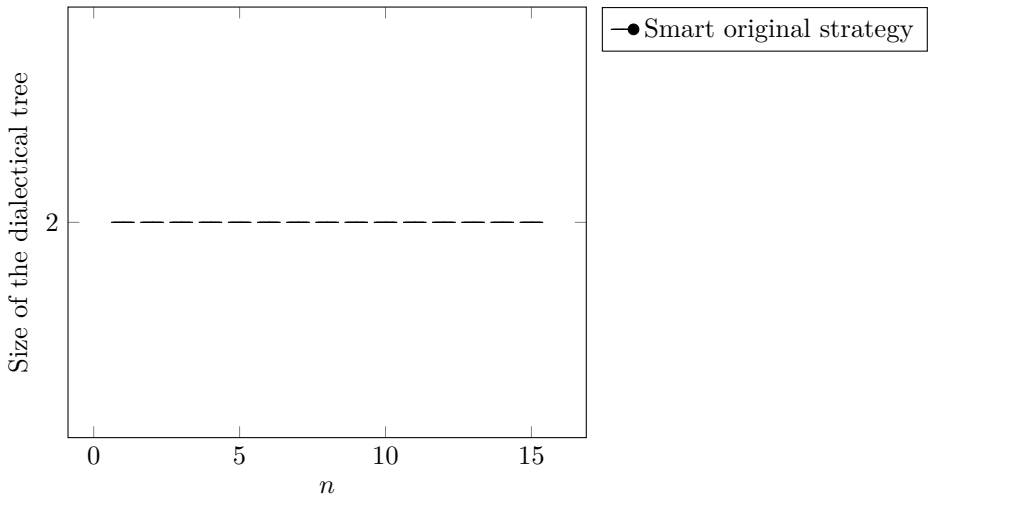

Size of the history
60
40
20
0
5
10
15
n
Smart dialogue strategy

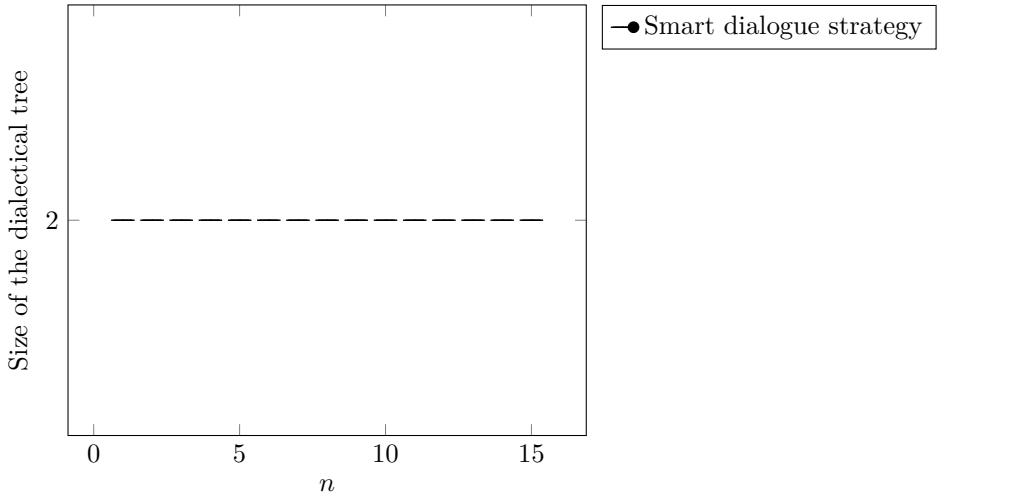

Size of the history
60
40
20
0
5
10
15
n
Limited Commitment strategy

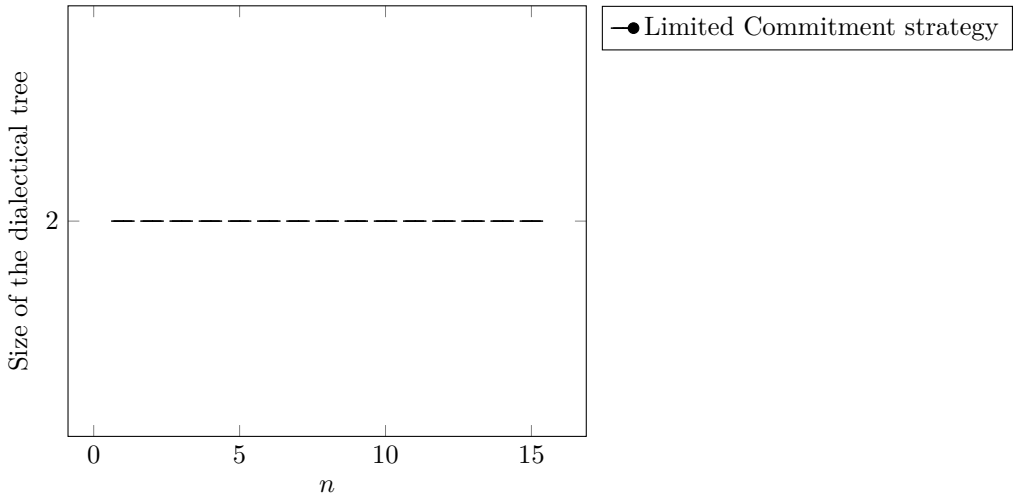


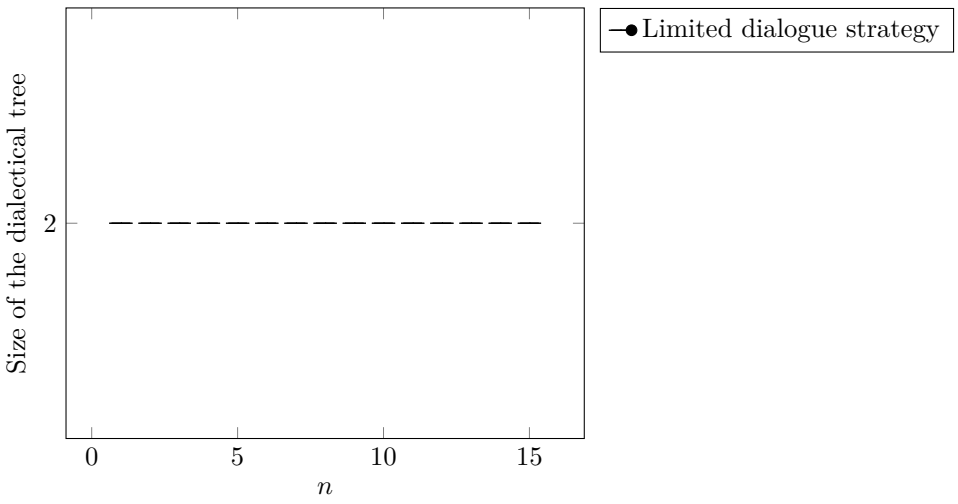
C.3.2 Size of dialectical trees




Smart original strategy
Size of the dialectical tree
2
0
5
10
15
n


Smart dialogue strategy
Size of the dialectical tree
2
0
5
10
15
n


Limited Commitment strategy
Size of the dialectical tree
2
0
5
10
15
n


Limited dialogue strategy
Size of the dialectical tree
2
0
5
10
15
n