Inquiry dialogues inside a framework for persuasion

Author:
Wouter Bouvy
(45 ECTS)

Supervisors:
Henry Prakken
Liz Black

February 7, 2011

Abstract

This thesis is a comparison of a system for inquiry dialogues and a framework for persuasion dialogues in order to reconcile their differences, determine which differences cannot be reconciled and determine the cost of this reconciliation to their fundamental properties.
The systems that are compared are Prakken's framework for persuasion dialogues and Black and Hunter's system for inquiry dialogues. Two new systems are defined where the structure of Black and Hunter's system is described using Prakken's framework. In one of the systems, the structure of Black and Hunter's system is adapted, while maintaining most of the fundamental properties. In the other system, Black and Hunter's system is unchanged, and Prakken's framework is adapted where necessary. For both systems, conversion functions are given and proofs are provided to show that dialogues in the systems are equivalent to dialogues in the original system.
From the two new systems, it is shown that inquiry and persuasion are closely related, and that both systems are capable of expressing both types of dialogue with a few specific alterations. Most research that has been done for persuasion dialogues is therefore likely to be applicable to inquiry dialogues as well.

Contents

1 Introduction

For years, the field of Artificial Intelligence has seen a shift from the focus on standalone computer systems, into a situation of multiple connected systems, agents, working together or competing to achieve certain goals. These days, there are many protocols and logics to choose from when constructing a system for agents.

Dialogue games are a method of formalising the communications between software agents. The agents (players) construct a dialogue using the rules of the dialogue game. Through dialogue, agents can model many types of behaviour. Legal reasoning can be modeled, either cooperative or adversarial. By nature, dialogue games are defeasible; any argument that has been asserted can be defeated by contrary evidence asserted by the other agent.

Walton and Krabbe set out to classify the different types of dialogue game. In [10], they identified six types of dialogue games, based on the main goal of the dialogue and the initial starting situation. The tree below was shown in [10].

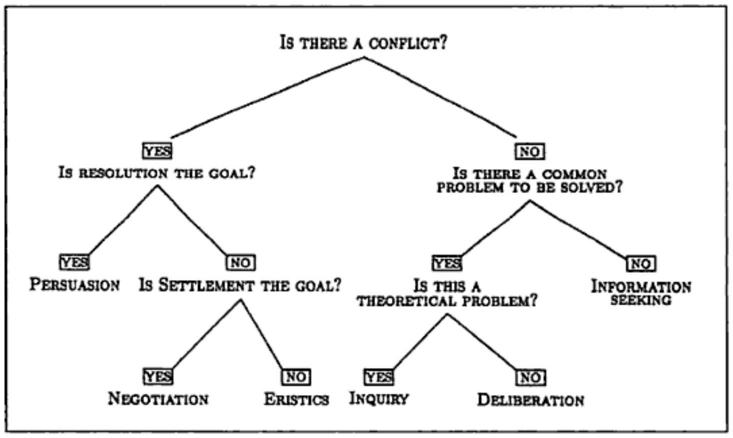


Figure 1: Walton and Krabbe's decision tree to determine a dialogue type, from [10].

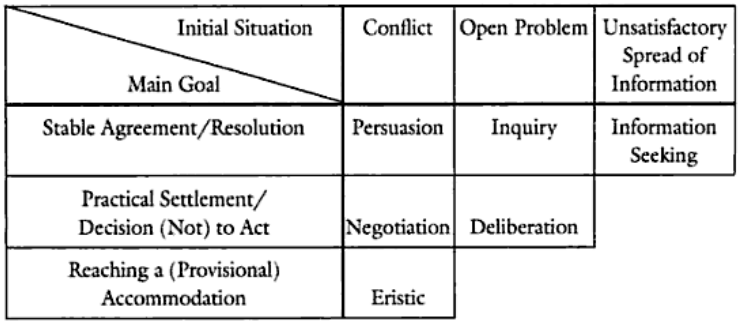
First, there are dialogue games that are based on a conflict. Conflicts about beliefs can be resolved through rational dialogue, wherein one player defends the belief and the other attacks it, until a stable agreement has been reached. This type of dialogue game is called **persuasion**. Walton and Krabbe define persuasion as:

> Persuasion dialogue (critical discussion) always arises from a conflict of opinions, and its goals is to resolve the conflict, to arrive at a final outcome of stable agreement in the end. [from: [10], page 79]

Alternatively, if there is no conflict about a belief, because the agents' individual belief base is not sufficient to support the belief or its negation, an **inquiry** dialogue allows agents to converse about a topic in a cooperative setting. Agents in an inquiry dialogue jointly construct an argument for or against the topic of the dialogue, sharing their beliefs whenever it is appropriate. Walton an Krabbe define inquiry as:

> Inquiry (. . .) is like persuasion dialogue (. . .) in that it aims at a stable agreement. However, (. . .) it arises from a problem rather than a conflict: something is not known definitely to be true or false. This not knowing can be a problem where it is useful to clarify or establish one way or the other whether something can be proved or not, according to standards of proof appropriate for the particular context of the inquiry. [from: [10], page 80]

The other dialogue types are also defined by their initial situation and main goal: **information seeking** allows agents to gather information from a source with vastly more knowledge than they have; **negotiation** allows agents to construct a resolution for their conflicting interests, **deliberation** allows agents to jointly decide on an action to take, proposing and evaluating different possible actions; **eristic** allows agents to air their grievances and seek compensation.



Initial Situation / Main Goal	Conflict	Open Problem	Unsatisfactory Spread of Information
Stable Agreement/Resolution	Persuasion	Inquiry	Information Seeking
Practical Settlement/ Decision (Not) to Act	Negotiation	Deliberation	
Reaching a (Provisional) Accommodation	Eristic		

Figure 2: Walton and Krabbe's table of dialogue types based on main goal and initial status, from [10].

Since Walton and Krabbe's definition much has been written about the different types of dialogues. Persuasion and deliberation are the more popular of the six types, while many of the others remain sparsely discussed.

The difference between the tree and the table above is striking. The table clearly shows similarities between several of the dialogue types that are not visible in the tree. Specifically, persuasion and inquiry appear to be quite different in the tree, but are somewhat related in the table. Persuasion and inquiry have the same main goal, according to the table, though they differ on the initial situation.

The question that arises from the table is whether the difference between a 'conflict' and an 'open problem' is sufficient to separate the persuasion and inquiry dialogue types[1] and similarly, negotiation and deliberation dialogue types. Since deliberation and negotiation are more complex than persuasion and inquiry, persuasion and inquiry are the easiest dialogue types to examine this difference with.

This thesis focuses on the similarity between persuasion and inquiry dialogues, using Prakken's framework for persuasion dialogues defined in [8] and Black and Hunter's system for inquiry dialogue defined in [2] and [3] to examine persuasion and inquiry dialogues, respectively. Given the limited amount of research relating to inquiry dialogues, this similarity with persuasion dialogues might be used to show that many of the research relating to persuasion dialogues can also be applied to inquiry dialogues.

Black and Hunter's inquiry dialogue system [3] is a relatively new addition to the collection of dialogue games, specifically to the relatively small and infrequently expanded collection of articles discussing inquiry dialogues[2]. The first of Black and Hunter's articles dates from 2007 [2], and the most detailed article dates from 2009 [3]. As such, it warrants more attention, in order to find distinctive properties of this under-explored form of dialogue system.

In [8] Prakken proposes a framework to support the specification and study of formal properties of specific dialogue protocols for persuasion dialogues. Prakken examines properties such as strong or weak relevance, soundness and completeness, termination, dialogical status and the outcome of dialogues.

The aim of this thesis is to determine whether Prakken's framework is compatible with Black and Hunter's system, and at what cost possible incompatibilities can be resolved. By determining whether these systems are equal, more can be learned about inquiry dialogues in general and Black and Hunter's system for inquiry dialogues specifically. By doing this, additional insight into both dialogue types and the compatibility of Prakken's framework with different dialogue types.[3]

As an aside, following the norm in dialogue systems, both systems assume two parties in the dialogue. However, the system for inquiry dialogues is claimed to be easily extendable to encompass more than two parties [3]: "Many of the

[1]For example, from the point of view of a judge, a court case (which is generally considered a persuasion dialogue) could be considered an inquiry dialogue where each player is limited to attacking or defending the dialogue topic.

[2]See [1], [6] and [7]

[3]In [4] Prakken's framework is adapted to function for deliberation dialogues, but it requires extensive changes.

difficult issues associated with multiparty dialogues (...) can be easily overcome here due to the collaborative and exhaustive nature of the dialogues we are considering". This will be considered throughout the thesis.

Black and Hunter's articles have also provided a general agent strategy for inquiry dialogues, which guarantees that dialogues generated with this strategy are both complete and sound, benchmarked on the union of the belief bases of the agents. This means that inquiry dialogues constructed in this way are a reliable source of information if the agents that construct them are considered reliable, which opens the possibility of practical applications in which this reliability is required.

1.1 Primary aim

The primary goal of the thesis is to learn more about inquiry dialogues, by determining basic properties through compatibility with Prakken's framework and by using Prakken's dialogue framework to specify a dialogue system for inquiry that builds on Black and Hunter's work. Where possible, properties from both systems are made applicable to the adapted system.

What are the consequences of adapting Black and Hunter's inquiry dialogues to Prakken's persuasion framework, and vice versa, and what properties can we derive from this? What can we say about the system for inquiry dialogues on the basis of its similarities and differences with persuasion dialogues?

1.1.1 Sub-questions

1. Can Prakken's framework be applied to inquiry dialogues?
2. What modifications are necessary to allow inquiry dialogues to be translated to the framework and vice versa, and at what cost?
3. Which properties can be shown to hold for the (partially) adapted system(s), which can be excluded, and which cannot be determined?
4. Given the above, how does the system for inquiry dialogues compare to persuasion dialogues?
5. Given the above, what makes the system for inquiry dialogues significantly distinct from persuasion dialogues?

1.2 Outline

First, the original systems are described: Black and Hunter's system for inquiry dialogue in chapter 2 and Prakken's framework for persuasion in chapter 3. Unless mentioned specifically, definitions in the chapters describing Prakken's framework and Black and Hunter's system are from respectively [8] and [3].

Next, in order to determine the compatibility between Prakken's framework for persuasion dialogues and Black and Hunter's system for inquiry dialogues, a detailed mapping of definitions and properties is made, determining which properties are comparable, which will need adjusting and which are problematic.

The core of this thesis consists of two systems: one system is a version of Prakken's framework in which Black and Hunter's system for inquiry dialogues has been adapted to fit into Prakken's framework without any changes to the framework whatsoever. The other system works the other way around: Prakken's framework is adapted to Black and Hunter's system without any changes to Black and Hunter's system.

The two systems will show that a bisimulation between Prakken's system for persuasion dialogues and Black and Hunter's system for inquiry dialogues is possible, if only well-formed dialogue states are considered. A relation is a bisimulation iff it is a simulation of one system in a second system and its inverse is a simulation of the second system in the first system. For our system, this basically means that every well-formed (but not necessarily finished) dialogue that can be formulated with either system, can be converted to a well-formed dialogue in the other system.

To elaborate, the schematic below [4] displays consecutive well-formed dialogues in the different systems. A proof will be provided for the bidirectional arrows.

$$\begin{tikzcd}
\text{Prakken's framework} & \text{Adapted system} & \text{Black and Hunter's system} \\
d_i \arrow[r, leftrightarrow] \arrow[d] & D_1^x \arrow[r, leftrightarrow] \arrow[d] & D_1^p \arrow[d] \\
d_j \arrow[r, leftrightarrow] \arrow[d] & D_1^y \arrow[r, leftrightarrow] \arrow[d] & D_1^q \arrow[d] \\
d_k \arrow[r, leftrightarrow] & D_1^z \arrow[r, leftrightarrow] & D_1^r
\end{tikzcd}$$

From this, conclusions are drawn about the similarity of the two systems, and generalizations are made about Prakken's framework, Black and Hunter's system and persuasion and inquiry dialogues in general.

[4]Such that

d_i, d_j, d_k	are well-formed dialogues in Prakken's framework
$D_1^x, D_1^y, D_!^z$	are well-formed dialogues in the adapted system
$D_1^p, D_1^q, D_!^r$	are well-formed dialogues in Black and Hunter's system
	and $i < j < k$ and $p < q < r$ and $x < y < z$

2 Black and Hunter's Inquiry Dialogues

As was mentioned in the introduction, inquiry dialogues are a form of dialogue game that allow two agents to jointly increase their beliefs by constructing arguments or dialectical trees of arguments from beliefs that may not have been previously available to all of them.
Elizabeth Black and Anthony Hunter [2],[3] expand on this by defining argument and warrant inquiry dialogues.
In argument inquiry dialogues the goal of the players is to find all arguments they can to support a given topic, by cooperatively providing beliefs that may support the topic and supporting arguments for those beliefs.

Warrant inquiries, on the other hand, construct a dialectical tree for the topic. This tree contains arguments constructed by the agents during the dialogue. These arguments have been constructed from the combination of an agent's individual beliefs and arguments that have already been made. An argument in the tree counters arguments in the node above. Unlike in persuasion dialogues, there is no commitment in inquiry dialogues; an agent may assert an argument in one move and a counterargument against it in the next move. The root of the tree is labelled with the first argument that has been asserted with the topic of the dialogue as claim, and status of the argument at the root can be established by labeling the dialectical tree, determining whether the topic is warranted.

2.1 Beliefs and argumentation

DeLP [5] considers two kinds of program rules: defeasible rules to represent tentative knowledge and strict rules to represent strict knowledge. Using these, DeLP allows the construction of arguments for or against a topic. These arguments for and against a topic are represented in a dialectical tree, consisting of different argumentation lines with the topic as root node. Several definitions that will follow are definitions of concepts from DeLP, for example the defeasible derivation from Definition 5, arguments and subarguments from Definition 6 and the attack relation between arguments and argumentation lines from Definition 10.

Black and Hunter's system for inquiry dialogues is based on a dialogue with two participants, though the authors claim that it is straightforward to extend it to allow for more than two players due to its cooperative and exhaustive nature. All beliefs in the system, whether it is a belief about the state of the world (fact) or a belief about the behavior of the world (rule), is assumed to be defeasible; everything can be argued about and no fact is protected from being disputed. These beliefs are stored in an agent's belief base, with a preference level attached to each fact and each rule[5]. The lower the preference level, the

[5]Note that this means that agents can have different preference levels for beliefs, and may even use a different scale (i.e. 1,2,3,4 or 1,1,1,2 for their 4 strongest beliefs).

more preferred a fact or rule is.

Before the notions of Players, defeasible facts and rules and belief bases are formally defined, a few comments about notation need to be made. A restricted set of propositional logic is used for which:

- All beliefs are assumed to be defeasible, the sets of strict rules and facts are assumed to be empty.
- A **literal** is either an atom α or a negated atom $\neg\alpha$
- $\vdash$ is used to denote classical consequence relation
- $\perp$ is used to denote classical contradiction

Definition 1. **Players** in Black and Hunter's system for inquiry dialogues are elements of the set $\mathcal{I}$, a set containing the names of the players according to some naming convention, in this case numeric. In Black and Hunter's articles, $\mathcal{I} = \{1, 2\}$.

Definition 2. **Defeasible facts** are denoted α, α being a literal. **Defeasible rules** consist of a conjunction of literals α_i $(0 \leq i \leq n)$, implying a conclusion α_0:
$\alpha_1 \wedge \ldots \wedge \alpha_n \rightarrow \alpha_0$

Next, a measure is determined to compare the strength of the defeasible facts and rules. For this measure, a preference level is determined and defeasible facts and defeasible rules are paired with a preference level to form beliefs.

Definition 3. A **belief** b is a tuple (ϕ, L) containing the defeasible fact or defeasible rule ϕ and a **preference level** $L \in \mathbb{N}_1$. A function $pLevel((\phi, L)) = L$ is defined accordingly. If ϕ is a defeasible fact, b is called a **state belief**; if ϕ is a defeasible rule, b is called a **domain belief**.
Several sets are defined accordingly: The set of beliefs is denoted $\mathcal{B}$. $\mathcal{S}$ is the set of all state beliefs, with the related $\mathcal{S}^*$ denoting all defeasible facts: $\mathcal{S}^* = \{\phi \,|\, (\phi, L) \in \mathcal{S}\}$. Similarly, $\mathcal{R}$ is the set of all domain beliefs, with $\mathcal{R}^*$ denoting all defeasible rules: $\mathcal{R}^* = \{\phi \,|\, (\phi, L) \in \mathcal{R}\}$. Finally, the set $\mathcal{B}^*$ denotes all defeasible facts and rules: $\mathcal{B}^* = \{\phi \,|\, (\phi, L) \in \mathcal{B}\} = \mathcal{S}^* \cup \mathcal{R}^*$.

Definition 4. The **belief base** of an agent x is a finite set denoted as Σ^x, where $x \in \mathcal{I}$ and $\Sigma^x \subseteq \mathcal{B}$.

We will use a running example to illustrate the definitions. An example of a belief base is: $\{(a, 1)$, $(\neg a, 2)$, $(a \rightarrow b, 3)\}$ and $(\neg a \rightarrow \neg b, 2)$ [6]. Some of these beliefs can be combined, for example the rule starting with a and the fact a.

[6]which are two facts and two rules, respectively

One way to combine these beliefs is in a defeasible derivation.

Definition 5. A **defeasible derivation** of α (based on a belief base $\Psi \subseteq \mathcal{B}$) is a sequence of unique literals which defeasibly derive the last item of the sequence, denoted as $\Phi|\!\sim \phi$: $\forall \alpha_m \in \alpha_1, \ldots, \alpha_m, \ldots, \alpha_n$ either $(\alpha_m, L) \in \Psi$ or $(\beta_i \wedge \ldots \wedge \beta_j \rightarrow \alpha_m, L') \in \Psi$ and every literal in β_i $(1 \leq i \leq j)$ is an element α_k in the sequence $(k < m)$.

The defeasible derivations from the running example belief base are a, a, b, $\neg a$ and $\neg a, \neg b$. Now, arguments are defined; each is structured as a tuple, with the latter part being the conclusion of the argument (the last element of a defeasible derivation), and the first a minimal set of beliefs that are needed to derive the conclusion of the argument (the facts and rules).

Definition 6. An **argument** A constructed from Φ is a tuple $\langle \Phi, \phi \rangle$ with ϕ being the **claim** of the argument, a defeasible fact denoted by $Claim(A)$, and Φ being the **support** of the argument, a minimal set of premises that defeasibly derive ϕ, denoted by $Support(A)$. A **subargument** of A is an argument A_1 where $Support(A_1) \subseteq Support(A)$.
Φ must **defeasibly derive** ϕ and it must not defeasibly derive anything that conflicts with ϕ: $\forall \phi, \phi'$ if $\Phi| \sim \phi$ and $\Phi| \sim \phi'$ then $\phi \cup \phi' \nvdash \bot$.
The set of arguments that can be constructed from a belief base Ψ is denoted $\mathcal{A}(\Psi)$.

Definition 7. The **preference level of an argument** is defined as
$pLevel(A) =$

$$\begin{cases} pLevel(\phi) & \text{such that } \phi \in Support(A) \\ & \text{and} \\ & \quad \left[\begin{array}{l} \forall \phi' \in Support(A): \\ pLevel(\phi') \leq pLevel(\phi) \end{array} \right. \end{cases}$$

Given the previously defined running example belief base consisting of $(a, 1)$, $(\neg a, 2)$, $(a \rightarrow b, 3)$ and $(\neg a \rightarrow \neg b, 2)$, the following arguments can be constructed.

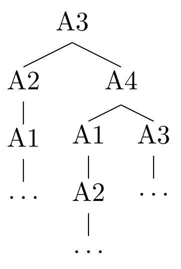
$$\begin{array}{ll} \text{A1} = & \langle \{(a, 1)\}, a \rangle \\ \text{A2} = & \langle \{(\neg a, 2)\}, \neg a \rangle \\ \text{A3} = & \langle \{(a, 1), (a \rightarrow b, 3)\}, b \rangle \\ \text{A4} = & \langle \{(\neg a, 2), (\neg a \rightarrow \neg b, 2)\}, \neg b \rangle \end{array}$$

Obviously, the above belief base contains conflicting fact and rules, but Black and Hunter allow this property for the belief base. Arguments, on the other hand, cannot be allowed to contain facts or rules that can lead to conflicting conclusions.

In the above list of arguments, several arguments have conflicting claims, for example A1 concludes a, but A2 concludes $\neg a$. These arguments could be used in the dialogue to attack each other or arguments that contain their conflicting counterparts.

Definition 8. An **attack relation** between two arguments A and A', or simply A attacks A', is defined as the attacker's claim conflicting with either the claim of the object of the attack, or one of the subarguments in the object of the attack:
A and A' **conflict** iff $Claim(A) \cup Claim(A') \vdash \bot$. If A conflicts with a subargument A'_1 of A' then A attacks A' at subargument A'_1. Note that if A and A' conflict, A attacks A' at A', and vice versa.

In the running example, A1 attacks A4 at A2, A2 attacks A3 at A1 and A3 attacks A4 at A4 (and the converse, A4 attacks A3 at A3). Of course, A1 and A2 attack each other as well.



The preference level determines whether an attack is successful. As was mentioned earlier, the lower the preference level, the more preferred a fact or rule is. For arguments, the preference level is determined to be the preference level of the least preferred (the highest value) of the beliefs supporting it. Given arguments ϕ and ψ, where ϕ attacks ψ, if the attacker is more preferred than the argument it is attacking, the attack succeeds and ϕ is a *proper defeater* for ψ. If the attacker has a higher preference level, ψ is preferred over its attacker and ϕ will not be able to defeat it. If the preference levels are equal, ϕ is a *blocking defeater* for ψ and vice versa.

Definition 9. Given arguments A, B and SB, with SB a subargument of B, where A attacks B at SB, if $pLevel(A) \leq pLevel(SB)$ then A **defeats** B, specifically if $pLevel(A) < pLevel(B)$ then A is a **proper defeater** of B, otherwise A is a **blocking defeater** of B and vice versa.

An attacker may itself also be attacked. If it is defeated by an attacker of its own, it should not be able to defeat any arguments, until its attackers are defeated. This means that in order to determine whether an attacker is truly successful, we must not only consider the preference level compared to its victim, but we must also evaluate all arguments that successfully defeat the attacker. Of course, to consider the attacker's defeaters, we must consider their defeaters, and so on.

Definition 10. An **argumentation line** is a sequence of arguments $[\langle\Phi_0, \phi_0\rangle, \langle\Phi_1, \phi_1\rangle, \ldots]$ such that each element $\langle\Phi_i, \phi_i\rangle$ is a defeater of its predecessor $\langle\Phi_{i-1}, \phi_{i-1}\rangle$.

Not any sequence of arguments is an acceptable sequence; Black and Hunter use the conditions for argumentation lines are defined in DeLP:

- It must be finite,
- The set of all odd and the set of all even elements must be consistent,
- Arguments cannot be used to attack an attacker of an argument that contains that argument as a subargument (irrespective of the distance in the argumentation line),
- No two blocking defeaters may follow each other in the argumentation line.

Definition 11. An argumentation line Λ is an **acceptable argumentation line** if it adheres to the following conditions:

- it is finite
- $\left[\bigcup_{0\leq i\leq \frac{L}{2}} \Phi_{2i}\right] \nvdash \perp$ and $\left[\bigcup_{1\leq i\leq \frac{L}{2}} \Phi_{2i-1}\right] \nvdash \perp$ where $L =$ the length of the argumentation line.
- $\forall\langle\Phi_k, \phi_k\rangle \in \Lambda \nexists\langle\Phi_j, \phi_j\rangle \in \Lambda$ in the same line where $j < k$ and $\langle\Phi_k, \phi_k\rangle$ is a subargument of $\langle\Phi_j, \phi_j\rangle$
- $\forall\langle\Phi_i, \phi_i\rangle \in \Lambda$ if $\langle\Phi_i, \phi_i\rangle$ is a blocking defeater of $\langle\Phi_{i-1}, \phi_{i-1}\rangle$ then $\langle\Phi_{i+1}, \phi_{i+1}\rangle$ must be a proper defeater of $\langle\Phi_i, \phi_i\rangle$ or $\langle\Phi_{i+1}, \phi_{i+1}\rangle \notin \Lambda$.

Combining all the argumentation lines starting with the same argument, produces a branching tree of arguments: the dialectical tree. Every path from the root to a leaf is an argumentation line, ending with leaves containing defeaters which are not defeated by any arguments. The dialectical tree is a tree representing all the defeat relations of a set of arguments.

Definition 12. Let Ψ be a, possibly inconsistent, belief base and A_0 be an argument such that $A_0 \in \mathcal{A}(\Psi)$. 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 $\Lambda_i = [A_0, \ldots, A_n]$ be the sequence of labels on the path from the root to node N. Let arguments $B_1, B_2, \ldots, B_k$ be all the defeaters for A_n that can be formed from Ψ.
   For each defeater B_j $(1 \leq j \leq k)$, if the argumentation line $\Lambda'_i = [A_0, \ldots, A_n, B_j]$ 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 Λ'_i is acceptable, then N is a leaf node.

Two **dialectical trees are equal** if the set of argumentation lines that can be read from the first tree is identical to the set of argumentation lines that can be read from the second tree.

In our running example, this would only be [A3, A4, A1]; the branch starting with A2 does not exist here, since A2 is weaker than A1.

$$\begin{tikzcd}
\text{A3} \arrow[d, dash] \\
\text{A4} \arrow[d, dash] \\
\text{A1}
\end{tikzcd}$$

We can examine the results when A1 and A2 would have had an identical preference level. Had A1 and A2 had the same preference level, they would have been able to keep repeating their attacks on each other, defeating one another every time, if the conditions for an acceptable argumentation line were not observed. A2 would have been able to defeat A3, and A1 could not be used to defend against it, for it would conflict with the third and fourth restriction on acceptable argumentation lines.

$$\begin{tikzcd}
& \text{A3} \arrow[dl, dash] \arrow[dr, dash] & \\
\text{A2} & & \text{A4} \arrow[d, dash] \\
& & \text{A1}
\end{tikzcd}$$

The tree can now be used to determine whether the claim made by the argument at the root of the tree is warranted or not. In order to determine this, leaf nodes in the tree are considered undefeated, and any node defeated by an undefeated node is considered to be defeated. If all the defeaters of a node are defeated, the node is undefeated. The status of the root node of the tree will then determine whether the argument on the root node is warranted.

Definition 13. The **marked dialectical tree** $MT(A_0, \Psi)$ is created from the dialectical tree $T(A_0, \Psi)$ as follows[7]:

- A node N of $T(A_0, \Psi)$ is marked undefeated when $\forall N'$ if N' is a child of N then N' has been marked as defeated.
- The marking starts at the leaf nodes, which are automatically undefeated as they have no children, and move upwards until the root has been marked.

The **status of an argument** is determined by using a marked dialectical tree: $Status(A, \Psi) = U$ iff the root of the tree $MT(A, \Psi)$ is undefeated, otherwise, $Status(A, \Psi) = D$. An argument with status U is warranted.

For example, if the tree from the running example was to be marked, it would look like this:

$$\begin{tikzcd}
A3\ \ U \arrow[d, dash] \\
A4\ \ D \arrow[d, dash] \\
A1\ \ U
\end{tikzcd}$$

And the altered example would look like this:

$$\begin{tikzcd}
 & A3\ \ D \arrow[dl, dash] \arrow[dr, dash] & \\
A2\ \ U & & A4\ \ D \arrow[d, dash] \\
 & & A1\ \ U
\end{tikzcd}$$

2.2 Dialogues

A dialogue consists of alternate moves between the players. Unlike many dialogue systems, Black and Hunter's inquiry dialogues use few possible moves. The player can either *open* or *close* warrant and argument inquiry dialogues, though a warrant inquiry dialogue can only be opened in the first move of a dialogue, or *assert* an argument.

Definition 14. A **move** is a tuple $\langle Agent, Act, Content\rangle$, such that $Sender(\langle Agent, Act, Content\rangle) = Agent$ and $Sender(\langle Agent, Act, Content\rangle) \in \mathcal{I}$.

Moves in Black and Hunter's system for inquiry are defined as

[7]In the original article, both the marked and normal dialectical tree are denoted $T(A_0, \Psi)$

Move	Format
open	$\langle x, open, dialogue(\theta, \gamma)\rangle$
assert	$\langle x, assert, \langle \Phi, \phi\rangle\rangle$
close	$\langle x, close, dialogue(\theta, \gamma)\rangle$

where
$x \in \mathcal{I}$
and
$\theta = wi$ and $\gamma \in \mathcal{S}^*$
or
$\theta = ai$ and $\gamma \in \mathcal{R}^*$

The set of moves is defined as $\mathcal{M}$.

An *open* move determines the type of dialogue and the topic to discuss. If the dialogue is chosen to be a warrant inquiry dialogue, the topic must be a defeasible fact; if it is chosen to be an argument inquiry dialogue, the topic must be a defeasible rule. Similarly, a *close* move contains the same information, in order to determine which dialogue or subdialogue the player proposes to *close* . Only once all players propose to *close* a dialogue, with no other moves in between, a so-called *matched close* , will it be terminated.[8]

Definition 15. A **dialogue** in Black and Hunter's system is denoted as D_r^t where $r, t \in \mathbb{N}_1$ and $r \leq t$. D_r^t is a sequence of moves $m_r, \ldots, m_t$. The first move of a dialogue D_r^t, move $m_r = \langle x, open, dialogue(\theta, \gamma)\rangle$, is always an *open* move, as this move determines both the **type** ($Type(D_r^t) = \theta$) and **topic** ($Topic(D_r^t) = \gamma$) of the dialogue, as defined in Definition 14. The **participants of the dialogue** are the elements of the set of players $\mathcal{I}$ defined in Definition 1[9].

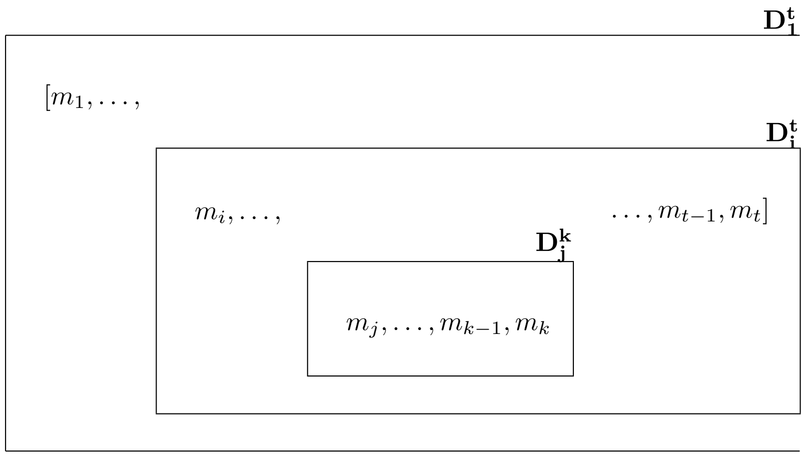
The set of all dialogues is defined as $\mathcal{D}$.

A player may also embed an argument inquiry dialogue inside an argument or warrant inquiry dialogue, if no dialogue with the same topic has been opened previously and its topic may have influence on the current dialogue. It is not possible to embed warrant inquiry dialogues inside either warrant or argument inquiry dialogues.

Since r marks the start move of a dialogue D_r^t, a subdialogue can easily be defined within a dialogue:

[8]The original definition contained turntaking restrictions here and in the well-formed dialogue definitions. These have been omitted here; having the turntaking restriction inside the dialogue definition would require it to be redefined to be used in later chapters where turntaking is a separate function.

[9]The original article defines this as $\{1, 2\}$, which is omitted here. Whenever the original definitions refer to player 1 and player 2, a reference to the elements of the set $\mathcal{I}$ is used instead.



$$1 < i < j < k < t-1$$
$$D_1^t = [m_1, \ldots, m_i, \ldots, m_j, \ldots, m_k, m_{k+1}, \ldots, m_{t-1}, m_t]$$
$$m_1 = \langle P_1, open, dialogue(\theta_1, \phi_1)\rangle \qquad m_i = \langle P_i, open, dialogue(\theta_i, \phi_i)\rangle$$
$$m_j = \langle P_j, open, dialogue(\theta_j, \phi_j)\rangle \qquad m_{k-1} = \langle P_{k-1}, close, dialogue(\theta_j, \phi_j)\rangle$$
$$m_k = \langle P_k, close, dialogue(\theta_j, \phi_j)\rangle \qquad m_{t-1} = \langle P_{t-1}, close, dialogue(\theta_i, \phi_i)\rangle$$
$$m_t = \langle P_t, close, dialogue(\theta_i, \phi_i)\rangle$$

Figure 3: Schematic of nested dialogues from [3]. From the original caption: D_1^t is a top level dialogue that has not yet terminated. D_i^t is a sub-dialogue of D_1^t that terminates at t. D_j^k is a sub-dialogue of both D_1^t and D_i^t, that terminates at k. D_1^t is a top-dialogue of D_1^t. D_1^k is a top-dialogue of D_j^k. D_1^t is a top-dialogue of D_i^t. D_1^k is a top-dialogue of D_1^k.

Definition 16. A dialogue D_u^v is a **subdialogue** of D_r^t if $r < u \leq v \leq t$ and D_u^v is a subsequence of D_r^t. Conversely, a **top-level dialogue** is a dialogue which is not a subdialogue of anything: D_r^t is a top-level dialogue if $r = 1$. The set of top-level dialogues is denoted $\mathcal{D}_{top}$. D_1^t is a **top-dialogue** of D_r^t iff they are equal or D_r^t is a sub-dialogue of D_1^t. A dialogue D_r^t **extends** D_u^v iff the first n moves in D_r^t are the sequence D_u^v, and D_u^v is n moves long.

An illustration to elaborate on the sub-dialogue structure from [3] is shown in Figure 3.

Next, close moves and termination are discussed.

Definition 17. Given a dialogue D_r^t and participants $\mathcal{I} = \{1, 2\}$, a **matched close** for D_r^t occurs at t when $m_u = \langle 1, close, dialogue(Type(D_r^t), Topic(D_r^t))\rangle$ and $m_v = \langle 2, close, dialogue(Type(D_r^t), Topic(D_r^t))\rangle$ have been moved, where

$u, v \leq t$ and $u = v - 1$ or $v = u - 1$ and $u = t$ or $v = t$.

Definition 18. A dialogue D^t_r terminates at t if

- m_t is a matched close
- there was no earlier termination: $\nexists D^v_u$ where $r = u$ and D^v_u terminates at v and D^t_r extends D^v_u
- $\forall D^v_u$ if D^v_u is a subdialogue of D^t_r then $\exists D^w_u$ such that D^w_u terminates at w, is a subdialogue of D^t_r and either D^w_u extends D^v_u or vice versa.

Definition 19. Let D^t_r be a dialogue. The **current dialogue** is given by $\mathsf{Current}(D^t_r)$ such that $\mathsf{Current}(D^t_r) = D^t_{r_1}$ $(1 \leq r \leq r_1 \leq t)$ where the following conditions hold:

1. $m_{r_1} = \langle x, open, dialogue(\theta, \gamma)\rangle$ for some $x \in \mathcal{I}$, some $\gamma \in \mathcal{B}^\star$ and some $\theta \in \{wi, ai\}$,
2. $\forall D^{t_1}_{r_2}$ if $D^{t_1}_{r_2}$ is a sub-dialogue of $D^t_{r_1}$,
   then $\exists D^{t_2}_{r_2}$ s.t. either $D^{t_2}_{r_2}$ extends $D^{t_1}_{r_2}$ or
   $D^{t_1}_{r_2}$ extends $D^{t_2}_{r_2}$,
   and $D^{t_2}_{r_2}$ is a sub-dialogue of $D^t_{r_1}$
   and $D^{t_2}_{r_2}$ terminates at t_2,
3. $\nexists D^{t_3}_{r_1}$ s.t. $D^t_{r_1}$ extends $D^{t_3}_{r_1}$ and $D^{t_3}_{r_1}$ terminates at t_3.

If the above conditions do not hold then $\mathsf{Current}(D^t_r) = null$.

The commitments of the player are stored in a commitment store. Paradoxically, this is not a set of beliefs that the agents has publicly committed to and cannot deviate from, as in most dialogue games. It is rather a set of the rules and facts asserted by this agent, and may be conflicting, as noted on page 12. Players may use the commitment store of the other players to augment their belief base, using beliefs from both without restriction when constructing an argument.

Definition 20. The **commitment store** is denoted $CS^t_x \subseteq \mathcal{B}$, where $x \in \mathcal{I}$ is an agent and $t \in \mathbb{N}_1$ is a point in time. The content of the commitment store is updated with the following function:

$$CS^t_x = \begin{cases} \emptyset & \text{iff } t = 0 \\ CS^{t-1}_x \cup \Phi & \text{iff } m_t = \langle x, assert, \langle \Phi, \phi\rangle\rangle \\ CS^{t-1}_x & \text{otherwise.} \end{cases}$$

2.2.1 Argument inquiry dialogue

Arguments inquiry dialogues are used to jointly construct arguments for a particular claim, by using the beliefs of each agent when it is applicable. A query store contains the literals that support the claim, and the agents will try to find support for these literals as well. They do not, however, consider counterarguments (nor counter-counterarguments, and so on).

Definition 21. The **query store** of an argument inquiry dialogue D_r^t is defined as

$$QS_r = \begin{cases} \{\alpha_1, \ldots, \alpha_n, \beta\} & \text{iff } m_r = \langle x, open, dialogue(ai, \alpha_1 \wedge \ldots \alpha_n \rightarrow \beta)\rangle \\ \emptyset & \text{otherwise.} \end{cases}$$

The query store of the current dialogue is determined by the function $cQS(D_r^t) = QS_u$ iff $Current(D_r^t) = D_u^t$.

A protocol is proposed for players in the dialogue. They may propose to *close* dialogues at any time they wish, though this needs a *matched close* to actually cause the dialogue to terminate. They may *assert* any argument they like as long as the conclusion of the argument is in the query store, ensuring that arguments are relevant to the dialogue, and it has not been asserted before.

They may also *open* an argument inquiry dialogue inside another inquiry dialogue. This allows them to focus the discussion on a specific argument, using an argument inquiry dialogue's query store to jointly provide arguments to support the content of the argument that is the topic of the dialogue. These cannot be opened multiple times for the same claim.

Definition 22. The **argument inquiry protocol** Π_{ai} is a function of type $\Pi_{ai} : \mathcal{D}_{top} \mapsto \mathcal{P}(\mathcal{M})$. Given a top-level dialogue D_1^t where $1 \leq t$ and $Topic(Current(D_1^t)) = \gamma$, the definition for $\Pi_{ai}(D_1^t) = \Pi_{ai}^{assert}(D_1^t) \cup \Pi_{ai}^{open}(D_1^t) \cup \{\langle x, close, dialogue(ai, \gamma)\rangle\}$, such that

$$\Pi_{ai}^{assert}(D_1^t) = \left\{ \langle x, assert, \langle \Phi, \phi \rangle\rangle \;\middle|\; \begin{array}{l} \phi \in cQS(D_r^t) \text{ and} \\ x \neq Sender(m_t) \text{ and} \\ \nexists v \text{ where} \\ \quad \left[\begin{array}{l} 1 < v \leq t \text{ and} \\ m_v = \langle x', assert, \langle \Phi, \phi \rangle\rangle \\ \text{and } x' \in \mathcal{I} \end{array} \right. \end{array} \right\}$$

$\Pi_{ai}^{open}(D_1^t) =$

$$\left\{ \langle x, open, dialogue(ai, \gamma)\rangle \;\middle|\; \begin{array}{l} \gamma = \beta_1 \land \ldots \land \beta_n \rightarrow \alpha \text{ and} \\ \alpha \in cQS(D_1^t) \text{ and} \\ x \neq Sender(m_t) \text{ and} \\ \nexists v \text{ where} \\ \quad \left[\begin{array}{l} 1 < v \leq t \text{ and} \\ m_v = \langle x', open, dialogue(ai, \gamma)\rangle \\ \text{and } x' \in \mathcal{I} \end{array} \right. \end{array} \right\}$$

Note that turntaking is defined in the protocol.

A dialogue is a well-formed argument inquiry dialogue if it meets the following criteria: it must start with an *open* that opens an argument inquiry dialogue, it must terminate at some point (though this may be at a point later than t) and every move made in the dialogue must be legal according to the protocol.

Definition 23. Dialogue D_r^t is a **well-formed argument inquiry dialogue** iff

- $m_r = \langle x, open, dialogue(ai, \gamma)\rangle$ where $x \in \mathcal{I}$ and $\gamma \in \mathcal{R}^*$
- $\exists v$ such that $t \leq v$, D_r^v extends D_r^t and D_r^v terminates at v
- $\forall s$ where $r \leq s < t$ and D_r^t extends D_r^s
  if
  $\left[\begin{array}{l} D_1^t \text{ is a top-dialogue of } D_r^t \text{ and} \\ D_1^s \text{ is a top-dialogue of } D_r^s \text{ and} \\ D_1^t \text{ extends } D_1^s \end{array} \right.$
  then
  $\left[\begin{array}{l} Sender(m_s), Sender(m_{s+1}) \in \mathcal{I}, \\ Sender(m_s) \neq Sender(m_{s+1}) \text{ and} \\ m_{s+1} \in \Pi_{ai}(D_1^s) \end{array} \right.$

If the dialogue is well-formed, the **argument inquiry outcome** is defined as

$$Outcome_{ai}(D_r^t) = \{\langle \Phi, \phi\rangle \in \mathcal{A}(\bigcup_{x \in \mathcal{I}} CS_x^t) | \phi \in QS_r\}$$

To summarise, the outcome of an argument inquiry dialogue is a set of arguments with the topic of the dialogue as conclusion, which is only available if the dialogue is well-formed. A protocol was defined to give the possible moves in a certain situation.

2.2.2 Warrant inquiry dialogue

Unlike the argument inquiry dialogue, a warrant inquiry dialogue aims to consider relevant counterarguments in order to determine the validity of a claim. It does this by constructing a dialectical tree of the topic of the dialogue, with the first asserted argument at the root of the tree. If, at the end of the dialogue, the root is undefeated by its children, the claim is said to be warranted.

Any belief asserted by the agents must be relevant: it must make some change to the dialectical tree; if the argument does not change the tree at all, there is no reason for it to be stated. As in argument inquiry dialogues, agents may propose to *close* the dialogue at any time.

Definition 24. The **warrant inquiry protocol** Π_{wi} is a function of type $\Pi_{wi} : \mathcal{D}_{top} \mapsto \mathcal{P}(\mathcal{M})$. Given a top-level dialogue D_1^t where $1 \leq t$ and $Topic(D_1^t) = \gamma$, the definition for $\Pi_{wi}(D_1^t) = \Pi_{wi}^{assert}(D_1^t) \cup \Pi_{wi}^{open}(D_1^t) \cup \{\langle x, \textit{close}, \textit{dialogue}\ (wi, \gamma)\rangle\}$ $(x \in \mathcal{I})$, such that

A $RootArg(D_1^t)$ function is defined as:
$RootArg(D_1^t) =$

$$\begin{cases} \langle \Gamma, \gamma \rangle & \text{if } \exists s \text{ such that} \\ & \quad \left[\begin{array}{l} 1 < s \leq t \text{ and} \\ m_s = \langle x, assert, \langle \Gamma, \gamma \rangle \rangle \text{ and} \\ Topic(D_1^t) = \gamma \text{ and } x \in \mathcal{I} \end{array} \right. \\ & \text{and } \nexists u, \Gamma' \text{ such that} \\ & \quad \left[\begin{array}{l} 1 < u < s \text{ and} \\ m_u = \langle x', assert, \langle \Gamma', \gamma \rangle \rangle \text{ and} \\ x' \in \mathcal{I} \end{array} \right. \\ null & \text{otherwise} \end{cases}$$

$\Pi_{wi}^{assert}(D_1^t) =$

$$\left\{ \langle x, assert, \langle \Phi, \phi \rangle \rangle \;\middle|\; \begin{array}{l} x \neq Sender(m_t) \text{ and} \\ \quad \left[\begin{array}{l} \text{either } RootArg(D_1^t) = null \text{ and } \phi = \gamma \text{ or} \\ \quad \left[\begin{array}{l} T(RootArg(D_1^t), \bigcup_{i \in \mathcal{I}} CS_i^t \cup \Phi) \\ \neq \\ T(RootArg(D_1^t), \bigcup_{i \in \mathcal{I}} CS_i^t) \end{array} \right. \end{array} \right. \\ \text{and} \\ \quad \left[\begin{array}{l} \nexists u \text{ such that} \\ \quad \left[\begin{array}{l} 1 < u \leq t \text{ and} \\ m_u = \langle x', assert, \langle \Phi, \phi \rangle \rangle \\ \text{and } x' \in \mathcal{I} \end{array} \right. \end{array} \right. \end{array} \right\}$$

$\Pi_{wi}^{open}(D_1^t) =$

$$\left\{ \begin{array}{l|l} \langle x, open, dialogue(ai, \gamma_1)\rangle & x \neq Sender(m_t) \text{ and} \\ & \gamma_1 = \beta_1 \wedge \ldots \wedge \beta_n \rightarrow \alpha \text{ and} \\ & \quad \left[\begin{array}{l} \text{either } RootArg(D_1^t) = null \\ \text{and } \alpha = \gamma \text{ or} \\ \exists \Psi \subseteq \bigcup_{i \in \mathcal{I}} CS_i^t \text{ such that } \Psi | \sim \neg\alpha \end{array} \right. \\ & \text{and } \nexists u \text{ such that} \\ & \quad \left[\begin{array}{l} 1 < u \leq t \text{ and} \\ m_u = \langle x', open, dialogue(ai, \gamma_1)\rangle \\ \text{and } x' \in \mathcal{I} \end{array} \right. \end{array} \right\}$$

Note that turntaking is defined in the protocol.

A dialogue D_r^t is a well-formed warrant inquiry dialogue if it meets the following criteria: it must start with an open move, it must terminate at some point (though this may be at a point later than t) and every move made in the dialogue must be legal according to the protocol.

Definition 25. A dialogue D_r^t is a **well-formed warrant inquiry dialogue** iff

- $m_r = \langle x, open, dialogue(wi, \gamma)\rangle$ where $x \in \mathcal{I}$ and $\gamma \in \mathcal{S}^*$
- $\exists v$ such that $t \leq v$, D_r^v extends D_r^t and D_r^v terminates at v
- $\forall s$ where $r \leq s < t$ and D_r^t extends D_r^s
  if
  $$\left[\begin{array}{l} D_1^t \text{ is a top-dialogue of } D_r^t \text{ and} \\ D_1^s \text{ is a top-dialogue of } D_r^s \text{ and} \\ D_1^t \text{ extends } D_1^s \end{array} \right.$$
  then
  $$\left[\begin{array}{l} Sender(m_s), Sender(m_{s+1}) \in \mathcal{I}, \\ Sender(m_s) \neq Sender(m_{s+1}) \\ \text{ and } m_{s+1} \in \Pi_\theta(D_1^s) \text{ where } \theta = Type(Current(D_1^s)) \end{array} \right.$$

If the dialogue is well-formed, the **warrant inquiry outcome** is defined as

$$Outcome_{wi}(D_r^t) = \left\{ \begin{array}{ll} RootArg(D_r^t) & \text{iff } Status(RootArg(D_r^t), \bigcup_{x \in \mathcal{I}} CS_x^t) = U \\ \emptyset & \text{iff } RootArg(D_r^t) = null \text{ or} \\ & Status(RootArg(D_r^t), \bigcup_{x \in \mathcal{I}} CS_x^t) = D \end{array} \right.$$

2.2.3 Agent strategies

While the above agent protocols constrain which moves the agents are allowed to make during the dialogue, they do not define agent behaviour. The choices agents make during a dialogue are defined in a strategy. The strategy deterministically defines which move to make at any given moment. An exhaustive strategy is given, which ensures that all moves which might have an influence on the outcome of the dialogue are made.

The strategy is as follows: if there are legal *assert* moves that can be made, make one of these moves, as selected by a deterministic function. Otherwise, if there are *open* moves that can be made, make one of those, also selected by a deterministic function. If neither *open* moves or *assert* moves can be made, make a move to *close* the dialogue.

Definition 26. The **exhaustive strategy** is defined as follows:

$$\Omega_x(D_1^t) = \begin{cases} Pick_a(Asserts_x(D_1^t)) & \text{iff } Asserts_x(D_1^t) \neq \emptyset \\ Pick_o(Opens_x(D_1^t)) & \text{iff } Asserts_x(D_1^t) = \emptyset \text{ and } Opens_x(D_1^t) \neq \emptyset \\ \langle x, close, dialogue(\theta, \gamma) \rangle & \text{iff } Asserts_x(D_1^t) = \emptyset \text{ and } Opens_x(D_1^t) = \emptyset \end{cases}$$

$$\text{where} \left\| \begin{array}{l} \theta = Type(Current(D_1^t)) \\ \gamma = Topic(Current(D_1^t)) \\ Asserts_x(D_1^t) = \\ \quad \left\{ \langle x, assert, \langle \Phi, \phi \rangle \rangle \in \Pi_\theta(D_1^t) \;\middle|\; \langle \Phi, \phi \rangle \in \mathcal{A}(\bigcup_{i \in \mathcal{I} \setminus x} CS_i^t \cup \Sigma^x) \right\} \\ Opens_x(D_1^t) = \\ \quad \left\{ \langle x, open, dialogue(ai, \psi) \rangle \in \Pi_\theta(D_1^t) \;\middle|\; (\psi, L) \in \Sigma^x \right\} \end{array} \right.$$

Definition 27. Given a $\Theta = \{\langle x, assert, \langle \Phi_1, \phi_1 \rangle \rangle, \ldots \langle x, assert, \langle \Phi_k, \phi_k \rangle \rangle\}$, **Pick**$_a$ is defined as $Pick_a(\Theta) = \langle x, assert, \langle \Phi_i, \phi_i \rangle \rangle$ where $1 \leq i \leq k$ and $\forall j$ where $1 \leq j \leq k$ and $i \neq j$, $\lambda(\langle \Phi_i, \phi_i \rangle) <_{lex} \lambda(\langle \Phi_j, \phi_j \rangle)$. Similarly, given a Θ of k open moves **Pick**$_o$ is defined as $Pick_o(\Theta) = \langle x, open, dialogue(\theta_i, \phi_i) \rangle$ where $1 \leq i \leq k$ and $\forall j$ where $1 \leq j \leq k$ and $i \neq j$, $\mu(\phi_i) < \mu(\phi_j)$.

The μ and λ functions are explained using a quote from the original article:

> Let us assume that $\mathcal{B}^\star$ is composed of a finite number Z of atoms. Let us also assume that there is a registration function μ over these atoms: so, for a literal α, $\mu(\alpha)$ returns a unique single digit number base Z (this number is only like an id number and can be arbitrarily assigned). For a rule $\alpha_1 \wedge \ldots \wedge \alpha_n \rightarrow \alpha_{n+1}$, $\mu(\alpha_1 \wedge \ldots \wedge \alpha_n \rightarrow \alpha_{n+1})$ is an $n+1$ digit number of the form $\mu(\alpha_1) \ldots \mu(\alpha_n)\mu(\alpha_{n+1})$. This gives a unique base Z number for each formula in $\mathcal{B}^\star$ and allows an agent to select a single open move using the natural ordering relation $<$ over base Z numbers. [from:[2]]
>
> The function λ returns a unique tuple of base Z numbers for each argument. We use a standard lexicographical comparison, denoted

$\prec_{lex}$, of these tuples of numbers to select a move to make (i.e. the one whose content is the maximum element in the lexicographical ordering). [from:[2]]

2.3 Examples

Both examples are taken from [3]. The example argument inquiry dialogue in Table 1 is based on the belief bases:

Σ^1	$\{(c \rightarrow d, 1), (b \rightarrow c, 1), (a \rightarrow b, 1)\}$
Σ^2	$\{(a, 1), (b, 1)\}$

t	CS_1^t	m_t	CS_2^t	QS_t
1		$\langle 1, open, dialogue(ai, c \rightarrow d)\rangle$		$QS_1 = \{c, d\}$
2		$\langle 2, close, dialogue(ai, c \rightarrow d)\rangle$		
3		$\langle 1, open, dialogue(ai, b \rightarrow c)\rangle$		$QS_3 = \{b, c\}$
4		$\langle 2, assert, \langle\{(b, 1)\}, b\rangle\rangle$	$(b, 1)$	
5	$(b, 1)$ $(b \rightarrow c, 1)$	$\langle 1, assert, \langle\{(b, 1), (b \rightarrow c, 1)\}, c\rangle\rangle$		
6		$\langle 2, close, dialogue(ai, b \rightarrow c)\rangle$		
7		$\langle 1, open, dialogue(ai, a \rightarrow b)\rangle$		$QS_7 = \{a, b\}$
8		$\langle 2, assert, \langle\{(a, 1)\}, a\rangle\rangle$	$(a, 1)$	
9	$(a, 1)$ $(a \rightarrow b, 1)$	$\langle 1, assert, \langle\{(a, 1), (a \rightarrow b, 1)\}, b\rangle\rangle$		
10		$\langle 2, close, dialogue(ai, a \rightarrow b)\rangle$		
11		$\langle 1, close, dialogue(ai, a \rightarrow b)\rangle$		
12		$\langle 2, close, dialogue(ai, b \rightarrow c)\rangle$		
13		$\langle 1, close, dialogue(ai, b \rightarrow c)\rangle$		
14		$\langle 2, close, dialogue(ai, c \rightarrow d)\rangle$		
15	$(c \rightarrow d, 1)$	$\langle 1, assert, \langle\{(a, 1), (a \rightarrow b, 1), (b \rightarrow c, 1), (c \rightarrow d, 1)\}, d\rangle\rangle$		
16		$\langle 2, close, dialogue(ai, c \rightarrow d)\rangle$		
17		$\langle 1, close, dialogue(ai, c \rightarrow d)\rangle$		

Table 1: Argument inquiry dialogue example.

Move 15 contains the result of the argument inquiry dialogue displayed above: $(\{(a, 1), (a \rightarrow b, 1), (b \rightarrow c, 1), (c \rightarrow d, 1)\}, d)$.

The example warrant inquiry dialogue in Table 2 is based on the belief bases:

Σ^1	$\{(a,4),(a \to b,4),(c,3),(c \to \neg b,3),(e,2)\}$
Σ^2	$\{(d,3),(d \to \neg a,3),(\neg d,1),(e \to \neg d,2),(\neg e,1)\}$

t	CS_1^t	m_t	CS_2^t	QS_t
1		$\langle 1, open, dialogue(wi, b)\rangle$		
2		$\langle 2, close, dialogue(wi, b)\rangle$		
3	$(a,4)$ $(a \to b,4)$	$\langle 1, assert, \langle\{(a,4),(a \to b,4)\}, b\rangle\rangle$		
4		$\langle 2, assert, \langle\{(d,3),(d \to \neg a,3)\}, \neg a\rangle\rangle$	$(d,3)$ $(d \to \neg a,3)$	
5	$(c,3)$ $(c \to \neg b,3)$	$\langle 1, assert, \langle\{(c,3),(c \to \neg b,3)\}, \neg b\rangle\rangle$		
6		$\langle 2, assert, \langle\{(\neg d,1)\}, \neg d\rangle\rangle$	$(\neg d,1)$	
7		$\langle 1, open, dialogue(ai, a \to b)\rangle$		$QS_7 = \{a, b\}$
8		$\langle 2, close, dialogue(ai, a \to b)\rangle$		
9		$\langle 1, close, dialogue(ai, a \to b)\rangle$		
10		$\langle 2, open, dialogue(ai, d \to \neg a)\rangle$		$QS_{10} = \{d, \neg a\}$
11		$\langle 1, close, dialogue(ai, d \to \neg a)\rangle$		
12		$\langle 2, close, dialogue(ai, d \to \neg a)\rangle$		
13		$\langle 1, open, dialogue(ai, c \to \neg b)\rangle$		$QS_{13} = \{c, \neg b\}$
14		$\langle 2, close, dialogue(ai, c \to \neg b)\rangle$		
15		$\langle 1, close, dialogue(ai, c \to \neg b)\rangle$		
16		$\langle 2, open, dialogue(ai, e \to \neg d)\rangle$		$QS_{16} = \{e, \neg d\}$
17	$(e,2)$	$\langle 1, assert, \langle\{(e,2)\}, e\rangle\rangle$		
18		$\langle 2, assert, \langle\{(e,2),(e \to \neg d,2)\}, \neg d\rangle\rangle$	$(e,2)$ $(e \to \neg d,2)$	
19		$\langle 1, close, dialogue(ai, e \to \neg d)\rangle$		
20		$\langle 2, close, dialogue(ai, e \to \neg d)\rangle$		
21		$\langle 1, close, dialogue(wi, b)\rangle$		
22		$\langle 2, assert, \langle\{(\neg e,1)\}, \neg e\rangle\rangle$	$(\neg e,1)$	
22		$\langle 1, close, dialogue(wi, b)\rangle$		
23		$\langle 2, close, dialogue(wi, b)\rangle$		

Table 2: Warrant inquiry dialogue example.

The outcome of the warrant inquiry dialogue displayed above is $\emptyset$, as the argument supporting b in move 3 is ultimately defeated by the argument made in move 5.

2.4 Privacy issue

One might wonder why a system for inquiry dialogues might be used, when the pooling of knowledge seems to be the most simple solution. The difference is in the sharing of information. As it is highly unlikely that all information from both agents is required to determine the validity of a topic, it seems more efficient not to share all information; this is especially useful if the knowledge base can contain sensitive information, such as a medical file.

Inquiry dialogues allow agents to only share the knowledge which is relevant, allowing an agent to keep certain information to himself unless a dialogue partner makes it clear this information is required to further the dialogue. Aside from being a possible performance enhancement [10] , this allows the agent to maintain a measure of privacy in applications when such privacy is highly valued.

2.5 Soundness and completeness

Black and Hunter have defined the benchmark for soundness and completeness for their system for inquiry dialogues as the union of the beliefs of the agents. They have proven that using the exhaustive search strategy on the finite belief base of the agents results in a use of all the relevant beliefs of that agent, and could not possibly continue forever.

Soundness of argument inquiry dialogues can be proven from the fact that the commitment store is a subset of the union of the belief bases of the agents, and the fact that the set of arguments created from a subset, is itself a subset of the set of arguments created from the whole set. Completeness follows from the fact that no *open* or *assert* moves are available when the dialogue ends, according to the definition of the exhaustive search.

Soundness and completeness of warrant inquiry dialogues is shown by proving that the resulting dialectical tree is equal to the tree generated by the union of the agent's belief bases. From the fact that everything that will have an influence on the tree will be stated by the exhaustive search strategy, and a proof that shows that every path from leaf to root in the dialogue's dialectical tree is also in the union's dialectical tree, it follows that the dialectical tree generated from the dialogue and the tree generated from the union of the agent's belief bases are equal.

[10]For example, if x percent of the agent's information is required for deduction at most x percent is used; less if some of the data is present in both knowledge bases and therefore not requested by the dialogue partner. Merging two knowledge bases with possibly conflicting, complex or inconsistent information, with $100 - x$ percent irrelevant data, can be very time-consuming, and may spend time on the merging of data that is not relevant for the dialogue.

2.6 Summary

This chapter has elaborated on the system for inquiry dialogues defined by Black and Hunter in [2] and the exhaustive strategy provided with it. The formal definitions have been given and some of the properties have been given as well. Soundness and completeness, for example, has been informally described and not formally defined.

3 Prakken's framework for persuasion dialogues

In [8] Prakken describes a framework for persuasion dialogue games, in order to make implicit design principles more visible and to investigate how to reconcile the static nature of nonmonotonic logic with the dynamic nature of dialogue games. Additionally, Prakken intended to build the framework to allow several types of protocols to be formulated, giving certain characteristics that must be common for the dialogue games.

3.1 Definition of the framework

Definition 28. A **dialogue system**, or dialogue system for argumentation, is a pair $(\mathcal{L}, \mathcal{D})$, the first parameter $\mathcal{L}$ being a logic for defeasible argumentation, and $\mathcal{D}$ a dialogue system proper.

Definition 29. A **logic for defeasible argumentation** $\mathcal{L}$ is a tuple $(L_t, R, Args, \rightarrow)$ containing the **topic language** L_t, the set R of **inference rules** over L_t, the set $Args$ which contains AND-trees where every node $\in L_t$ and every AND-link is an instantiation of a rule $\in R$, and finally a binary relation $\rightarrow$ of defeat defined on $Args$. An argument A is a tree where $prem(A)$ is the set of leaves and $conc(A)$ is the root of the tree.
Given a set of arguments $\mathcal{A}$, several properties can be defined. An **argumentation theory** T_F where $f \subseteq L_t$ within $\mathcal{L}$ is a pair $(\mathcal{A}, \rightarrow_{/\mathcal{A}})$ consisting of a set of arguments in $Args$ with only nodes from F and the $\rightarrow$ relation restricted to $\mathcal{A} \times \mathcal{A}$. If all arguments in T_F have a finite number of defeaters, $t_{\mathcal{A}}$ is called finitary.
For any set $\mathcal{A} \subseteq Args$ the set of all formulae that are a premise of an argument is $\mathcal{A}$ is called the **information base** $I(\mathcal{A})$. The **closure** of a set of arguments $\mathcal{A} \subseteq Args$ is the argumentation theory $T_{I(\mathcal{A})}$.

An argument B **extends** an argument A if $conc(B) \in prem(A)$[11]. Arguments are concatenated using the $\otimes$ operator, and concatenation of A and B, denoted as $B \otimes A$, is only possible if B extends A. The assumption is made that the defeat relation in $\mathcal{L}$ satisfies the property: if A defeats B then $\forall C, D$ if C extends A and D extends B then $C \otimes A$ defeats $D \otimes B$.

Definition 30. A **dialogue system proper** $\mathcal{D}$ is defined as a tuple (L_c, P_{Pr}, C) containing the communication language L_c, P_{Pr}, the protocol for L_c and a set C containing the effective rules of locutions, which specify the effect of locutions on commitments.
The **communication language** is a set of locutions. Every element $p(c)$ of the set is a performative $\in Perf$ with either a subset of the topic language L_t as argument or a member of $Args$. Two binary relations are defined on L_c, the

[11]So $r\, since\, s$ extends $p\, since\, q, r$. Prakken chooses to list an argument with a single inference as *conclusion since premises.*

attacking reply relation is defined with R_a while the surrendering reply relation is defined with R_s. Both are irreflexive and a locution cannot be the subject in both an attacking and a surrendering relation (i.e. it is either an attacking reply, or it is a surrendering reply, but cannot be both) and if it is a surrendering reply to some other locution, it cannot be attacked:

- $\forall a, b, c$ if $(a, b) \in R_a$ then $(a, c) \notin R_s$
- $\forall a, b, c$ if $(a, b) \in R_s$ then $(c, a) \notin R_a$

The function att is defined to assign to every pair $(a, b) \in R_s$ one or more **attacking counterparts** $(c, b) \in R_a$.

Definition 31. The set M of **moves** is defined as $\mathbb{N} \times \{P, O\} \times L_c^p \times \mathbb{N}$, containing the identifier $id(m)$, the player $pl(m)$, the speech act $s(m)$ and the target $t(m)$ of the move.
The set of **dialogues** $M^{\leq\infty}$ is the set of all sequences of moves $\in M$ such that for every sequence $m_1, \ldots, m_i, \ldots$ the first move has no target ($t(m_1) = 0$), the ith element has identifier i ($id(m_i) = i$) and $\forall i$ if $i > 1$ then $\exists j$ where $1 \leq j < i$ and $t(m_i) = j$. The set of finite sequences that satisfy these properties is called $M^{<\infty}$.
A sequence $m_0, \ldots, m_i$ that is part of a dialogue $d = m_0, \ldots, m_i, \ldots$ is denoted by d_i, so d_0 denotes the empty dialogue. The continuation of d with move m is denoted d, m.

Definition 32. A **turntaking function** is defined as $T : M^{<\infty} \rightarrow \mathcal{P}(\{P, O\})$, where $T(\emptyset) = \{P\}$.

Definition 33. A **protocol** on M is a set $P_{Pr} \subseteq M^{<\infty}$ such that if $d \in P_{Pr}$ then $\forall i, d_i \in P_{Pr}$. A function $Pr : M^{<\infty} \rightarrow \mathcal{P}(M)$ is defined to produce the moves that are allowed after a dialogue d as

$$Pr(d) = \begin{cases} undefined & \text{if } d \notin P_{Pr} \\ \{m \mid d, m \in P_{Pr}\} & \text{otherwise.} \end{cases}$$

The domain for Pr, denoted $dom(Pr)$ is the set of **legal finite dialogues**. Note the difference between $Pr(d)$ returning $\emptyset$ and *undefined*, the first means that the dialogue is legal, but it is **terminated** as there are no more possible moves; the second means that the given dialogue is not legal.

A few conditions must be met by any protocol; it must obey the turntaking function, it must obey the relations of attack and surrender in the protocol language, no player can reply to its own moves, the same reply may not be used twice to attack the same move and a surrendering reply cannot be undone.
Definition 34. $\forall d, m$ if $m \in Pr(d)$ then

R_1 $pl(m) \in T(d)$

R_2 if $d \neq d_0$ and $m \neq m_1$ then $s(m)$ is a reply to $s(t(m))$ according to L_c

R_3 $\forall m' \in d$ if $t(m) = m'$ then $pl(m) \neq pl(m')$

R_4 $\forall m' \in d$ if $t(m) = t(m')$ then $s(m) \neq s(m')$

R_5 $\forall m' \in d$ if $t(m) = t(m')$ and $s(m')$ is a surrendering reply, then m is not an attacking counterpart of m'

Definition 35. Lastly, a **commitment function** is defined as $C : M^{\leq\infty} \times \{P, O\} \rightarrow \mathcal{P}(L_t)$. $C_d(p)$ denotes the commitments a player has made in a dialogue d, so $C_\emptyset(p) = \emptyset$.

3.2 Elements of the framework

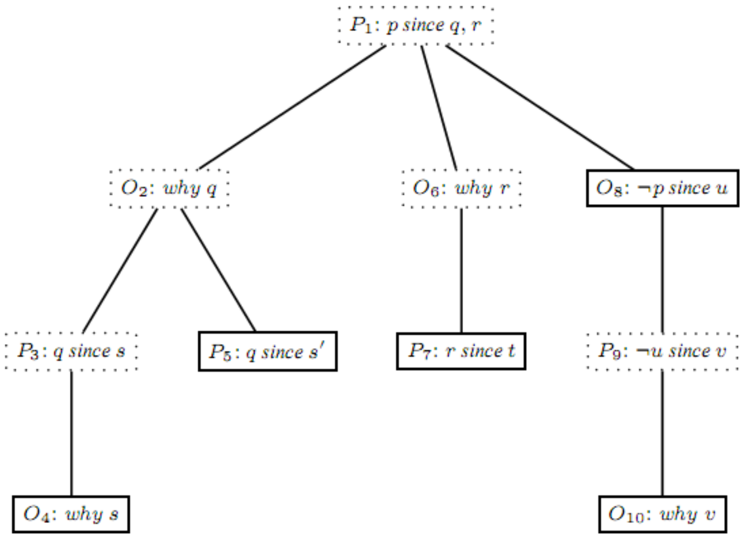
Prakken intended for the framework to impose a common structure on (a selection of) dialogue games; an explicit reply structure, where moves either attack or surrender to a move that was made at some earlier point in the dialogue. Since many dialogue games already implicitly use this structure, Prakken hoped to make the structure explicit with the framework.

Dialogues that are less focused on conflict of opinion, and more on investigation or deliberation might not want this rigid reply structure and may choose to allow a more loose structure for turntaking. Of course, variations should be allowed; for example, a dialogue system might wish to have a different set of locutions, a different argument-based logic, or allow for backtracking, multiple replies or postponing of replies.

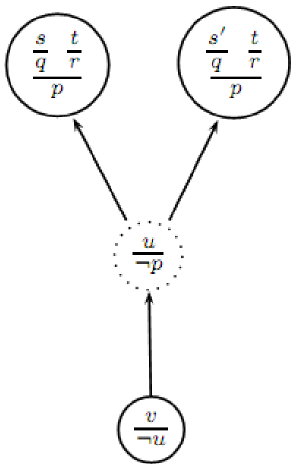
3.2.1 Dialectical graph

During a dialogue, players implicitly build a structure of arguments and counterarguments for and against the initial claim. This structure can be represented in several ways. First, it can be represented as a linear order, structured only by the time or turn the moves were made.

Another way to represent the dialogue is to structure the dialogue by the reply relation between moves. This results in a tree of moves and countermoves, the dialogue tree. The dialogue tree contains all the moves against the initial claim (the root of the tree) and all its attackers, and their attackers, etc.

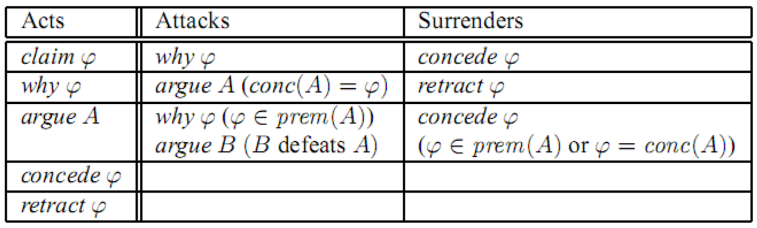


A third way to structure the dialogue is to create a dialectical graph of the content of the arguments moved in the dialogue, combining arguments that were made over several moves or replied implicitly to several moves.


$\dfrac{\dfrac{s}{q} \quad \dfrac{t}{r}}{p}$
$\dfrac{\dfrac{s'}{q} \quad \dfrac{t}{r}}{p}$
$\dfrac{u}{\neg p}$
$\dfrac{v}{\neg u}$

3.2.2 Moves

Below is an example of a possible set of speech acts for the framework.



Acts	Attacks	Surrenders
claim φ	*why* φ	*concede* φ
why φ	*argue* A ($conc(A) = \varphi$)	*retract* φ
argue A	*why* φ ($\varphi \in prem(A)$) *argue* B (B defeats A)	*concede* φ ($\varphi \in prem(A)$ or $\varphi = conc(A)$)
concede φ		
retract φ		

The above moves have the following effect on the commitment store: *claim* and *concede* append the fact to the commitment store, *argue* adds both the premises and conclusion of the argument to the commitment store, *retract* removes an argument from the commitment store and *why* does not influence the commitment store.

The proponent of the initial claim starts with a unique move, introducing the initial claim to the dialogue. A restriction to concessions is defined, to ensure concessions are made as a reply to the move that introduced the conceded argument, not the last move to have used the conceded argument.

3.2.3 Attack and defeat

The framework considers three possible ways to attack an argument (for example ϕ since ψ):

1. premise-attack
   countering the premise of the argument (example: $\neg\psi$)

2. undercutting
   arguing that the rule used in the argument is incorrect or incomplete (example: *not*(ϕ since ψ))

3. rebuttal
   countering the conclusion of the argument with a contradictory argument (example: $\neg\phi$ since χ)

Not all systems need use the three possible ways of attacking; many systems opt to ignore or disallow one or even two of these types of attack.

3.3 Termination and outcome

Ideally, dialogues end when the player taking turn has run out of legal moves to make. This is not always the case; some dialogue types may allow information to come from other sources and may allow belief bases to evolve during the dialogue as new information is gathered during the dialogue. It may also be possible to endlessly challenge arguments using the *why* move.

It is reasonable to assume that many dialogues will not end in the ideal fashion, but end with an external agreement to end the dialogue, or a decision to terminate the dialogue. Therefore, the outcome must be determined on a per move basis, determining who would be the winner were the dialogue to end.

3.3.1 Dialogical status and outcome of dialogue

As can be expected, only the arguments which have not been successfully defeated must be considered to determine the dialogical status of the initial claim. The dialogical status of moves, whether they are *in* or *out*, is determined for a move *m* by determining whether every move that attacks move *m* has been successfully replied to. If there are attacking moves that have not been replied to, or have been replied to with a surrendering move, move *m* is *out*.

Prakken uses the *in* and *out* states of moves to determine whether the initial claim is substantiated or disproved by the dialogue. The losing player has the 'burden to attack', and failing to attack will at some point lose him the dialogue.

3.4 Soundness and fairness

Soundness is defined as the fact that a dialogue that concludes supporting a claim, must mean that the information that is agreed upon in the dialogue must also imply the claim. The reverse, if the agreed upon information implies a certain conclusion about the initial claim of the dialogue, this conclusion must be the result of the dialogue, is called fairness.

Both of these properties require a property called logical completion. If this property is present in the dialogue system, soundness and fairness can be proven from it.

3.4.1 Logical completion

A dialogue that has terminated may conclude a different status for the initial claim than would be concluded from the information that is agreed upon in the dialogue. This can happen when the dialogue has ended before all the logical counterarguments which can be constructed from the commitment store for either the initial claim, or any supporting or counterclaim have been given, or when several premises stated during the dialogue can be combined to conclude something that was not explicitly stated in the dialogue.

Compare the following trees, adapted from Prakken's article. Ending with the first tree results in a different conclusion from the dialogue than ending with the second tree. The second is logically complete, while the first did not include O4, and as such was not complete. Note that O4 must have been constructed from the arguments that have been stated by the agents in this dialogue.

$$\begin{tikzcd}
P1^{+} \arrow[d, no head] \\
O2^{-} \arrow[d, no head] \\
P3^{+}
\end{tikzcd}$$

$$\begin{tikzcd}
 & P1^{-} \arrow[dl, no head] \arrow[dr, no head] & \\
O2^{-} \arrow[d, no head] & & O4^{+} \\
P3^{+} & &
\end{tikzcd}$$

Soundness and fairness can only be proven for dialogues that are logically complete; only when all minimal arguments that defeat an argument in the dialogue have been used against that argument, can soundness and fairness be concluded.

3.5 Relevance

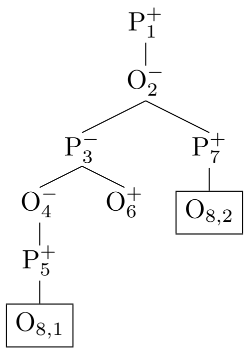
A relevant move is a move that has an influence on the outcome of the dialogue; in other words, if a move does not achieve a change in dialogical status for the initial claim, it is not relevant. There is, of course, an exception: a surrender move, such as concede or retract, would not change the dialogical status of the initial claim, but it is relevant if the move it surrenders to is a relevant target.

Definition 36. An attacking move in a dialogue d is **relevant** iff it changes the dialogical status of d's initial move. A surrendering move is relevant iff its attacking counterparts are relevant.

Relevance [12] can be a useful condition to have for a dialogue system, it limits the players from cluttering the dialogue with longwinded and irrelevant argumentation.

An illustrative example adapted from [8] shows the difference:

[12] Or strong relevance, when compared to the later mentioned weak relevance.



While $O_{8,1}$ might influence a few nodes of the dialectical tree, its influence does not extend as high as the root node, so the argument is irrelevant. On the other hand, $O_{8,2}$ would affect the root node, so it is relevant.

3.6 Summary

This chapter has shown the basics of Prakken's framework. The formal definition has been given and some of the properties Prakken examines in his framework have been elaborated.

4 Comparing Black and Hunter's inquiry dialogues with Prakken's framework

Any dialogue system can be adapted to fit Prakken's framework (or the framework adapted to the fit the system); some properties may need to be changed to achieve the reconciliation of differences, but it can be adapted to a system. However, the important question is not so much whether it is possible, but *how* and at what cost.

Which properties of a system are lost when adapting to the framework? How important are these properties to the fundamental idea of the system to be adapted to the framework? In order to determine the successfulness of the adaptation to the framework, it is necessary to define what is fundamental to the system, and how the adaptation influences those fundamental properties.

This chapter examines which properties are fundamental to the concept of an inquiry dialogue system defined by Black and Hunter. It also examines what similarities and differences can be determined between the system for inquiry dialogues and the framework as defined by Prakken. Finally, it determines to what extent it is possible to reconcile the differences between Prakken's framework and the system for inquiry dialogues.

4.1 Fundamental properties of inquiry dialogues

The most important properties of Black and Hunter's inquiry dialogue system are also the ones that differentiate it from other dialogue systems. These properties may be shared by some dialogue systems, but are uncommon in others, and the list of fundamental properties might be considered a compact definition of Black and Hunter's inquiry dialogue system as variation on a hypothetical basic dialogue system[13]. Fundamental properties are essential to the system; if one were to remove, for example, the cooperative nature from inquiry dialogues, the very concept of inquiry dialogue will be severely, if not completely, crippled.

Properties that should be considered fundamental are the cooperative nature of inquiry dialogues, the possibility of keeping information private unless there is cause to share it, and the fact that agents can (and must) slip in and out of roles as the dialogue progresses.

Other properties that the system for inquiry dialogues has that might be considered fundamental properties, but which can be influenced by agent's strategies (as they are built upon the provided exhaustive strategy), are properties such as the predetermined result of a dialogue, which allows statements about the soundness and completeness of dialogues using the union of belief bases as a

[13]i.e. a dialogue system that only has the basic properties that (almost) every dialogue system has.

benchmark.

Additionally, the simplicity of the possible moves, with only *open* , *close* and *assert* moves available, could be an essential property that might be preserved in some way.

Also a notable difference, and another fundamental property of Black and Hunter's system, is the fact that there is no notion of targeting for moves, while Prakken's framework requires every move to have a target. In reconciling this difference, either all moves from Black and Hunter's system have to be altered in some way to allow them to have targets, or Prakken's framework has to be altered to allow the absence of a target.

So, to summarize, the fundamental properties of Black and Hunter's inquiry dialogue system are:

1. the cooperative nature of the dialogue
2. the distinction between agents and agent roles because of the absence of role definitions, allowing for agents to make moves both supporting and opposing the topic of the dialogue
3. the predetermined (reliable) result of dialogues when the exhaustive strategy is used, ensuring soundness and completeness
4. soundness and completeness can be benchmarked on the belief base when the exhaustive strategy is used
5. the simplicity of available moves
6. moves have no target, the dialectical tree determines the targeting

4.2 Differences

Of the fundamental properties, the fourth can be considered to be available in a somewhat similar fashion in Prakken's framework. The others are the major differences that must be overcome to make a successful adaptation that allows the systems for inquiry dialogues and Prakken's framework to be compatible.

Finally, there is a difference in determining the outcome of the dialogue. The system for inquiry dialogues generates a dialectical tree with all the arguments that have been made for and against the topic, and determine the outcome based on this tree. Prakken's framework, however, creates a dialogue tree of moves that have been made, and a dialectical graph with all the arguments, and bases the outcome on the dialogue tree. This is because Prakken's framework has several types of moves that have an influence on the status of moves, but should not be in the dialectical graph. The graph is used to make the arguments constructed using argument extension visible.

4.2.1 Minor Differences

The framework allows lies and holding back information that is relevant, while the agents using the exhaustive strategy in Black and Hunter's system for inquiry dialogues share anything relevant. As such, this should still be considered a minor difference; it may require some adaptation, but it should not be a large or problematic adaptation.

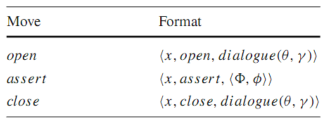
Another minor difference is the query store that is available in Black and Hunter's system for inquiry dialogues. It contains the literals that need to be proven in order to construct an argument for the topic of the dialogue. Prakken's framework has no comparable feature. Since the same information can be easily obtained from the dialogue itself, this is not considered to be important.

4.3 Mapping

The following table contains a mapping of definitions; for many definition from Prakken there is a similar idea in Black and Hunter's system. The overlap or compatible definition have been marked with a ✓, the problematic definition or the ones that require some modification to be compatible have been marked with a !.

Prakken's framework		Black & Hunter's system
L_t topic language contains defeasible facts	✓	$\mathcal{S}^*$ and $\mathcal{S}$ set of defeasible facts and set of defeasible facts with preference level
R inference rules contains strict rules	!	$\mathcal{R}^*$ and $\mathcal{R}$ set of defeasible rules and set of defeasible rules with preference level
$Args$ set of all arguments contains all possible arguments that can be built from L_t and R	✓	$\mathcal{A}(\mathcal{S} \cup \mathcal{R})$ set of all arguments generated from a given belief base
Argument A AND-tree where nodes $\in L_t$ links $\in R$	✓	Argument A defeasible derivation a minimal set of premises leading to a claim facts $\in \mathcal{S}$ and rules $\in \mathcal{R}$

Continued on Next Page...

Prakken's framework		Black & Hunter's system
$prem(A)$	✓	$Support(A)$
$conc(A)$	✓	$Claim(A)$
$\rightarrow$ binary defeat relation on $Args$	✓	attack relation with definition of defeat based on preference level
L_c communication language set of elements $p(c)$ where p is a performative $\in Perf$ and $c \subseteq L_t$ or $c \in Args$	!	table of moves  Move – Format *open* – $\langle x, open, dialogue(\theta, \gamma)\rangle$ *assert* – $\langle x, assert, \langle \Phi, \phi \rangle\rangle$ *close* – $\langle x, close, dialogue(\theta, \gamma)\rangle$
$Perf$ set of performatives p	✓	implicit in the table of moves table of moves: *open* , *close* , *assert*
R_a binary attacking reply relation on L_c	!	attack relation on c not part of the system, no attack relation is defined on performatives / moves, only on arguments
R_s binary surrendering reply relation on L_c	✓	$\emptyset$ not part of the system
Dialogue $d_n = m_0, \ldots, m_n$ sequence of n moves with the following conditions: every move must have a target except m_0 moves may not target themselves	!	Dialogue $D_r^t = m_r, \ldots, m_t$ sequence of moves from index r to index t no targets have been defined for moves
M, $M^{\leq\infty}$ and $M^{<\infty}$ set of moves, set of possibly infinite dialogues and set of finite dialogues	✓	$\mathcal{M}$ and $\mathcal{D}$ the set of moves and the set of dialogues

Continued on Next Page...

Prakken's framework		Black & Hunter's system
move 4-tuple (id, pl, s, t) $id(m)$ = number in the sequence of a dialogue $pl(m)$ = player who moved $s(m)$ = speech act $t(m)$ = target of the move	!	move $\langle x$, performative, content$\rangle$ not saved in move, but in dialogue $x \in \mathcal{I}$ performative and content not defined
T Turntaking function $Dialogue \rightarrow \{P, O\}$	!	defined in protocol the system alternates players, if the last move was 1, the next move is by 2 and vice versa.
$\{P, O\}$ player roles (Proponent, Opponent) bound to players	!	$\mathcal{I}$ set of players (there are no specific roles)
P_{Pr} and function Pr set of dialogues that are valid and function that produces the allowed moves for a given dialogue must obey turntaking rules	✓	$\Pi_{ai}(D_r^t)$ or $\Pi_{wi}(D_r^t)$ depending on which dialogue is currently active in the given dialogue (given by $Type(Current(D_r^t)))$
C commitment function determines the effect moves have on the commitment store	✓	CS CS_x^t defined in Definition 20
$\otimes$ argument concatenation	!	not defined the modification of arguments is not allowed, the original argument remains and a new one is created that has the addition arguments as subarguments
Outcome outcome is based on the dialogue tree of moves	!	outcome is based on the dialectical tree of arguments

Continued on Next Page...

Prakken's framework	Black & Hunter's system

Table 3: Mapping definitions in Prakken's framework and Black and Hunter's system for inquiry dialogues

The next sections consider the items marked with an exclamation mark in the table above, and some that are marked with a ✓, but require some additional elaboration.

4.3.1 Arguments

To show that arguments in Prakken's framework and arguments in Black and Hunter's system for inquiry dialogues are interoperable, two functions are provided to convert Prakken's AND-trees to Black and Hunter's defeasible derivations, and vice versa.

Definition 37. The function **argumentToInquiry**, of type $argumentToInquiry : Args \mapsto \mathcal{A}(\mathcal{S}^* \cup \mathcal{R}^*)$, converts an argument in the form of an AND-tree to an argument as defined by Black and Hunter, by returning a postorder traversal of the nodes of the AND-tree.

$argumentToInquiry(A) =$

```
    $E$ = postorder traversal of $A$
    $l$ =last element in $E$, then remove $l$ from $E$
    for every element $e$ in $E$
        if the node marked with $e$ has children in $A$
        replace $e$ with a rule $\left[\bigwedge_{children} \rightarrow e\right]$
    return $\langle E, l\rangle$
```

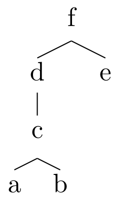
Definition 38. The function **argumentToFramework**, which is of type $argumentToFramework : \mathcal{A}(\mathcal{S}^* \cup \mathcal{R}^*) \mapsto Args$, converts an argument in the form of a defeasible derivation to an AND-tree as follows.

$argumentToFramework(\langle \Phi, \phi\rangle) =$

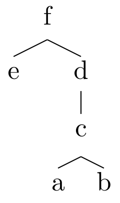
```
Create an AND-tree $ANDT$ with root node $\phi$
while($\Phi \neq \emptyset$)
    for every node $N \in ANDT$ with label $\lambda$
        $\forall \Psi$ where the rule $\left[\bigwedge_{\psi \in \Psi} \psi\right] \to \lambda \in \Phi$
        add nodes for all $\psi \in \Psi$, each a child of $N$
        and $\Phi = \Phi \setminus \left\{\left[\bigwedge_{\psi \in \Psi} \psi\right] \to \lambda\right\}$
```

Note that if the preference level needs to be preserved, the label for the nodes can be modified to contain the preference level as well.

For example, given the argument $\langle\{(a, 1), (b, 1), (a \wedge b \to c, 1), (c \to d, 1), (e, 1), (d \wedge e \to f, 1)\}, f\rangle$, the following AND-tree would be produced:



Postorder traversal of the tree returns the original sequence: $\{a, b, c, d, e, f\}$. Note that if the order of the children was deemed irrelevant, the same tree could be produced with the d and e node switched, however postorder traversal would provide a different, though correct, defeasible derivation: $\{e, a, b, c, d, f\}$



With these functions defined, there can be no doubt that arguments in Prakken's framework and those in Black and Hunter's system are compatible.

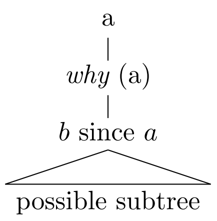
4.3.2 Argument concatenation $\otimes$

Prakken has defined a rule for the consistency of $\otimes$: if A defeats B then $\forall C, D$ if C extends A and D extend B then $C \otimes A$ defeats $D \otimes B$.

In Black and Hunter's system, this rule does not hold, it is true that if $pLevel(A) > pLevel(B)$ then $pLevel(A \otimes B) > pLevel(B)$; this means that argument concatenation may weaken an argument, so[14]:
if A defeats B then $\forall C, D$ if C extends A and D extend B then $C \otimes A$ defeats $D \otimes B$ only if $pLevel(C) \leq pLevel(A)$ or $pLevel(C) \leq pLevel(D)$.

However, Black and Hunter's system does not give players the ability to modify arguments that have been made. Any combined argument is a new argument in a new move, it does not replace the previous argument.

Agents in Prakken's framework build an argument like this, for example[15]:



Obviously, a provided explanation responding to an attacker needs to strengthen, and never weaken the original move.

Black and Hunter use a dialectical tree that is constantly generated from all the arguments that have been asserted by the agents. If an argument is asserted that is weaker than the argument it might have intended to attack, it simply does not appear in the dialectical tree. If a move combines two previously asserted arguments into a new argument, this is simply a new move, and a new argument.

4.4 Optional modifications after conversion

4.4.1 The underlying logic

One thing to consider when looking at a dialogue system is its underlying logic. Black and Hunter's system is based on DeLP[16], and while Prakken's framework allows (theoretically) most logics for nonmonotonic reasoning, it is worth considering whether this underlying logic should be preserved, and at what cost. As most of the options for underlying logics are functionally comparable, it might be worth it to change the underlying logic for the inquiry framework to be of a

[14]While Black and Hunter have no definition for extending arguments, Prakken's definition is hypothetically applied to the arguments here to show that it would not be consistent if it were in Black and Hunter's system.
[15]Note that Prakken enforces no moves in the framework, and the *why* used here is just an example.
[16]Which, as you may remember, is Garca and Simari's Defeasible Logic Programming [5]

more reliable kind.

The use of DeLP can be considered to be somewhat problematic. The list of restrictions for the argument line has been shown to be somewhat problematic at times, allowing structures that should not be allowed. A small example of this would be the following argumentation line:

$$\begin{tikzcd}
\langle \{(a,2)\}, a \rangle \arrow[d, dash] \\
\langle \{(b,1),(b \to \neg a,1)\}, \neg a \rangle \arrow[d, dash] \\
\langle \{(\neg b,1)\}, \neg b \rangle
\end{tikzcd}$$

Note that $\langle \{(b,1),(b \to \neg a,1)\}, \neg a \rangle$ and $\langle \{(\neg b,1)\}, \neg b \rangle$ are blocking defeaters of each other, according to the definition on page 13. Sceptically, this argumentation line should not be acceptable; if $\neg b$ is true, then a is as well, but if b is true, which is just as likely, then a is not true. However, according to the restrictions, this argumentation line is acceptable, so a would be considered true (or warranted).

Since none of the fundamental properties require any specific property of DeLP, any semantics is likely a viable choice; there are no restrictions in the systems except for the structure of arguments. Black and Hunter also mentions this in [3] at the end of chapter 7, stating that it is possible to "adapt our system to use the semantics and defeat relation of any argumentation system where the acceptability of an argument depends on the argument graph constructed around it".

However, integrating specific semantics into the adapted systems is not considered. A textual explanation of the process is given, and an intuition of the ease of this additional modification is provided. Furthermore, no reference is made of a specific underlying logic, so that future work may attempt to use the systems with any underlying logic they please.

4.4.2 Expanding for more than two agents using role switching

Given that agents in Black and Hunter's inquiry dialogue system are not bound to the role of either proponent or opponent, it is easy to expand the system to more than 2 agents. Other dialogue systems lock the agents in their role, and as the number of roles is hard to increase, they might have a lot of difficulty increasing the number of agents.

In Black and Hunter's system, however, one can easily add any umber of agents, and each takes up whichever of the two roles is appropriate to use when they have a contribution to make. This will be shown during the construction of the adapted systems.

4.5 Summary

In this chapter the definitions of the two systems have been compared and a measure of difference has been established. It has become clear that while there are some fundamental differences, there are also many similarities. From the difference that were established in this chapter, the next chapters will define two systems that either adapt Black and Hunter's system to Prakken's framework, or adapt Prakken's framework to Black and Hunter's system.

5 Adapted version of Black and Hunter's system for inquiry dialogues that is compatible with Prakken's framework

This chapter examines the changes needed to Black and Hunter's system for inquiry dialogues in order to adapt it to Prakken's framework, without changing anything that was formally defined in Prakken's framework definition. Functions that were not explicitly defined, such as the definition for the turntaking function are given where necessary.

This provides an instantiation of Prakken's framework that is capable of formulating the same dialogues as can be formulated with Black and Hunter's system.[17] As such, any statement made about Prakken's framework is applicable to the adapted system for inquiry dialogues.

First, the core of the framework is repeated with some notational alterations. After that, the necessary adaptions to Black and Hunter's system and the functions that have to be defined in Prakken's framework are given. Next, a function is defined to provide the outcome of a dialogue and a method to create the dialectical graph is described. After that, some example dialogues are displayed, along with with dialogue trees, to clarify the changes. Then, functions are provided to translate a dialogue from Black and Hunter's system for inquiry dialogues to the adapted system and Black and Hunter's exhaustive strategy is given for the adapted system. Finally, the fundamental properties are evaluated for the adapted system.

5.1 Framework definition

The basic definition of Prakken's framework is given below. Note that the following definitions are given to make explicit some notational changes for Prakken's framework[18] and to avoid conflicts of notation. A clear table of notation for both Black and Hunter's and Prakken's article can be found on page 40. Note that comments made about notation on page 11 apply here as well.

Occasionally, the notation $\left[\bigwedge \phi\right] \rightarrow \psi$ is used as a more readable notation of $\left[\bigwedge_{\phi \in \Phi} \phi\right] \rightarrow \psi$ or $\phi_0 \wedge \ldots \wedge \phi_n \rightarrow \psi$. The aim is to allow statements to be made about the premises of the rules, using ϕ to denote any of the premises.

Additionally, a dialogue d_t as denoted in Prakken's framework or D_r^t as denoted in Black and Hunter's system is a sequence of moves. Since the moves

[17]Note that for this comparison, the equality is not defined on the basis of moves, but on the basis of content and outcome.

[18]For the original notation and definition, see Definition 28 through 30

are denoted with natural numbers, and natural numbers are well-ordered, these dialogues are now assumed to be well-ordered sets of moves, allowing notation such as $m_t \in D_r^t$ and $m_t \in d_t$.

Definition 39. An **inquiry dialogue system** is a pair $\mathcal{L}, \mathcal{DSP}$ where

$\mathcal{L}$	is a logic for defeasible argumentation
$\mathcal{DSP}$	is an inquiry dialogue system proper

Definition 40. A **logic for defeasible argumentation** $\mathcal{L}$ is a 4-tuple $L_t, R, Args, \Rightarrow$ where

L_t	is the logical language, consisting of defeasible facts $\mathcal{S}^*$ and defeasible rules $\mathcal{R}^*$
R	is a set of inference rules
$Args$	is a set of arguments
$\Rightarrow$	is a defeat relation between arguments

The set of arguments and the defeat relation between arguments are defined later under argumentation theory. The set R is defined to contain a single rule, the modus ponens.

$$R = \left\{ \frac{\begin{array}{l} L_0 \wedge \ldots \wedge L_j \rightarrow L_n \\ L_0 \wedge \ldots \wedge L_j \end{array}}{L_n} \right\}$$

Note that Prakken's framework requires the 'defeasible modus ponens' rule to support the constructing of arguments from its logical language L_t, as the set of strict rules which build arguments in Prakken's framework is defined to be empty by Black and Hunter. In [9] Prakken encountered a similar situation and used this rule to solve it[19].

Using this rule, arguments from Black and Hunter of the form $\langle\{(a,1)(a \rightarrow b,1)\}, b\rangle$, which only contain beliefs, can be constructed in Prakken's framework in the form b since $(a,1)(a \rightarrow b,1)$, implicitly using the above modus ponens for defeasible rules. Definitions 37 and 38 explained this in more detail.

Definition 41. An **inquiry dialogue system proper** $\mathcal{DSP}$ is a 3-tuple (L_c, Π, C) where

L_c	is a communication language
Π	is a protocol
C	is a commitment store update function

[19]Note that the original rule in [9] contains elements $\sim L_k$, which means that there is no evidence for L_k. However, as this is not used in Black and Hunter's defeasible rules, it has been excluded here. Also, Prakken used $\Rightarrow$ for defeasible rules, while we use $\rightarrow$; this has been replaced accordingly.

Several changes are necessary to adapt Black and Hunter's system for inquiry dialogue to Prakken's framework. Each change is described and explained below. Some examples are provided in between, however the most complete examples will be shown after the explanations, in Table 6 through 9 and their accompanying dialogue trees.

5.2 Argumentation theory

For every literal or rule in L_c ($\phi \in \mathcal{S}^*$ and $\left[\bigwedge \phi\right] \rightarrow \psi \in \mathcal{R}^*$), there can be tuple of that element with a preference level p, (ϕ, p) or $\left(\left[\bigwedge \phi\right] \rightarrow \psi, p\right)$. The set of these literals with preference level is defined as $\mathcal{S}$, the set of rules with preference level is defined as $\mathcal{R}$. Arguments contain these tuples and the premises of an argument derive the conclusion using the modus ponens rule defined above. Note that the function that gives the *pLevel* of an argument was defined in Definition 7.

The set of arguments *Args* is defined as in Black and Hunter's system, as all the arguments that can be constructed from all the beliefs (defeasible facts and defeasible rules with preference level).

Definition 42. The set of arguments is defined as:

$Args = \mathcal{A}(\mathcal{S} \cup \mathcal{R})$

The $\Rightarrow$ relation, which is the defeat relation between arguments, is derived from the definition of the defeat relation between arguments given by Black and Hunter.

Definition 43. The **defeat relation between arguments** is a set of tuples of arguments, using the attack defined in Definition 8.

$\Rightarrow$ (defeat relation between arguments) =

$$\left\{ (A, B) \;\middle|\; \begin{array}{l} A, B, C \in Args \\ \text{and } A \text{ attacks } B \text{ at subargument } C \\ \text{and } pLevel(A) \leq pLevel(C) \end{array} \right\}$$

5.2.1 Roles, players and agents

Players in Black and Hunter's system for inquiry dialogues do not have roles, while players in Prakken's framework are tied to the specific roles. However, this is not a problem for the adaption: we use the term role for Prakken's players (the set $\{P, O\}$), and player for Black and Hunter's players (the set $\mathcal{I}$). Players may make a move as they see fit, and the role is assigned to the move accordingly (i.e. if a move targets a move made by P, it is made by O, and vice versa).

First, the roles in Prakken's framework require a slight change of notation in order to have a notation for players. Where the notation for Proponent was P, and a move by *Proponent* denoted P_{id} (with *id* being the *id* of the move), it is changed to $P(x)$ where $x \in \mathcal{I}$ and a move using the *Proponent* role is denoted $P_{id}(x)$. The notation for Opponent is adapted in a similar fashion.

Definition 44. The **set of roles** is defined as:
$$Roles = \quad \{\ P(x) \mid x \in \mathcal{I}\ \} \cup \{\ O(x) \mid x \in \mathcal{I}\ \}$$

Note that this is not a change that has any influence on Prakken's framework, it is merely notation in order to see which agent has made a move.

5.3 Communication language

This section must start with a comment on notation, to avoid confusion. Black and Hunter define a move as $\langle Agent, Act, Content\rangle$, yet also use the term for what Prakken would refer to as a speech act. From now on, Prakken's definition is used and these are referred to separately.

Prakken's framework does not explicitly support sub-dialogues. However, these sub-dialogues are a sequence of moves that are related to each other because the dialogue definition enforces their connection. In Prakken's framework, a similar effect can be achieved by using the targets of moves.

In order to achieve this, the *open* speech act for argument inquiry dialogues has to be redefined. Since argument inquiry dialogues were already required to discuss a topic related to the topic of the dialogue, the move containing the *open* speech act could itself be given a target, but it would never be defeated successfully[20].

The *open* speech act that is defined in Black and Hunter's system must be split into parts, and each is somewhat different from the original *open* speech act. The *open* speech act for argument inquiry dialogues is replaced with a *propose* speech act, which is followed by several moves with a *query* speech act. For example:

$$\begin{tikzcd}
 & propose(a \rightarrow b, p) \arrow[dl, dash] \arrow[dr, dash] & \\
query(a \rightarrow b, p) & & query(a, p)
\end{tikzcd}$$

[20]Unless one considers *close* moves as attackers of open moves, but that would give a single *close* move the power to close a dialogue

$$\begin{tikzcd}
 & \mathit{propose}(a \land b \to c, p) \arrow[dl, dash] \arrow[d, dash] \arrow[dr, dash] & \\
\mathit{query}(a \land b \to c, p) & \mathit{query}(a, p) & \mathit{query}(b, p)
\end{tikzcd}$$

Note that p denotes the preference level of the nearest asserted argument higher in the tree (in a direct line to the root), or ∞ if no such argument exists. The *open* speech act for warrant inquiry dialogues is still an *open* speech act, but it does not accept the type of dialogue, and must also be followed by a move with a *query* speech act. For example:

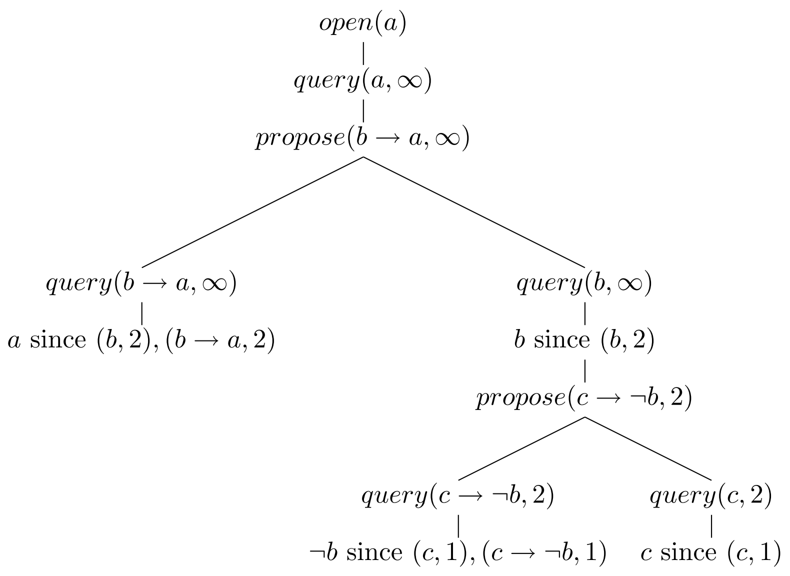
$$\begin{tikzcd}
\mathit{open}(a) \arrow[d, dash] \\
\mathit{query}(a, \infty)
\end{tikzcd}$$

Neither the *propose* nor the *open* add its content to the role's commitment store, but it is considered an successful attacker for the move it targets.

The *query* speech act is similar to the *why* speech act that can be found in many persuasion dialogue systems, except it can only attack moves that contain an *open* or *propose* move, not an *assert* move; the move is used to simulate that the players are not stating that the proposed rule or fact is true, they are proposing that the inquiry considers it. For the *open* speech act, which only accepts a single fact from $\mathcal{S}^*$, the *query* move that follows invites inquiry about this fact (and attacks the move with the *open* speech act, which ensures that nothing is warranted when there are no arguments for it). For *propose* speech acts, where Black and Hunter's system would create a query store[21], a *query* move must be moved for each literal in the query store, except for the consequent of the rule. Instead of a *query* move for the consequent of the rule, for which there may already be a *query* move, a *query* for the entire rule is moved.

An example dialogue tree to show the moves in action:

[21] As defined in Definition 21


$open(a)$
$query(a, \infty)$
$propose(b \rightarrow a, \infty)$
$query(b \rightarrow a, \infty)$
a since $(b, 2), (b \rightarrow a, 2)$
$query(b, \infty)$
b since $(b, 2)$
$propose(c \rightarrow \neg b, 2)$
$query(c \rightarrow \neg b, 2)$
$\neg b$ since $(c, 1), (c \rightarrow \neg b, 1)$
$query(c, 2)$
c since $(c, 1)$

This tree is equivalent to a dialogue from Black and Hunter's system for inquiry dialogues of the form:

$$\begin{array}{l|l}
\multicolumn{2}{l}{\langle 1, open, dialogue(wi, a)\rangle} \\
\hline
 & \langle 2, open, dialogue(ai, b \rightarrow a)\rangle \\
 & \langle 1, assert, \langle\{(b, 2)\}, b\rangle\rangle \\
 & \langle 2, assert, \langle\{(b, 2), (b \rightarrow a, 2)\}, a\rangle\rangle \\
 & \langle 1, close, dialogue(ai, b \rightarrow a)\rangle \\
 & \langle 2, close, dialogue(ai, b \rightarrow a)\rangle \\
\hline
 & \langle 1, open, dialogue(ai, c \rightarrow \neg b)\rangle \\
 & \langle 1, assert, \langle\{(c, 1)\}, c\rangle\rangle \\
 & \langle 2, assert, \langle\{(c, 1), (c \rightarrow \neg b, 1)\}, \neg b\rangle\rangle \\
 & \langle 1, close, dialogue(ai, c \rightarrow \neg b)\rangle \\
 & \langle 2, close, dialogue(ai, c \rightarrow \neg b)\rangle \\
\hline
\multicolumn{2}{l}{\langle 1, close, dialogue(wi, a)\rangle} \\
\multicolumn{2}{l}{\langle 1, close, dialogue(wi, a)\rangle}
\end{array}$$

Note that a move that contains a *propose* speech act, while it semantically opens an argument inquiry dialogue, either inside another dialogue or as the first move of a dialogue, does not actually force any restrictions on the dialogue, as sub-dialogues do in Black and Hunter's system. Moves that are made later in the dialogue can target any other move (except for the querying moves which must follow immediately), and as such, there is no closing move for these 'sub-dialogues'. However, moves that directly target the querying moves that target the move with the *propose* speech act can all be considered to be inside the sub-dialogue.

Prakken's framework does not allow a move to behave as a *close* move does in Black and Hunter's system for inquiry dialogues. The *close* move only has an effect if it is moved by all the players and it closes a dialogue and locks its topic. Since neither is allowed in Prakken's framework, the *close* move is omitted. The section above has shown that *close* moves is no longer required for sub-dialogues, so the behaviour that has to be replaced is the option for agents to skip a turn by moving a *close* move, and the matched close to signify the end of the dialogue.

The matched close functionality is not reproducible in Prakken's framework. However, since Prakken defines a way to evaluate the dialogue after every turn, the matched close to end a dialogue is not required. The second behaviour, to allow players to skip turns, will be dealt with when the turntaking function is defined.

With the *open* speech act adapted for Prakken's framework, the only speech act left is the *assert* speech act. This only needs a slight modification to assert the type of arguments that Prakken defined, but with preference levels. As it has been shown that the arguments from Prakken and Black and Hunter's systems are interchangeable, this is in fact a notational change. Note that the difference between the *propose* and *assert* acts is that the *assert* speech act provides an argument which has a preference level, as defined by Black and Hunter, containing facts and rules with preference levels. The *propose* contains a rule and a preference level as separate parameter. Also, *assert* acts are the only acts that affect the commitment store.

5.3.1 Definition of performatives and communication language

Based on the above, a set of performatives and the corresponding communication language L_c can be defined.

Definition 45. The **performatives** are
$$Perf = \left\{ \begin{array}{ll} assert & \text{for asserting arguments,} \\ propose & \text{to open an argument inquiry,} \\ open & \text{to open a warrant inquiry, and} \\ query & \text{follows } \textit{propose} \text{ and } \textit{open} \text{ to signify inquiry} \end{array} \right.$$
Assert is unary, the others are binary, with restrictions on the parameters to be defined in the communication language.

Based on these performatives, the communication language can be defined.

Definition 46. The **communication language** is defined as the set $L_c =$

$$\begin{array}{l} \left\{ \ assert(c) \mid c \in Args \ \right\} \\ \cup \\ \left\{ \ propose(c,p) \mid c \in \mathcal{R}^*, p \in \mathbb{N}_1 \cup \{\infty\} \ \right\} \\ \cup \\ \left\{ \ open(c,p) \mid c \in \mathcal{S}^*, p \in \mathbb{N}_1 \cup \{\infty\} \ \right\} \\ \cup \\ \left\{ \ query(c,p) \mid c \in \mathcal{S}^* \cup \mathcal{R}^*, p \in \mathbb{N}_1 \cup \{\infty\} \ \right\} \end{array}$$

5.3.2 Attack and surrender relations

The new speech acts have specific attack relations. The definition of the attack relation below shows the attack relations between speech acts.

Attacking speech act	Type of speech act it can attack
$open(\phi)$	*open* cannot attack any moves
$assert(A)$	$assert(A')$, if A defeats A',
	$query(\phi, p)$ if A is an argument for ϕ that can defeat an argument with pLevel p
	$query\left(\left[\bigwedge \phi\right] \to \psi, p\right)$ if A is an argument for ψ that can defeat an argument with pLevel p, and the queried rule is in the premises of A
$propose\left(\left[\bigwedge \phi\right] \to \psi, p\right)$	$assert(A)$ if A is an argument for $\neg\psi$ that has pLevel p
	$query(\psi, p)$
$query(\phi, p)$	$propose\left(\left[\bigwedge \phi\right] \to \psi, p\right)$,
	$open(\phi)$, where $p = \infty$
$query\left(\left[\bigwedge \phi\right] \to \psi, p\right)$	$propose\left(\left[\bigwedge \phi\right] \to \psi, p\right)$

Table 4: Table of moves adapted from Black and Hunter's move to fit Prakken's framework. The first column contains the attacker, the second the object of the attack. Note that it is not true that any speech act in the left column will attack every speech act from the right column that is in the dialogue when moved, this table merely provides the possible speech acts that can be attacked.

Some examples are show below to clarify these definitions. In addition to the informal definition in the table, the sets R_a and R_s are formally defined below the examples.

First, a *propose* speech act and an *open* speech act can be attacked only by *query* speech acts.

$$\begin{tikzcd} & \mathit{propose}(a \to b, p) \arrow[dl, no head] \arrow[dr, no head] & \\ \mathit{query}(a \to b, p) & & \mathit{query}(a, p) \end{tikzcd}$$

$$\begin{tikzcd} \mathit{open}(a) \arrow[d, no head] \\ \mathit{query}(a, \infty) \end{tikzcd}$$

Second, an example of an *assert* attacked by a *propose* speech act.

$$\begin{tikzcd} & b \text{ since } (c,1),(c \to b,1) \arrow[d, no head] & \\ & \mathit{propose}(d \to \neg c, p) \arrow[dl, no head] \arrow[dr, no head] & \\ \mathit{query}(d \to \neg c, p) & & \mathit{query}(d, p) \end{tikzcd}$$

Definition 47. The **attack relation between speech acts** is defined as $R_a =$

$$\left\{ \ (\mathit{assert}(A), \mathit{assert}(A')) \ \middle| \ (A, A') \in \Rightarrow \ \right\}$$

$$\cup$$

$$\left\{ \ (\mathit{assert}(A), \mathit{query}(\phi, p)) \ \middle| \ \begin{array}{l} \mathit{pLevel}(A) \leq p \\ \text{and } \phi = \mathit{conc}(A) \end{array} \right\}$$

$$\cup$$

$$\left\{ \ \left(\mathit{assert}(A), \mathit{query}\left(\left[\bigwedge \phi\right] \to \psi, p\right)\right) \ \middle| \ \begin{array}{l} \mathit{pLevel}(A) \leq p \\ \text{and } \psi = \mathit{conc}(A) \\ \text{and } \left[\bigwedge \phi\right] \to \psi, p' \\ \qquad \in \mathit{prem}(A) \end{array} \right\}$$

$$\cup$$

$$\left\{ \ \left(\mathit{propose}\left(\left[\bigwedge \phi\right] \to \psi, p\right), \mathit{assert}(A)\right) \ \middle| \ \begin{array}{l} \neg\psi = \mathit{conc}(A) \\ \text{and } p = \mathit{pLevel}(A) \end{array} \right\}$$

$$\cup$$

$$\left\{ \ \left(\mathit{propose}\left(\left[\bigwedge \phi\right] \to \psi, p\right), \mathit{query}(\psi', p')\right) \ \middle| \ \begin{array}{l} \psi = \psi' \\ \text{and } p = p' \end{array} \right\}$$

$$\cup$$

$$\left\{ \ \left(\mathit{query}(\phi, p'), \mathit{propose}\left(\left[\bigwedge \phi\right] \to \psi, p\right)\right) \ \middle| \ p = p' \ \right\}$$

$$\cup$$

$$\left\{ \ (\mathit{query}(\phi, \infty), \mathit{open}(\phi')) \ \middle| \ \phi = \phi' \ \right\}$$

$$\cup$$

$$\left\{ \ \left(\mathit{query}\left(\left[\bigwedge \phi\right] \to \psi, p\right), \mathit{propose}\left(\left[\bigwedge \phi\right] \to \psi, p'\right)\right) \ \middle| \ p = p' \ \right\}$$

Note that $\forall(x, y) \in R_a$, $x \in L_c$ and $y \in L_c$.

Since there are no speech acts that can be surrendered to, the set R_s can be empty.

Definition 48. The **surrender relation between speech acts** is defined as $R_s = \emptyset$

5.4 Moves

Moves are defined as they are in Prakken's framework, but with the altered role notation. Extra functions provide information about the moves more easily, for example to get the player without the role.

Definition 49. A **move** m_{id} is defined as a 4-tuple (id, $role(player)$, $perf$ $(content)$, $target$) where

$$\left\| \begin{array}{ll} id & \text{number in the sequence of a dialogue} \in \mathbb{N}_1 \\ role(player) & role(player) \in Roles \\ perf(content) & perf(content) \in L_c \\ target & \text{target of the move} \in \mathbb{N}_1 \end{array} \right.$$

The following functions return the various elements of a move.

$$\left\| \begin{array}{ll} id(m_{id}) & = id \\ pl(m_{id}) & = role(player) \\ & = \left\| \begin{array}{ll} Role(pl(m_{id})) & = role \\ Player(pl(m_{id})) & = player \end{array} \right. \\ s(m_{id}) & = perf(content) \\ & \quad \left\| \begin{array}{ll} Performative(s(m_{id})) & = perf \\ Content(s(m_{id})) & = content \end{array} \right. \\ t(m_{id}) & = target \end{array} \right.$$

5.5 Commitment function

The commitment function needs to be defined for the adapted system. Recall that in Black and Hunter's system, only assert moves influence the commitment store. The same is true in this adaption, so the C function is defined accordingly.

Definition 50. The **commitment function** is defined as $C : M^{\leq\infty} \times \{P, O\} \rightarrow \mathcal{P}(L_t)$. $C_d(p)$ denotes the commitments a role has made in a dialogue d.

$$C_{d_t}(p) = \begin{cases} \emptyset & \text{iff } d_t = \emptyset (t = 0) \\ C_{d_{t-1}}(p) \cup \Phi & \text{iff } s(m_t) = assert(\phi \text{ since } \Phi) \text{ and} \\ & p = Role(pl(m_t)) \\ C_{d_{t-1}}(p) & \text{otherwise.} \end{cases}$$

where $d_t = m_1 \dots m_t$.

5.6 Turntaking

Turntaking is one of the functions where the distinction between roles and players is quite important. The turntaking function specifically talks about what we have defined as roles. Therefore, by allowing both P and O to move at any time through the turntaking function, players get the freedom to move as whatever role is convenient.

Definition 51. The **turntaking function** is defined as $T : M^{<\infty} \rightarrow \mathcal{P}(\{P, O\})$.

$$T(d) = \begin{cases} \{P(x)\} & \text{iff } d = \emptyset \text{, where } P(x) \in \mathit{Roles} \\ \{P(x), O(x)\} & \text{otherwise, where } P(x), O(x) \in \mathit{Roles} \end{cases}$$

Note that this function forces the first move to be made by P. If this was not a restriction made by Prakken's framework, the *open* speech act could have been omitted, allowing a warrant inquiry dialogue to be opened with a *query* move made by O.

5.7 Protocol

Some moves are tightly connected, and as such, limitations on these moves are required; these are noted in the protocol additions. First, Prakken's protocol conditions are repeated below for convenience. Note that while the notation of the protocol P_{Pr} was changed to Π above, the protocol function Pr will still be used.

Definition 34, repeated[22]**.** $\forall d, m$ if $m \in Pr(d)$ then

R_1 $pl(m) \in T(d)$

R_2 if $d \neq d_0$ and $m \neq m_1$ then $s(m)$ is a reply to $s(t(m))$ according to L_c

R_3 $\forall m' \in d$ if $t(m) = m'$ then $pl(m) \neq pl(m')$

R_4 $\forall m' \in d$ if $t(m) = t(m')$ then $s(m) \neq s(m')$

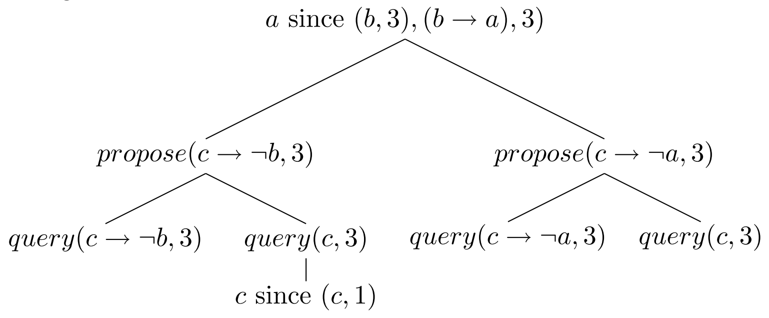
R_5 $\forall m' \in d$ if $t(m) = t(m')$ and $s(m')$ is a surrendering reply, then m is not an attacking counterpart of m'

Several protocol rules have to be added to Prakken's rules. First, only *query* moves may follow an *open* or *propose* move, until all required *query* moves have

[22]Originally defined on page 31

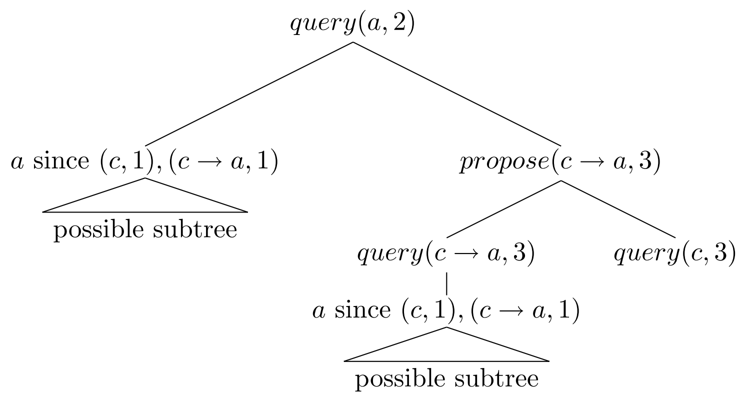
been made. Second, the *open* may only be moved as the first move of a dialogue.

There are also several problems which stem from the differences between the approaches of Black and Hunter's system and Prakken's framework. For example, an asserted argument in Black and Hunter's system can effectively be a counterargument to several arguments, where Prakken's framework explicitly allows only a single target per move. An example dialogue tree will clarify this problem. Note that for assert moves, only the content of the speech act is show in the dialogue tree.


a since $(b, 3), (b \rightarrow a), 3)$
$propose(c \rightarrow \neg b, 3)$
$propose(c \rightarrow \neg a, 3)$
$query(c \rightarrow \neg b, 3)$
$query(c, 3)$
c since $(c, 1)$
$query(c \rightarrow \neg a, 3)$
$query(c, 3)$

In order to defeat both $query(c, 3)$ moves, $assert(c$ since $(c, 1))$ has to be moved twice. Prakken's framework allows the repetition of moves as long as they do not attack the same move (see R_4), and the adaption to Black and Hunter's inquiry dialogue system has to do the same to allow the outcome of the dialogue above to be the same as it would be in Black and Hunter's system.

This change provides a new problem. While dialogues in Black and Hunter's article assert arguments without opening an argument inquiry dialogue, these are often followed by an argument inquiry dialogue to inquire about other possible arguments that can be constructed. Allowing repetition results in the following dialogue tree:



$query(a, 2)$
a since $(c, 1), (c \rightarrow a, 1)$
possible subtree
$propose(c \rightarrow a, 3)$
$query(c \rightarrow a, 3)$
$query(c, 3)$
a since $(c, 1), (c \rightarrow a, 1)$
possible subtree

Because the $assert(a$ since $(c, 1), (c \rightarrow a, 1))$ has been moved before the *propose* $(c \rightarrow a, 3)$ was moved, it is also moved inside the argument inquiry dialogue opened with the $propose(c \rightarrow a, 3)$. This would cause an entire part of the dialogue to be repeated, while having no effect whatsoever on the status of the nodes above the $propose(c \rightarrow a, 3)$ move.

The status of $assert(a$ since $(c, 1), (c \rightarrow a, 1))$ and $propose(c \rightarrow a, 3)$ both influence the status of $query(a, 2)$, but the status of $assert(a$ since $(c, 1), (c \rightarrow a, 1))$ also determines the status of $propose(c \rightarrow a, 3)$. If $assert(a$ since $(c, 1), (c \rightarrow a, 1))$ is undefeated, $propose(c \rightarrow a, 3)$ has no influence higher in the tree, if $assert(a$ since $(c, 1), (c \rightarrow a, 1))$ is defeated, $propose(c \rightarrow a, 3)$ is as well.

In order to avoid this, a condition has to be added to the protocol for use of the *assert* speech act: all of the facts in the premises of the asserted argument must have been asserted earlier, if the premises of the argument contain a rule. This condition ensures that all arguments are constructed properly through an argument inquiry dialogue.

Since the *propose* moves are moved anyway, the worst consequence of this change is that that number of moves increases, as compared to the original dialogue. Compared to the tree above, however, where entire parts of the tree can end up being repeated, the number of moves is certainly decreased. The same dialogues can be expressed, the moves are just shuffled around slightly (and split into separate assert moves for their facts). This will be more clear when the example dialogue on page 27 is converted on page 71.

To summarise, the following additions are made to the protocol rules:

Query moves Only *query* moves may follow an *open* or *propose* move, until all required *query* moves have been made;[23] (Added as protocol rule R_6)

[23]Note that while these may be made by any player (but not any role!), it is assumed in

Open and propose The *open* move may only be moved as the first move of a dialogue, and the first move of a dialogue may only be *open* or *propose* ; (Added as protocol rule R_7)

Complete arguments All of the facts in the premises of an asserted argument must have been asserted earlier, if the premises of the argument contain a rule; (Added as protocol rule R_8)

The changes to the protocol are now formally defined.

Definition 52. The adapted inquiry dialogue system adds three protocol rules to those of Prakken's framework.

If $m_t \in Pr(d)$ and $d = m_1 \ldots m_{t-1}$, then:

R_6 if $\mathit{Performative}(s(m_{t-1})) = \mathit{propose}$
or $\mathit{Performative}(s(m_{t-1})) = \mathit{open}$
then $t(m_t) = m_{t-1}$ and $(s(m_t), s(m_{t-1})) \in R_a$
else if

$$\left[\begin{array}{l} \mathit{Performative}(s(m_{t-1})) = \mathit{query} \\ \text{and } t(m_{t-1}) = m_u \\ \text{and } n = \text{number of unique literals in } s(m_u) \\ \text{and } [n - (t-1) + u] > 0 \end{array}\right.$$

then

$$\left[\begin{array}{l} t(m_t) = m_u \\ \text{and } (s(m_t), s(m_u)) \in R_a \end{array}\right.$$

R_7 if $d = d_0$ then $\mathit{Performative}(s(m_t)) = \mathit{open}$ or $\mathit{Performative}(s(m_t)) = \mathit{propose}$
and if $\mathit{Performative}(s(m_t)) = \mathit{open}$ then $d = d_0$

R_8 if

$$\left[\begin{array}{l} s(m_t) = \mathit{assert}(A) \text{ and} \\ \exists r \in \mathcal{R}^*, p \in \mathbb{N}_1 \\ \quad \text{such that } (r, p) \in \mathit{prem}(A) \end{array}\right.$$

then

$$\left[\begin{array}{l} \forall \phi \in \mathcal{S}^*, p' \in \mathbb{N}_1 \\ \text{if } (\phi, p') \in \mathit{prem}(A) \\ \text{then } \exists m_u \in d \text{ such that} \\ s(m_u) = \mathit{assert}(\phi \text{ since } (\phi, p)) \end{array}\right.$$

the examples that the player moving the *propose* also moves the *query* acts. This is, however, not a requirement.

A small example dialogue is provided below to elaborate on R_6 and R_7.

Role$_{\text{Turn}}$(Player)	Performative(Content)	Target
$P_1(1):$	$propose(a \wedge b \rightarrow c)$	0
$O_2(1):$	$query(a \wedge b \rightarrow c)$	1
$O_3(1):$	$query(a)$	1
$O_4(1):$	$query(b)$	1

Table 5: An example for R_6

If $t = 1$, $d = d_0$, so either *open* or *propose* can be moved. However, *open* can only be moved as the first move. In the example above, *propose* was moved. If $t = 2$, $s(m_{t-1}) = propose$, so m_t must be a *query* move. If $t = 4$, $s(m_{t-1}) = query$. The target of m_{t-1} is m_1 and the number of literals in $a \wedge b \rightarrow c$ is 3. Since $n - (t-1) + u = 3 - 3 + 1 = 1$, which is larger than 0, another *query* move is required. If $t = 5$, $n - (t-1) + u = 3 - 4 + 1 = 0$, so no more *query* moves are required. Note that since no two attackers can have the same speech act, this automatically forces all correct *query* moves to be moved.

5.7.1 Well-formed dialogues

A well-formed dialogue is a dialogue that is correct according to the protocol, in which every required *query* move has been moved (so if the last move was a *propose* , the dialogue is not well-formed until all *query* moves are moved) and in which speech acts are moved against every target that is available for the speech acts.

Definition 53. A dialogue d is **well-formed** iff

- $d \in \Pi$
- $\nexists m_i \in Pr(d)$ such that $Performative(s(m_i)) = query$
- $\nexists m_i \in Pr(d)$ such that $\exists m_j \in d$ where $s(m_i) = s(m_j)$

If a dialogue is not well-formed, the outcome can be unreliable. For example, a dialogue that is not well-formed may have ended just after a *propose* move was made. Because the *query* moves have not been moved, the *propose* move is undefeated, allowing the *propose* move to defeat other moves.

If a speech act can be moved that is already present in the dialogue, there are apparently more moves that it can reply to. In order for the outcome of the dialogue to be correct given te information inside the dialogue, the speech act must be used against every possible move it can target.

5.7.2 Protocol additions based on underlying logic

As discussed earlier, there are few properties that Black and Hunter's system for inquiry dialogues require of the underlying logic. It was shown that the system they chose, DeLP, has some disadvantages when compared to other systems.

This adapted system does not specifically consider DeLP or alternate underlying logics, as any underlying logic can be used with the adapted system, simply by adding protocol rules to match the logic. For example, if DeLP were to be used, a protocol rule would be added to enforce acceptable argumentation lines.

It must be noted that an assumption that is made by Black and Hunter is that defeasible rules consist of a conjunction of literals that imply a single literal. This has to be reflected in any underlying logic, though it is possible that allowing complex structures in place of the literals in Black and Hunter's system will not affect it at all.

5.8 Outcome

As in Prakken's framework, the status of the nodes in the dialogue tree determines whether the first move is defeated or undefeated, and as such determines whether the topic of the dialogue is warranted. Technically the two inquiry dialogue types in Black and Hunter's system are still allowed, and each has a different outcome; starting with an *open* move to create a warrant inquiry dialogue or starting with a *propose* move to create an argument inquiry dialogue.

In order to get an outcome on the basis of the last move that has been made, a definition for the last relevant move is added.

Definition 54. The **last relevant move** *lastRelevant* in a dialogue d is defined as:

$$lastRelevant(d) = \begin{cases} m_i & \text{iff } m_i \in d \text{ and} \\ & Performative(s(m_i)) = assert \text{ and} \\ & \text{the status of } m_1 \in d \neq \text{the status of } m_1 \in d_i \\ & \text{and } \nexists m_j \in d \text{ such that} \\ & \quad \left[\begin{array}{l} Performative(s(m_j)) = assert \text{ and} \\ \text{the status of } m_1 \in d \neq \text{the status of } m_1 \in d_j \\ id(m_j) > id(m_i) \end{array} \right. \end{cases}$$

Using relevant moves, it is possible to determine the last change to the status of the topic of the dialogue; iff the last relevant move that was made changed the status of the topic to defeated, it must have been moved by the *Opponent*. The other way around, this mean that if a role made the last relevant move, it is the winning role.

Definition 55. The **winner of an inquiry dialogue** d is defined as a function $Winner : \mathcal{P}(M) \mapsto \{P, O\}$.

$$Winner(d) = \begin{cases} P & \text{iff } Role(pl(lastRelevant(d))) = P \\ O & \text{otherwise.} \end{cases}$$

Based on the winner, the outcome for the dialogue can be determined.

Definition 56. The **outcome of an inquiry dialogue** d is defined as a function $Outcome : \mathcal{P}(M) \mapsto \mathcal{P}(Args \cup \mathcal{S}^*)$.

$Outcome(d) = \emptyset$ iff $Winner(d) = O$, otherwise

$$\begin{array}{l} \left\{ \;\phi \;\middle|\; s(m_1) = open(\phi)\; \right\} \\ \cup \\ \left\{ \begin{array}{c|l} A & s(m_1) = propose\left(\left[\bigwedge \phi\right] \to \psi, p\right) \\ & \exists m_i, m_j \text{ such that} \\ & \quad \left[\begin{array}{l} t(m_i) = 1 \\ t(m_j) = i \\ s(m_i) = query\left(\left[\bigwedge \phi\right] \to \psi, p\right) \\ s(m_j) = assert(A) \end{array} \right. \end{array} \right\} \end{array}$$

Note that the separate parts of the outcome result represent the different types of dialogues in Black and Hunter's system. Warrant inquiry dialogues start with an *open* move and return a set with the single literal that the dialogue was opened with, if it is warranted. Argument inquiry dialogues start with a *propose* move and return a set of all the arguments moved to support the topic of the dialogue, if any could be found.

5.9 Dialectical tree and graph

Black and Hunter use a dialectical tree to determine the outcome of the dialogue. Additionally, Prakken's framework allows the construction of a dialectical graph. Since these are somewhat similar constructs, this chapter will discuss both in relation to the adapted system.

5.9.1 Dialectical tree

The dialectical tree is a tree of the arguments moved in an inquiry dialogue, as defined in Definition 12. Since Black and Hunter's system has only three types of moves, of which only one contributes to the content of the dialogue, it is relatively straightforward to build a dialectical tree in their system. In the adapted system, only two alterations to the dialogue tree are required to build a dialectical tree. The algorithm defined below can be used to construct the dialectical tree from the dialogue tree.

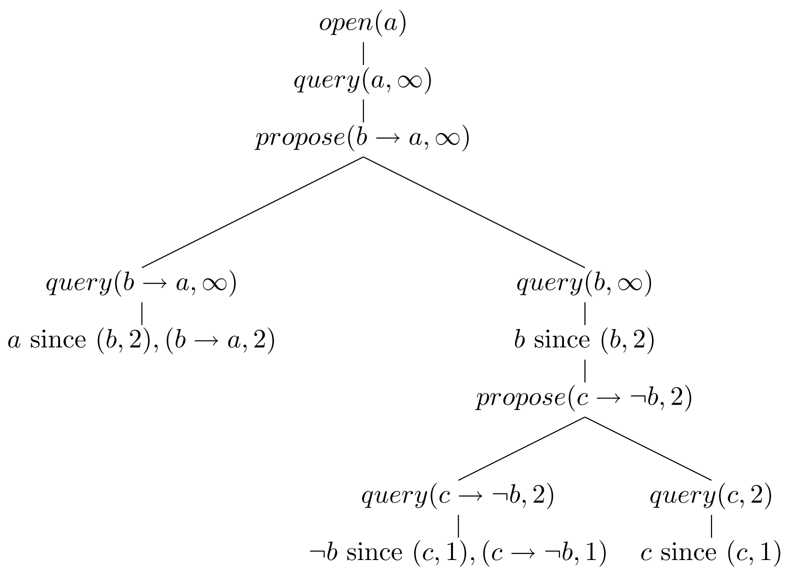
Definition 57. Given a dialogue, the algorithm **dialecticalTree** is defined below:

```
dialecticalTree(d) =
    First
        If $Performative(s(m_1)) = open$
        then drop $m_1$ and $m_2$ from $d$
        and $\forall m_i \in d$ where $t(m_i) = 2, t(m_i) = 0$
    then while
        $\exists m_j \in d$ such that $Performative(s(m_j)) = propose$
        and $\forall m_k \in d$ if $Performative(s(m_j)) = propose$,
        $id(m_j) > id(m_k)$
    do
        $\forall m_u, m_v \in d$ such that $t(m_u) = v$ and $t(m_v) = j$
        if $s(m_v) = query\left(\left[\bigwedge \phi\right] \to \psi, p\right)$
        then
            add new node $m_n$
            with $Content(s(m_u))$ as label
            as a child of $t(m_j)$
            and for all children $m_w$ of the node $m_u$
            make $m_w$ a child of $m_n$
        and
            $\forall m_p, m_q, m_r \in d$
            where $t(m_p) = q$ and $t(m_q) = r$ and $t(m_r) = j$
            if $s(m_v) \neq query\left(\left[\bigwedge \phi\right] \to \psi, p\right)$
            and $Content(s(m_p))$ is an argument
            that defeats $Content(s(m_n))$
            then $t(m_p) = id(m_n)$
            and drop $m_q$ and $m_r$ from $d$
        and drop $m_j$, $m_u$ and $m_v$ from $d$
```

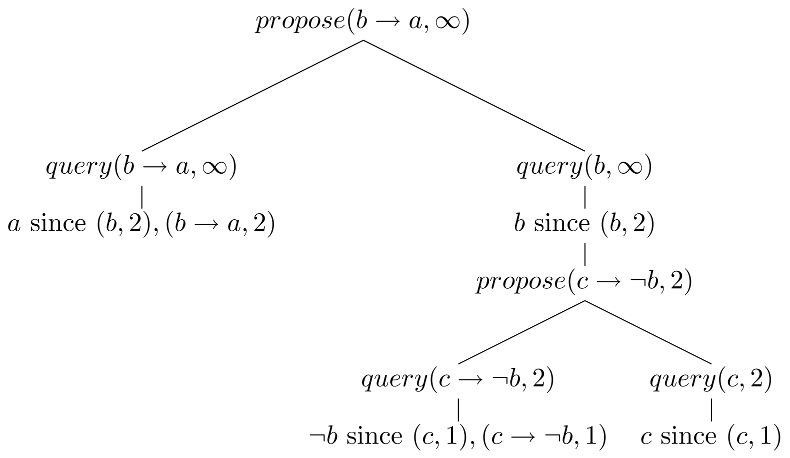
First, the *open* move and the *query* move that replies to it are removed, if they are present at all. Second, moving upwards from the leaves (since we're moving through *propose* moves from the highest to the lowest index), any argument inquiry dialogue[24] is reduced to the argument moved within it. In the case that it contains several arguments, these become separate replies. For example:

[24] Recall that this is a *propose* move, its attacking *query* moves and all moves that *directly* attack the *query* moves, for example

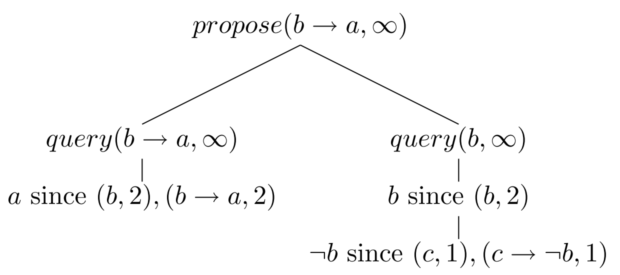
$$\begin{tikzcd}
& propose(b \to a, \infty) \arrow[dl, no head] \arrow[dr, no head] & \\
query(b \to a, \infty) \arrow[d, no head] & & query(b, \infty) \arrow[d, no head] \\
a \text{ since } (b,2), (b \to a, 2) & & b \text{ since } (b,2)
\end{tikzcd}$$


$open(a)$
$query(a, \infty)$
$propose(b \rightarrow a, \infty)$
$query(b \rightarrow a, \infty)$
a since $(b, 2), (b \rightarrow a, 2)$
$query(b, \infty)$
b since $(b, 2)$
$propose(c \rightarrow \neg b, 2)$
$query(c \rightarrow \neg b, 2)$
$\neg b$ since $(c, 1), (c \rightarrow \neg b, 1)$
$query(c, 2)$
c since $(c, 1)$

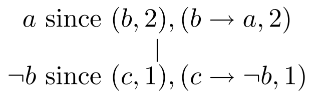
This tree would first be reduced to the tree below by removing the *open* and its *query* move.


$propose(b \rightarrow a, \infty)$
$query(b \rightarrow a, \infty)$
a since $(b, 2), (b \rightarrow a, 2)$
$query(b, \infty)$
b since $(b, 2)$
$propose(c \rightarrow \neg b, 2)$
$query(c \rightarrow \neg b, 2)$
$\neg b$ since $(c, 1), (c \rightarrow \neg b, 1)$
$query(c, 2)$
c since $(c, 1)$

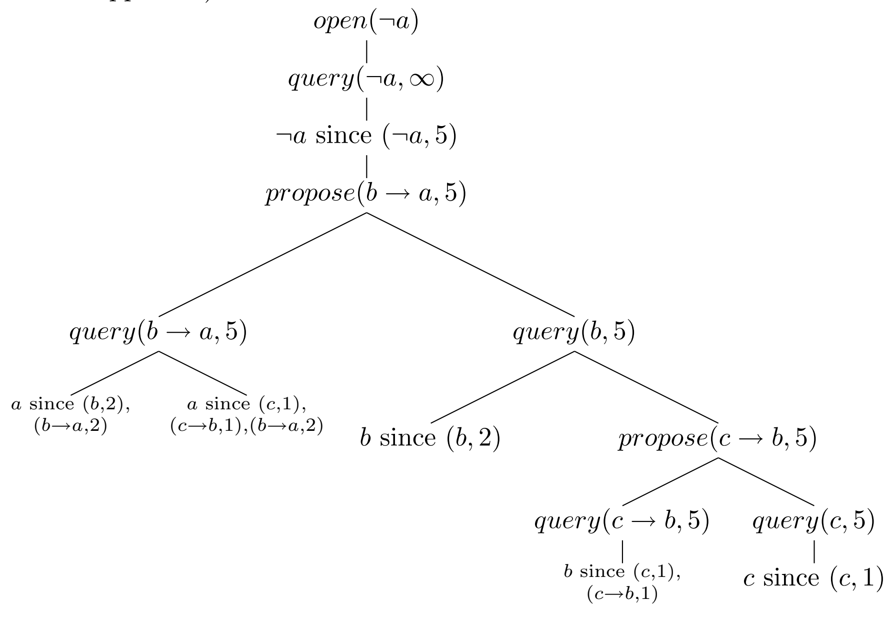
Next, the argument inquiry about $c \rightarrow \neg b$ is reduced to $\neg b$ since $(c, 1), (c \rightarrow \neg b, 1)$.


$propose(b \to a, \infty)$
$query(b \to a, \infty)$
$query(b, \infty)$
a since $(b, 2), (b \to a, 2)$
b since $(b, 2)$
$\neg b$ since $(c, 1), (c \to \neg b, 1)$

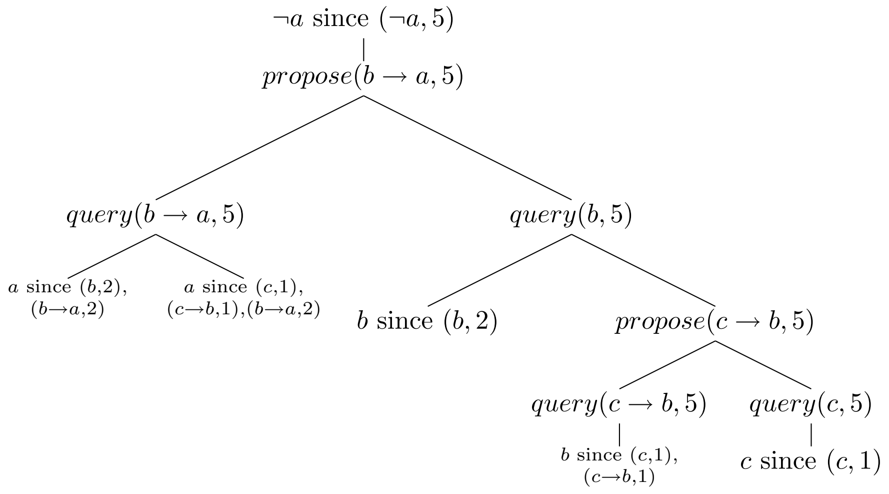
And finally the argument inquiry about $b \to a$ is reduced to a since $(b, 2), (b \to a, 2)$. Note that $\neg b$ since $(c, 1), (c \to \neg b, 1)$ is outside this argument inquiry dialogue, therefore it now targets the argument that the *propose* was reduced to.

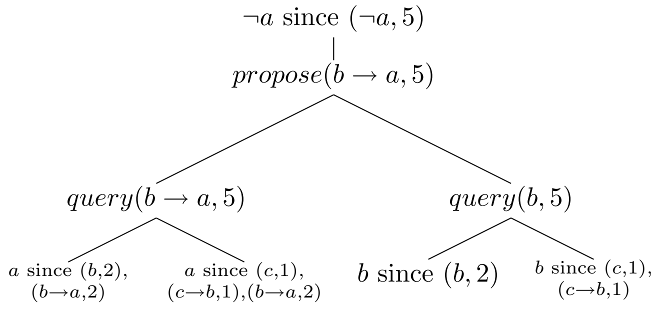

a since $(b, 2), (b \to a, 2)$
$\neg b$ since $(c, 1), (c \to \neg b, 1)$

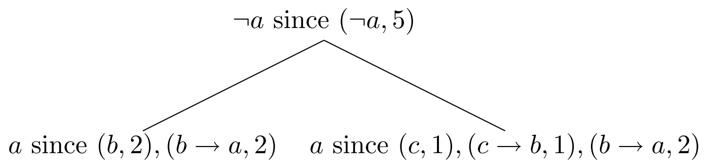
If one of the argument inquiry dialogues had contained a second *assert* move using the rule of the propose move, it would split the tree (note that in the tree below c supports b):



In this dialogue tree, the second argument inquiry dialogue is contained within the first. Also, there are two moves replying to the *query* move with the rule. When reduced fully, this give the dialectical tree in three steps:


$\neg a$ since $(\neg a, 5)$
$propose(b \rightarrow a, 5)$
$query(b \rightarrow a, 5)$
$query(b, 5)$
a since $(b,2)$, $(b \rightarrow a,2)$
a since $(c,1)$, $(c \rightarrow b,1),(b \rightarrow a,2)$
b since $(b, 2)$
$propose(c \rightarrow b, 5)$
$query(c \rightarrow b, 5)$
$query(c, 5)$
b since $(c,1)$, $(c \rightarrow b,1)$
c since $(c, 1)$


$\neg a$ since $(\neg a, 5)$
$propose(b \rightarrow a, 5)$
$query(b \rightarrow a, 5)$
$query(b, 5)$
a since $(b,2)$, $(b \rightarrow a,2)$
a since $(c,1)$, $(c \rightarrow b,1),(b \rightarrow a,2)$
b since $(b, 2)$
b since $(c,1)$, $(c \rightarrow b,1)$


$\neg a$ since $(\neg a, 5)$
a since $(b, 2), (b \rightarrow a, 2)$
a since $(c, 1), (c \rightarrow b, 1), (b \rightarrow a, 2)$

5.9.2 Dialectical graph

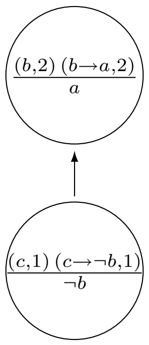
The dialectical graph is a construct similar to the dialectical tree. In Prakken's framework, arguments are extended by replying to the *why* moves. This process allows an argument to be extended at multiple positions, effectively creating multiple arguments which are not clearly distinguishable. The dialectical graph displays these different arguments as different nodes in the graph.

However, as was discussed earlier, Black and Hunter do not use argument extension as defined by Prakken; instead, arguments are built inside argument inquiry dialogues. If an argument is provided with additional premises, this is a new argument separate from the original argument. Thus, the distinction between dialectical graph and dialectical tree becomes one of notation: Prakken denotes arguments in the dialectical graph in the form $\frac{a}{b}$.

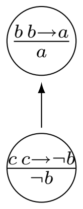
Therefore, in order to create a dialectical graph from a dialogue tree, it must first be converted to a dialectical tree, and then the arguments will be rewritten to fit Prakken's notation. The two dialectical trees created above is now converted to dialectical graphs:

$$\begin{tikzcd}
a \text{ since } (b,2),(b \to a,2) \arrow[d, no head] \\
\neg b \text{ since } (c,1),(c \to \neg b,1)
\end{tikzcd}$$

becomes:

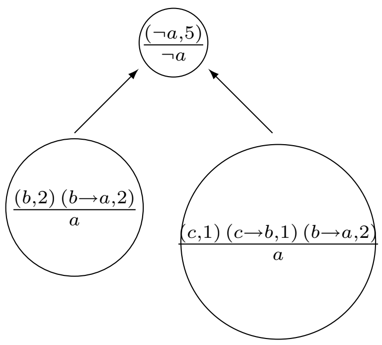


or

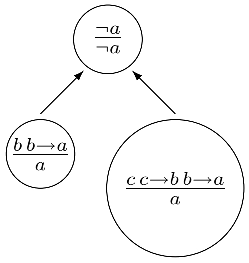


$$\begin{tikzcd}
 & \neg a \text{ since } (\neg a,5) \arrow[dl, no head] \arrow[dr, no head] & \\
a \text{ since } (b,2),(b \to a,2) & & a \text{ since } (c,1),(c \to b,1),(b \to a,2)
\end{tikzcd}$$

becomes:


$\frac{(\neg a,5)}{\neg a}$
$\frac{(b,2)\ (b\rightarrow a,2)}{a}$
$\frac{(c,1)\ (c\rightarrow b,1)\ (b\rightarrow a,2)}{a}$

or


$\frac{\neg a}{\neg a}$
$\frac{b\ b\rightarrow a}{a}$
$\frac{c\ c\rightarrow b\ b\rightarrow a}{a}$

5.10 Example dialogues

Three example dialogues from [3] are adapted according to the changes described in this section. The first is a warrant inquiry dialogue from page 27.

Role$_{\text{Turn}}$(Player)	Performative(Content)	Target
$P_1(1)$:	$open(b)$	0
$O_2(1)$:	$query(b, \infty)$	1
$P_3(1)$:	$propose(a \rightarrow b, \infty)$	2
$O_4(1)$:	$query(a \rightarrow b, \infty)$	3
$O_5(1)$:	$query(a, \infty)$	3
$P_6(1)$:	$assert(a \text{ since } (a, 4))$	5
$P_7(1)$:	$assert(b \text{ since } (a, 4), (a \rightarrow b, 4))$	4
$O_8(2)$:	$propose(d \rightarrow \neg a, 4)$	6
$P_9(2)$:	$query(d \rightarrow \neg a, 4)$	8
$P_{10}(2)$:	$query(d, 4)$	8
$O_{11}(2)$:	$assert(d \text{ since } (d, 3))$	10
$O_{12}(2)$:	$assert(\neg a \text{ since } (d, 3), (d \rightarrow \neg a, 3))$	9
$P_{13}(2)$:	$assert(\neg d \text{ since } (\neg d, 1))$	11
$O_{14}(1)$:	$propose(c \rightarrow \neg b, 4)$	7
$P_{15}(1)$:	$query(c \rightarrow \neg b, 4)$	14
$P_{16}(1)$:	$query(c, 4)$	14
$O_{17}(1)$:	$assert(c \text{ since } (c, 3))$	16
$O_{18}(1)$:	$assert(\neg b \text{ since } (c, 3), (c \rightarrow \neg b, 3))$	15
$P_{19}(2)$:	$propose(e \rightarrow \neg d, 3)$	11
$O_{20}(2)$:	$query(e \rightarrow \neg d, 3)$	19
$O_{21}(2)$:	$query(e, 3)$	19
$P_{22}(1)$:	$assert(e \text{ since } (e, 2))$	21
$P_{23}(2)$:	$assert(\neg d \text{ since } (e, 2), (e \rightarrow \neg d, 2))$	20
$O_{24}(2)$:	$assert(\neg e \text{ since } (\neg e, 1))$	22

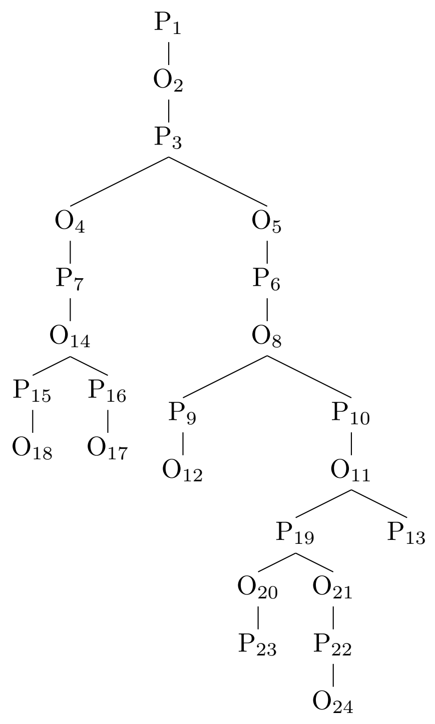
Table 6: An adapted warrant inquiry dialogue in Prakken's framework

To show the difference protocol rule R_8 makes, below is part of the dialogue if R_8 was not in the protocol. It resembles the original warrant inquiry dialogue example in [3] and on page 27 more closely.

Role$_{\text{Turn}}$(Player)	Performative(Content)	Target
$P_1(1):$	$open(b)$	0
$O_2(1):$	$query(b, \infty)$	1
$P_3(1):$	$assert(b$ since $(a, 4), (a \rightarrow b, 4))$	2
$O_4(2):$	$assert(\neg a$ since $(d, 3), (d \rightarrow \neg a, 3))$	3
$O_5(1):$	$assert(\neg b$ since $(c, 3), (c \rightarrow \neg b, 3))$	3
$P_6(2):$	$assert(\neg d$ since $(\neg d, 1))$	4
$P_7(1):$	$propose(a \rightarrow b, \infty)$	2
$O_8(1):$	$query(a \rightarrow b, \infty)$	7
$O_9(1):$	$query(a, \infty)$	7
$O_{10}(2):$	$propose(d \rightarrow \neg a, 4)$	3
$P_{11}(2):$	$query(d \rightarrow \neg a, 4)$	10
$P_{12}(2):$	$query(d, 4)$	10
...	...	

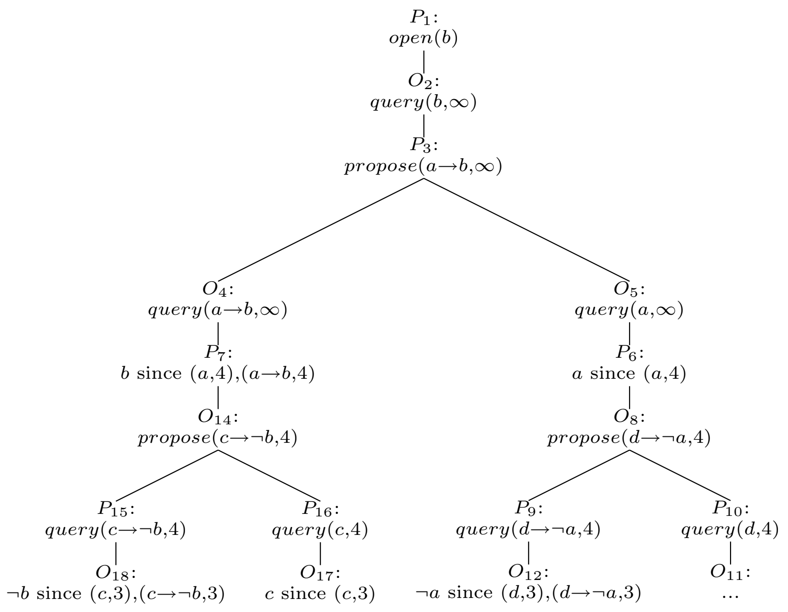
Table 7: An adapted warrant inquiry dialogue in Prakken's framework without using R_8 in the protocol

The dialogue in Table 6 would produce the dialogue tree shown below.

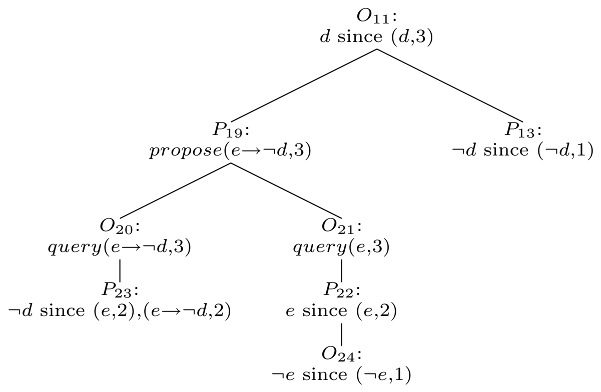


For readability and formatting reasons, the dialogue tree with the content of the moves[25] has been split into its subtrees, which will follow below the tree.

[25]As Prakken does with argue moves, for assert moves the performative is not shown, only the argument.


P_1: $open(b)$
O_2: $query(b,\infty)$
P_3: $propose(a\rightarrow b,\infty)$
O_4: $query(a\rightarrow b,\infty)$
O_5: $query(a,\infty)$
P_7: b since $(a,4),(a\rightarrow b,4)$
P_6: a since $(a,4)$
O_{14}: $propose(c\rightarrow\neg b,4)$
O_8: $propose(d\rightarrow\neg a,4)$
P_{15}: $query(c\rightarrow\neg b,4)$
P_{16}: $query(c,4)$
P_9: $query(d\rightarrow\neg a,4)$
P_{10}: $query(d,4)$
O_{18}: $\neg b$ since $(c,3),(c\rightarrow\neg b,3)$
O_{17}: c since $(c,3)$
O_{12}: $\neg a$ since $(d,3),(d\rightarrow\neg a,3)$
O_{11}: ...

$O_{11}\ldots$:

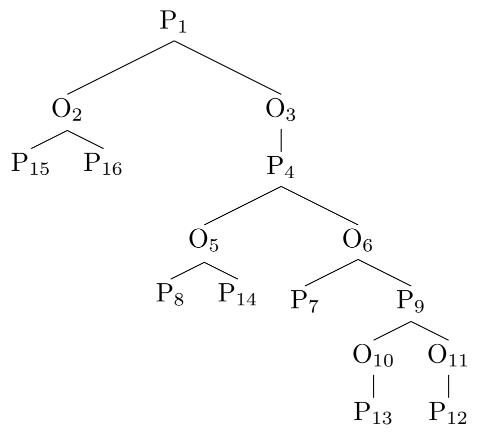

O_{11}: d since $(d,3)$
P_{19}: $propose(e\rightarrow\neg d,3)$
P_{13}: $\neg d$ since $(\neg d,1)$
O_{20}: $query(e\rightarrow\neg d,3)$
O_{21}: $query(e,3)$
P_{23}: $\neg d$ since $(e,2),(e\rightarrow\neg d,2)$
P_{22}: e since $(e,2)$
O_{24}: $\neg e$ since $(\neg e,1)$

Next, the argument inquiry dialogue from page 26 is shown with the adaptions. Note that the original example seems to be incomplete; two moves that are possible at different stages in the dialogue but are not present in the original example, they are present here as moves 14 and 16.

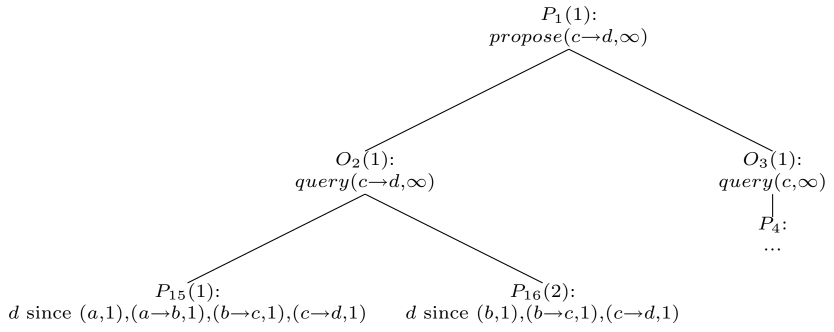
Role$_{\text{Turn}}$(Player)	Performative(Content)	Target
$P_1(1)$:	$propose(c \rightarrow d, \infty)$	0
$O_2(1)$:	$query(c \rightarrow d, \infty)$	1
$O_3(1)$:	$query(c, \infty)$	1
$P_4(1)$:	$propose(b \rightarrow c, \infty)$	3
$O_5(1)$:	$query(b \rightarrow c, \infty)$	4
$O_6(1)$:	$query(b, \infty)$	4
$P_7(2)$:	$assert(b \text{ since } (b, 1))$	6
$P_8(1)$:	$assert(c \text{ since } (b, 1), (b \rightarrow c, 1))$	5
$P_9(1)$:	$propose(a \rightarrow b, \infty)$	6
$O_{10}(1)$:	$query(a \rightarrow b, \infty)$	9
$O_{11}(1)$:	$query(a, \infty)$	9
$P_{12}(2)$:	$assert(a \text{ since } (a, 1))$	11
$P_{13}(1)$:	$assert(b \text{ since } (a, 1), (a \rightarrow b, 1))$	10
$P_{14}(2)$:	$assert(c \text{ since } (a, 1), (a \rightarrow b, 1), (b \rightarrow c, 1))$	5
$P_{15}(1)$:	$assert(d \text{ since } (a, 1), (a \rightarrow b, 1), (b \rightarrow c, 1), (c \rightarrow d, 1))$	2
$P_{16}(2)$:	$assert(d \text{ since } (b, 1), (b \rightarrow c, 1), (c \rightarrow d, 1))$	2

Table 8: An adapted argument inquiry dialogue in Prakken's framework

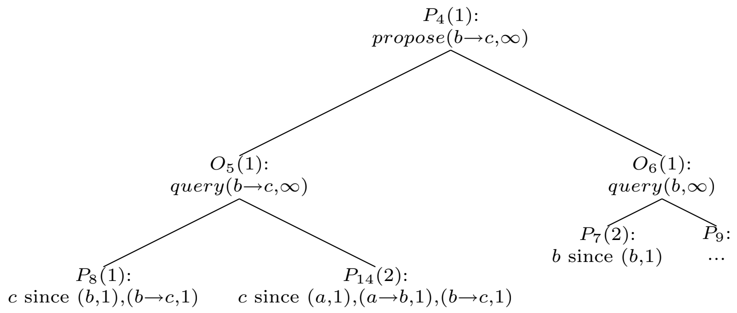
The above dialogue would produce the dialogue tree shown below.



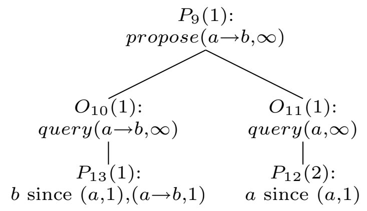
Again, for readability and formatting reasons, the dialogue tree with the content of the moves has been split.


$P_1(1)$: $propose(c{\to}d,\infty)$
$O_2(1)$: $query(c{\to}d,\infty)$
$O_3(1)$: $query(c,\infty)$
P_4: …
$P_{15}(1)$: d since $(a,1),(a{\to}b,1),(b{\to}c,1),(c{\to}d,1)$
$P_{16}(2)$: d since $(b,1),(b{\to}c,1),(c{\to}d,1)$

$P_4 \ldots$:


$P_4(1)$: $propose(b{\to}c,\infty)$
$O_5(1)$: $query(b{\to}c,\infty)$
$O_6(1)$: $query(b,\infty)$
$P_7(2)$: b since $(b,1)$
P_9: …
$P_8(1)$: c since $(b,1),(b{\to}c,1)$
$P_{14}(2)$: c since $(a,1),(a{\to}b,1),(b{\to}c,1)$

$P_9 \ldots$:


$P_9(1)$: $propose(a{\to}b,\infty)$
$O_{10}(1)$: $query(a{\to}b,\infty)$
$O_{11}(1)$: $query(a,\infty)$
$P_{13}(1)$: b since $(a,1),(a{\to}b,1)$
$P_{12}(2)$: a since $(a,1)$

Finally, the medical example discussed in [3] will be converted.

$$\begin{array}{ll} \Sigma^1 & = \{(pain, 1), (nonCyc, 1), (preg, 2)\} \\ \Sigma^2 & = \{(pain \land nonCyc \to mam, 3), (preg \to \neg mam, 1)\} \end{array}$$

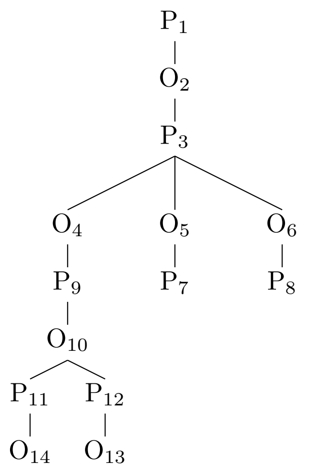
Role$_{\text{Turn}}$(Player)	Performative(Content)	Target
$P_1(2)$:	$open(mam)$	0
$O_2(2)$:	$query(mam, \infty)$	1
$P_3(2)$:	$propose(pain \land nonCyc \to mam, \infty)$	2
$O_4(2)$:	$query(pain \land nonCyc \to mam, \infty)$	3
$O_5(2)$:	$query(pain, \infty)$	3

Continued on Next Page…

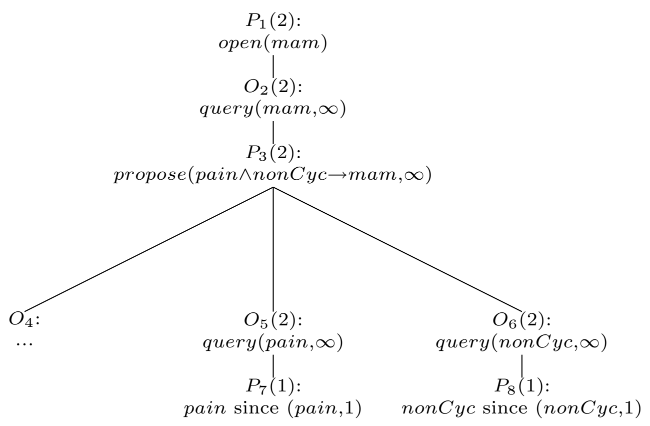
Role$_{\text{Turn}}$(Player)	Performative(Content)	Target
$O_6(2)$:	$query(nonCyc, \infty)$	3
$P_7(1)$:	$assert(pain$ since $(pain, 1))$	5
$P_8(1)$:	$assert(nonCyc$ since $(nonCyc, 1))$	6
$P_9(2)$:	$assert(mam$ since $(pain, 1), (nonCyc, 1), (pain \wedge nonCyc \rightarrow mam, 3))$	4
$O_{10}(2)$:	$propose(preg \rightarrow \neg mam, 3)$	9
$P_{11}(2)$:	$query(preg \rightarrow \neg mam, 3)$	10
$P_{12}(2)$:	$query(preg, 3)$	10
$O_{13}(1)$:	$assert(preg$ since $(preg, 2))$	12
$O_{14}(2)$:	$assert(\neg mam$ since $(preg, 2), (preg \rightarrow \neg mam, 1))$	11

Table 9: An example from the medical domain, adapted to fit in Prakken's framework

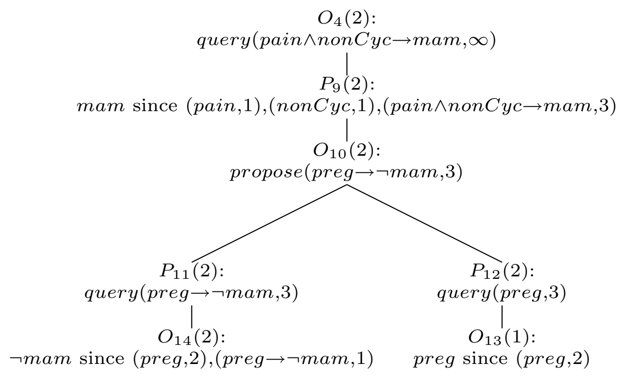
The above dialogue would produce the dialogue tree shown below.



Again, for readability and formatting reasons, the dialogue tree with the content of the moves has been split.


$P_1(2)$: $open(mam)$
$O_2(2)$: $query(mam,\infty)$
$P_3(2)$: $propose(pain \land nonCyc \rightarrow mam,\infty)$
O_4: ...
$O_5(2)$: $query(pain,\infty)$
$O_6(2)$: $query(nonCyc,\infty)$
$P_7(1)$: $pain$ since $(pain,1)$
$P_8(1)$: $nonCyc$ since $(nonCyc,1)$

$O_4 \ldots :$


$O_4(2)$: $query(pain \land nonCyc \rightarrow mam,\infty)$
$P_9(2)$: mam since $(pain,1),(nonCyc,1),(pain \land nonCyc \rightarrow mam,3)$
$O_{10}(2)$: $propose(preg \rightarrow \neg mam,3)$
$P_{11}(2)$: $query(preg \rightarrow \neg mam,3)$
$P_{12}(2)$: $query(preg,3)$
$O_{14}(2)$: $\neg mam$ since $(preg,2),(preg \rightarrow \neg mam,1)$
$O_{13}(1)$: $preg$ since $(preg,2)$

5.11 Converting dialogues to and from Black and Hunter's system for inquiry dialogues

This section discusses the relation of dialogues in the adapted system to dialogues in the original system defined by Black and Hunter. It shows that it is possible to convert dialogues from and to the adapted system. It also shows that it is not possible to formulate dialogues in the adapted system that cannot be formulated in the original system.

5.11.1 Converting from Black and Hunter's system

Converting a dialogue in the adapted system to Black and Hunter's system and vice versa cannot be done on a move per move basis, since one move in Black and Hunter's system can be expanded to multiple moves in the adapted system, and the order of moves may be significantly different after translation. Some dialogues in Black and Hunter's system are not allowed in the adapted system, due to protocol rule R_8. These dialogues can be converted to the adapted system, but they are slightly altered to adhere to rule R_8.

First, a function is determined to allow a rule to be evaluated as an argument, for example for the purposes of determining a target for a move with a rule. For this purpose, *ruleToArg* turns a rule into an argument, for example: $ruleToArg(a \rightarrow b) = \langle \{(a, 1), (a \rightarrow b, 1)\}, b\rangle$. This is useful to determine the consequence of finding support for a rule. It is defined as:

Definition 58. Given a rule $\left[\bigwedge_{\phi \in \Phi} \phi\right] \rightarrow \psi$ and optionally a *pLevel* p (which defaults to 1), the function **ruleToArg** is defined as:

$$ruleToArg(\left[\bigwedge_{\phi \in \Phi} \phi\right] \rightarrow \psi, p) = A \text{ such that}$$

$$\left\| \begin{array}{l} prem(A) = \bigcup_{\phi \in \Phi} (\phi, p) \cup \left\{ \left[\bigwedge_{\phi \in \Phi} \phi\right] \rightarrow \psi, p \right\} \\ conc(A) = \psi \end{array} \right.$$

Next, an algorithm is required to insert moves into a dialogue at any position; this is required to allow moves to be repeated after the conversion. Basically, the algorithm inserts a new move into d, after m_j, with the speech act and player provided. Moves that have an *id* higher than j have their *id* increased, and moves that target one of the increased moves have their target increased as well.

Definition 59. Given a target move m_j, a dialogue d, a speech act for the new move and a player, the algorithm **insertAfterTarget** is defined as:

$$insertAfterTarget(m_j, d, speechact, x) =$$

$$\left[\begin{array}{l} \text{if } \nexists m_i \text{ such that } t(m_i) = m_j \text{ and } s(m_i) = speechact \\ \text{then insert } m_i \text{ into } d \text{ such that} \\ \forall m_k \in d \text{ where } id(m_k) > id(m_j), id(m_k) := id(m_k) + 1 \text{ and} \\ \forall m_l \in d \text{ where } t(m_l) > id(m_j), t(m_l) := t(m_l) + 1 \text{ and} \\ \quad \left\| \begin{array}{ll} id(m_i) & = id(m_j) + 1 \text{ and} \\ s(m_i) & = speechact \\ pl(m_i) & = P(x) \text{ if } Role(pl(t(m_u))) = O, \text{ otherwise } O \\ t(m_i) & = j \end{array} \right. \end{array} \right.$$

Additionally, two functions are required to provide respectively one (deterministically selected) or all targets of a speech act, in order to see which moves need to be repeated.

Definition 60. Given an speech act *speechact* and a dialogue d, the function **determineTarget**, of type $\mathit{determineTarget} : (L_c \times M^{\leq\infty}) \mapsto \mathbb{N}_1$, and **determineTargets**, of type $\mathit{determineTargets} : (L_c \times M^{\leq\infty}) \mapsto \mathcal{P}(\mathbb{N}_1)$, determine respectively the first and all the moves in the dialogue that have a speech act that can be attacked by the given speech act.

$$\mathit{determineTarget}(\mathit{speechact}, d) = \\ \begin{cases} x & \text{iff } x \in \mathit{determineTargets}(\mathit{speechact}, d) \text{ and} \\ & \forall y \in \mathit{determineTargets}(\mathit{speechact}, d), x \leq y \end{cases}$$

$$\mathit{determineTargets}(\mathit{speechact}, d) = \\ \left\{ x \;\middle|\; \begin{array}{l} m_x \in d \text{ and} \\ (\mathit{speechact}, s(m_x)) \in R_a \end{array} \right\}$$

If a speech act contains an unknown *pLevel* (i.e. it is a *query* or *propose* move, which have the *pLevel* of their target) the *pLevel* in the speech act is determined by R_a and can differ per target in the set provided by *determineTargets*.

With the functions defined above, the function to convert a dialogue from Black and Hunter's system for inquiry dialogues to the adapted system can be defined.

Definition 61. Given a well-formed dialogue D_1^t consisting of moves $m_1, \ldots, m_t$, the algorithm **dialogueToFramework** of type $\mathit{dialogueToFramework} : \mathcal{D} \mapsto \mathcal{P}(M)$ is defined as

$dialogueToFramework(D_1^t) = d$ such that $d \in \Pi$ and d is created by the following algorithm:

```
For every move $m_i \in D_1^t$:
  || $i$  is the index in the given dialogue
  ||      from $i = 1$ to $i = t$ and
  || $j$  is the index to append new moves to the result dialogue
  ||      starts at $j = 1$
  | if $m_i = \langle x, close, dialogue(\theta, \phi)\rangle$
  | do not add anything to $d$
  |
  | else if $m_i = \langle x, open, dialogue(wi, \phi)\rangle$
  | add $m_1, m_2$ to $d$ such that
  |     || $id(m_1)$  $= 1$
  |     || $pl(m_1)$  $= P(x)$
  |     || $s(m_1)$   $= open(\phi, \infty)$
  |     || $t(m_1)$   $= 0$
  |     || $id(m_2)$  $= 2$
  |     || $pl(m_2)$  $= O(x)$
  |     || $s(m_2)$   $= query(\phi, \infty)$
  |     || $t(m_2)$   $= 1$
  | and set $j$ to 3
  |
  | else if $m_i = \left\langle x, open, dialogue\left(ai, \left[\bigwedge \phi\right] \rightarrow \psi\right)\right\rangle$
  | add $m_j, m_{j+1} \ldots, m_{j+v}$ to $d$ such that
  |     || $id(m_j)$                  $= j$
  |     || $s(m_j)$                   $= propose\left(\left[\bigwedge \phi\right] \rightarrow \psi, p\right)$
  |     ||                            (where $p$ is set indirectly
  |     ||                             by the $determineTarget$ function)
  |     || $t(m_j)$                   $= determineTarget(s(m_j), d)$
  |     || $pl(m_j)$                  $= P(x)$ if $Role(pl(t(m_j))) = O$, otherwise $O$
  |     || $m_{j+1}, \ldots, m_{j+v}$ are query moves according to protocol rule $R_6$
  | and set $j$ to $j + v + 1$
  |
  | else if $m_i = \langle x, assert, A\rangle$
  | add $m_j$ to $d$ such that
  |     || $id(m_j)$  $= j$
  |     || $s(m_j)$   $= assert(A)$
  |     || $t(m_j)$   $= determineTarget(s(m_j), d)$
  |     || $pl(m_j)$  $= P(x)$ if $Role(pl(t(m_j))) = O$, otherwise $O$
  | and set $j$ to $j + 1$
  |
  | and if $i = t$, do $repeatMoves(d)$
```

where
$repeatMoves(d) =$ for each $m_t \in d$:

```
    if $s(m_t) = assert(A)$
    then $repeatLiteralPremises(m_t, d)$
next
    if $|determineTargets(s(m_t), d)| > |\{m_i \in d | s(m_i) = s(m_t)\}|$
    then for every target $w \in determineTargets(s(m_t), d)$
    $insertAfterTarget(m_w, d, s(m_t), Player(pl(m_t)))$
```

and
$repeatLiteralPremises(m_t, d) =$

```
if $s(m_t) = assert(A)$
then $\forall(z, p) \in prem(A)$ where $z$ is a literal
    if $\nexists m_u \in d$
    such that $s(m_u) = assert(Z)$
    and $prem(Z) = \{(z, p)\}$
    then for every target
    $w \in determineTargets(assert(z \text{ since } (z, p)), d)$
    $insertAfterTarget(m_w, d,$
        $assert(z \text{ since } (z, p)), Player(pl(m_t)))$
```

To summarise:

- every *close* move is ignored
- every move that opens a warrant inquiry dialogue is translated to an *open* move;
- every move that opens an argument inquiry dialogue is translated to one *propose* move with corresponding *query* moves;
- every move that asserts an argument is translated to one *assert* move;
- every translated move is repeated at every location it is allowed to, according to the protocol *and* for arguments, their literal premise(s) are moved as separate arguments, also at every location it is allowed to according to the protocol.

The following Lemmas are used to construct the proof for Proposition 1.

For the following Lemmas, assume D_1^t is a well-formed dialogue in Black and Hunter's inquiry dialogue system and $dialogueToFramework(D_1^t) = d$.

Lemma 1. Every type of move in Black and Hunter's system is translated or dropped: Definition 61 contains cases for the *open* moves, *close* moves and *assert* moves.

All the translated moves are translated in accordance with the protocol.

Lemma 2. In Definition 61, the *open* moves are translated to *open* or *propose* moves, and all the required *query* moves are added, in accordance with the protocol. Similarly, *assert* moves are translated to *assert* moves. Every resulting move is given a target and a role.

Next, in order to show that the dialogue is well-formed, the three conditions for well-formed dialogues must be met. From the above Lemmas and the fact that D_1^t is well-formed we can conclude that the resulting dialogue is correct according to the protocol.

Lemma 3. The resulting dialogue d is correct according to the protocol of the adapted system.

Proof Since D_1^t is well-formed, the structure of D_1^t ensures that there is always a move that can be targeted in the resulting dialogue. Combined with Lemma 2, the protocol is satisfied. $\square$

Next, the second condition is shown; every possible *query* move must have been moved.

Lemma 4. Every *open* and *propose* will be followed by all the required *query* moves.

Proof From Lemma 2. $\square$

Finally, moves must be repeated whenever necessary, according to Definition 53.

Lemma 5. In the resulting dialogue d every speech act that has been moved, has been moved against all its targets.

Proof Definition 61 contains a function $repeatMoves$ that ensures that every speech act present is moved against every possible target. This function is called on the resulting dialogue if $i = t$, so will be applied to the entire dialogue. $\square$

Proposition 1. If D_1^t is a well-formed dialogue in Black and Hunter's inquiry dialogue system and $dialogueToFramework(D_1^t) = d$, then d is a well-formed dialogue in the adapted system.

Proof From Lemma 1 through 5 we can conclude that if $dialogueToFramework(D_1^t) = d$ and D_1^t is a well-formed dialogue, then d is a well-formed dialogue. $\square$

5.11.2 Converting to Black and Hunter's system

The conversion from a dialogue in the adapted system to a dialogue in Black and Hunter's system is significantly more simple. First, a function is defined to remove all duplicate subtrees from a dialogue.

Definition 62. Given a well-formed dialogue d, the function **removeDuplicateSubtrees** of type $\mathcal{P}(M) \mapsto \mathcal{P}(M)$ is defined as:

```
$removeDuplicateSubtrees(d) =$
        while
            $\exists m_u, m_v$ such that
            $duplicateSubtree(m_u, m_v, d)$ and
                $\nexists m_w, m_z$ such that $duplicateSubtree(m_w, m_z, d)$
                and $|determineSubmoves(m_u, d)|$
                        $< |determineSubmoves(m_w, d)|$
        do
            if $id(m_u) < id(m_v)$
            then remove $m_v$ and every
            $m_x \in determineSubmoves(m_v, d)$ from $d$
            otherwise remove $m_u$ and every
            $m_x \in determineSubmoves(m_u, d)$ from $d$
```

where

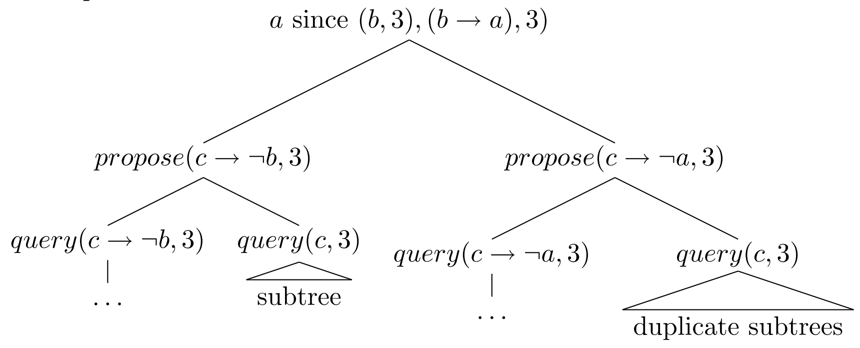
```
$duplicateSubtree(m_u, m_v, d) =$
        return $true$ iff $m_u \neq m_v$ and $s(m_v) = s(m_u)$
        and
            $\forall m_x \in determineSubmoves(m_u, d)$
            $\exists m_y \in determineSubmoves(m_v, d))$
            such that $s(m_x) = s(m_y)$
        and
            $\forall m_y \in determineSubmoves(m_v, d)$
            $\exists m_x \in determineSubmoves(m_u, d))$
            such that $s(m_x) = s(m_y)$
```

and

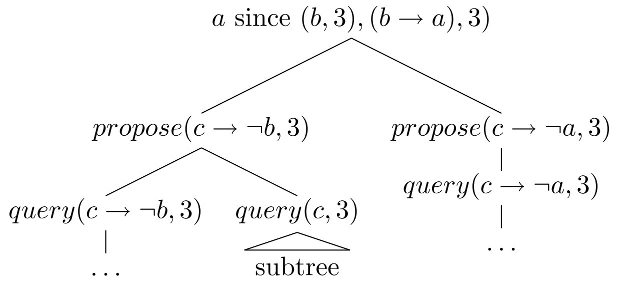
$$determineSubmoves(m_v, d) = \left\{ m_x \;\middle|\; \begin{array}{l} m_x \in d \\ \text{and } t(m_x) = m_v \end{array} \right\} \cup \left\{ m_y \;\middle|\; \begin{array}{l} m_y \in determineSubmoves(m_x, d) \\ \text{where } m_x \in d \\ \text{and } t(m_x) = m_v \end{array} \right\}$$

An example tree is shown below. Note that a dialogue modified in this way is not well-formed in the adapted system, however in Black and Hunter's system it will be, as the method of determining the status of the topic of the dialogue is different in their system.

For example:



is reduced to:

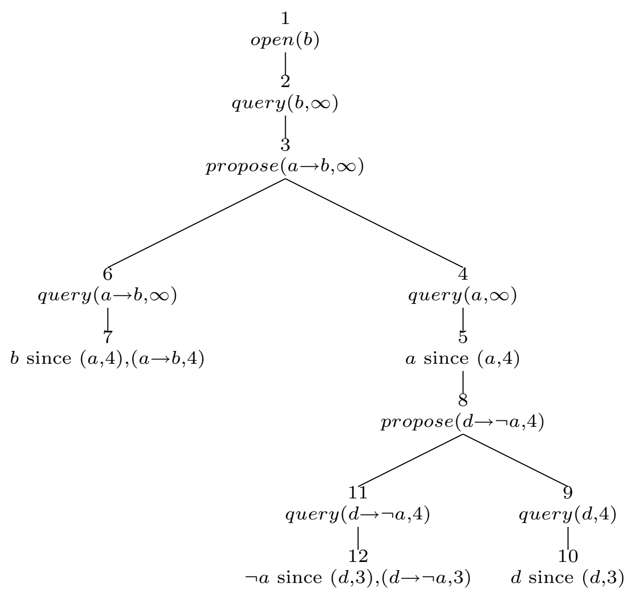


a since $(b,3),(b \to a),3)$
$propose(c \to \neg b, 3)$
$propose(c \to \neg a, 3)$
$query(c \to \neg a, 3)$
$query(c \to \neg b, 3)$
$query(c, 3)$
...
...
subtree

Now, the function to convert dialogues from the adapted system to Black and Hunter's system can be defined. The function does not follow the ids of the moves for translation, rather it traverses the dialogue tree in order to get the correct order of moves for Black and Hunter's system, which has more restrictions on move ordering. The tree traversal is a modified preorder traversal, where *assert* moves (or *query* followed by *assert*) are considered to be on the left and *propose* (or *query* followed by *propose*) on the right[26].

The path that the algorithm takes through the tree is displayed in the tree below. The numbers at the nodes represent the order in which the algorithm moves through the tree.

[26]For convenience, the distinction between a move and the node labelled with a move are ignored in the definition, read 'the node labelled with m' when no functions are applied to a move m.


1
$open(b)$
2
$query(b,\infty)$
3
$propose(a\rightarrow b,\infty)$
6
$query(a\rightarrow b,\infty)$
7
b since $(a,4),(a\rightarrow b,4)$
4
$query(a,\infty)$
5
a since $(a,4)$
8
$propose(d\rightarrow\neg a,4)$
11
$query(d\rightarrow\neg a,4)$
12
$\neg a$ since $(d,3),(d\rightarrow\neg a,3)$
9
$query(d,4)$
10
d since $(d,3)$

Step 7 and 12 add *close* moves to the dialogue, for their respective argument inquiry dialogues.

Definition 63. Given a well-formed dialogue d_t consisting of moves $m_1, \ldots, m_t$, the function **dialogueToInquiry** of type $dialogueToInquiry : \mathcal{P}(M) \mapsto \mathcal{D}$ is defined using two stacks *Guide* and *CurrentDialogue* of nodes to visit and *push*, *pop* and *top* to respectively add an element to the top of the stack, and view the top element of the stack respectively with or without removing it.

$dialogueToInquiry(d) = D_1^v$ such that

d	= the well-formed original dialogue
D_1^v	= the resulting dialogue
j	= index to add new moves at in D_1^v
$Guide$	= stack to guide the algorithm over the dialogue tree
$CurrentDialogue$	= stack to mark the current dialogue

```
$d_t = removeDuplicateSubtrees(d)$
$push(m_1, Guide)$
while $m_i = pop(Guide)$
    if $s(m_i) = open(\phi, \infty)$
        $push(dialogue(wi, \phi), CurrentDialogue)$ and
        $\forall m_v, m_w \in d$ if $t(m_w) = i$ and $t(m_v) = w$
        then $push(m_v, Guide)$; first all *propose* , then all *assert*
        and $m_j \in D_1^v = \langle Player(pl(m_i)), open, dialogue(wi, \phi)\rangle$
        and increment $j$
    else if $s(m_i) = propose\left(\left[\bigwedge \phi\right] \rightarrow \psi, p\right)$
        $push(dialogue(ai, \phi), CurrentDialogue)$
        and $m_j \in D_1^v = \langle Player(pl(m_i)), open, dialogue(ai, \left[\bigwedge \phi\right] \rightarrow \psi)\rangle$
        and increment $j$ and
            $\forall m_v, m_w \in d$ if $t(m_w) = i$ and $t(m_v) = w$
            and $s(m_w) = query(\chi, p)$(where $\chi$ is a literal)
            and $Performative(s(m_v)) = assert$
            then $m_j \in D_1^v = \langle Player(pl(m_v)), assert, Content(s(m_v))\rangle$
            and increment $j$ and
            $\forall m_z \in d$ if $t(m_z) = v$
            then $push(m_z, Guide)$; first all *propose* , then all *assert*
        next
            $\forall m_v, m_w \in d$ if $t(m_w) = i$ and $t(m_v) = w$
            and $s(m_w) = query\left(\left[\bigwedge \phi\right] \rightarrow \psi, p\right)$
            and $Performative(s(m_v)) = assert$ then $push(m_v, Guide)$
        next
            $\forall m_v, m_w \in d$ if $t(m_w) = i$ and $t(m_v) = w$
            and $Performative(s(m_v)) = propose$ then $push(m_v, Guide)$
    else if $s(m_i) = assert(A)$
        if $\nexists m_w \in d_t$ such that $s(m_w) = assert(B)$
        and $prem(A) \subset prem(B)$
        and $Player(pl(m_w)) = Player(pl(m_i))$
        then
            $\forall m_v \in d$ if $t(m_v) = i$
            then $push(m_v, Guide)$; first all *propose* , then all *assert*
            and $m_j \in D_1^v = \langle Player(pl(m_i)), assert, A\rangle$
            and increment $j$
            and if $s(t(m_i)) = query\left(\left[\bigwedge \phi\right] \rightarrow \psi, p\right)$
            and $m_u = top(Guide)$ and $t(m_u) \neq t(m_i)$
            then $dialogue = pop(CurrentDialogue)$
            and $\forall x \in \mathcal{I}$, add $\langle x, close, dialogue\rangle$ to $D_1^v$
Finally, if $CurrentDialogue$ is not empty
then $\forall x \in \mathcal{I}$, add $\langle x, close, top(CurrentDialogue)\rangle$
and $\forall m_j, m_k$ where $id(m_j) = id(m_k) + 1$
if $Player(pl(m_j)) = Player(pl(m_k))$ then insert
$\langle x, close, dialogue(Type(Current(D_1^j)), Topic(Current(D_1^j)))\rangle$
between $m_j$ and $m_k$ such that $x \neq Player(pl(m_k))$
```

To summarise:

- all duplicate subtrees are removed from the dialogue;
- *open* moves are changed to moves that open a warrant inquiry dialogue;
- *propose* moves are translated to moves that open an argument inquiry dialogue;
- *assert* moves are translated to assert moves;
- *close* moves are added when an argument inquiry is done;
- every *query* move is ignored;
- *close* moves for the warrant inquiry dialogue are added;
- *close* moves are moved inbetween two consecutive moves if they are made by the same player;
- the restriction on order of moves has been met by traversing the tree with a specific pattern.

The following Lemmas are used to construct the proof for Proposition 2.

For the following Lemmas, assume d is a well-formed dialogue in the adapted system and $dialogueToInquiry(d) = D_1^v$.

Lemma 6. Every type of move in the adapted system is translated or dropped: Definition 63 contains cases for the *open* moves, *close* moves and *assert* moves.

All the translated moves are translated in accordance with the protocol.

Lemma 7. The content of the moves in D_1^v is correct according to Black and Hunter's protocol.

Proof In Definition 63, every translated move is translated to a sequence of moves that is correct according to the protocol defined by Black and Hunter. By using a *Guide* to move over the tree, the order of the moves is ensured to be correct in D_1^v. Finally, d is well-formed and duplicate moves have been removed by the algorithm defined in Definition 62. □

Next, in order to show that the dialogue is well-formed, the three conditions for well-formed dialogues must be met.

Lemma 8. The first move in D_1^v move be an *open* move.

Proof Since d is well-formed, the first move is either an *open* or *propose* move. In Definition 63, *open* and *propose* moves are translated to a *open* move. $\square$

Next, the second condition is shown: termination.

Lemma 9. D_1^v is a terminated dialogue.

Proof In Definition 63, *close* moves are added for each subdialogue, and at the end of the translation, *close* moves are added for the top-dialogue. This is done by maintaining a stack that records the current dialogue. $\square$

Finally, subdialogues must be well-formed argument inquiry dialogues.

Lemma 10. Each subdialogue in D_1^v is a well-formed argument inquiry dialogue.

Proof From Lemma 9 and Lemma 7. $\square$

Proposition 2. If d is a well-formed dialogue in the adapted system and $dialogueToInquiry(d) = D_1^v$, then D_1^v is a well-formed dialogue in Black and Hunter's inquiry dialogue system.

Proof From Lemma 6 through 10 we can conclude that if $dialogueToInquiry(d)$ $= D_1^v$ and d is a well-formed dialogue, then D_1^v is a well-formed dialogue. $\square$

5.11.3 No 1:1 relation

Since there is such a direct correlation between moves in Black and Hunter's system and this adapted system, any dialogue formulated in Black and Hunter's system has an equivalent dialogue in the adapted system. However, since there are restrictions on the *assert* moves, the set of dialogues that can be formulated in Black and Hunter's system is larger than the set of dialogues that can be formulated in the adapted system. Since the adapted system can formulate these dialogues, but forces assert moves inside argument inquiry dialogues, several dialogues in Black and Hunter's system convert to the same dialogue in the adapted system.

Aside from the fact that the ordering of many of the moves could be different, every *propose* move $m_i \in d_t$ that attacks a move $m_j \in d_t$ creates two possible situations: either the *assert* move containing the entire argument is moved directly against m_j (i.e. if all the premises of this argument are in one agent's belief base) or the premises are move individually before the entire argument is moved, in an argument inquiry dialogue. Note that this situation was also the reason R_8 was added to the protocol; if it were not, both options would be represented in the tree.

5.11.4 Proof of equivalence

In order to prove that a dialogue in one system is equivalent to a dialogue it has been converted to in another system, a measure for equivalence has to be determined. First, this measure is defined. After this, the statement that is to be proved will be formalised. Third, the proof is split in smaller lemmas and worked out.

In order for two dialogues to be equivalent, the following must hold:

Outcome The dialogues must have the same outcome;

Committed facts The dialogues must have the same facts and rules in the commitment stores, though not necessarily in the same commitment stores (because of the difference between players and roles in the adapted system);

Arguments The dialectical trees must be equal to show that the same arguments have been used.

Definition 64. Let d, d' be two dialogues with the same topic. Dialogue d and d' are **equivalent** if they meet three conditions:

Outcome $Outcome(d) = Outcome(d')$;

Committed facts $C_d = C_{d'}$;

Arguments $dialecticalTree(d) = dialecticalTree(d')$.

Note that these criteria are not restricted to well-formed dialogues; dialogues that are not well-formed are still equivalent if these conditions are met. However, since the conversion produces well-formed dialogues from well-formed dialogues, the proof below is restricted to well-formed dialogues.

In order to prove that the conversion provides an equivalent dialogue, but in a different system, a proof is constructed to show that a dialogue before the conversion and the dialogue after the conversion are equivalent according to the above criteria. This is proven in reverse order of the conditions, by first proving that the same arguments are used, second that the same committed facts are used, and finally that the outcome is identical.

The following Lemmas are used to construct the proof for Proposition 3.

Arguments In order to prove that the arguments asserted in the dialogues are identical, the dialectical trees of the dialogues must be shown to be equal. While the conversion from Black and Hunter's system to the adapted system does create more arguments, these are contained within the arguments that are already present in the original dialogue, so they are not new arguments and are not visible in the adapted system's dialectical tree.

Lemma 11. The function in Definition 61 does not add arguments that are visible in the adapted system's dialectical tree, and does not remove arguments.

Conversion to Black and Hunter's system only removes arguments that do not need to be asserted in Black and Hunter's system, they already appear in the dialectical tree.

Lemma 12. The function in Definition 63 does not remove arguments that are visible in the dialectical tree[27] and does not add arguments.

A function has been provided to create the dialectical tree from a dialogue in the adapted system. Following these steps removes all the additional moves from the dialogue tree until only the complete asserted arguments remain. Since Lemma 11 and Lemma 12 show that no arguments are added or removed and the dialogue tree contains all moves and repeated moves where necessary during conversion to compensate for the fact that the tree in Black and Hunter may contain arguments at multiple locations, this must necessarily be the same tree.

Lemma 13. When a dialogue is translated to or from the adapted system, the dialectical tree in the original dialogue is equal to the dialectical tree in the resulting dialogue.

Proof From Lemma 11, Lemma 12 and Definition 57. □

Committed facts In order to prove that the same committed facts have been used, the commitment functions are examined. Both Black and Hunter's system and the adapted system only make changes to the commitment stores when an assert move is made. Lemma 11 and Lemma 12 have shown that the same arguments are asserted.

We can conclude that there cannot be more committed facts than there are in the original dialogue, but we can also conclude that there cannot be less, by looking at the commitment functions. Since these function are nearly identical (except for the difference of players and roles, which is ignored as it has no consequence for the content of the combined commitment stores), it is safe to

[27]It may remove arguments that are also present as a subargument of another argument, these duplicates are not necessary in Black and Hunter's system.

say there are no difference in this area either.

Lemma 14. When a dialogue is translated to or from the adapted system, the same facts are committed in the original and the resulting dialogue.

Proof From Lemma 11, Lemma 12 and the commitment function in Definition 50, which only updates for *assert* moves and is structurally identical to the commitment function in 20. □

Outcome It has already been shown that the dialectical trees are equal. Black and Hunter base the outcome of a dialogue on the status of nodes in the dialectical tree. While the adapted system does not, the outcome is designed to give a similar result: a set of arguments when the dialogue is an argument inquiry dialogue, or a fact that is deemed warranted if the dialogue is a warrant inquiry dialogue.

Since the validity of the topic of the dialogue is based on the moves that attack it, the attack relation that defines the dialogue tree is also visible in the dialectical tree (since, in the adapted system, dialectical tree is created from the dialogue tree).

Lemma 15. In Definition 57 the attack relations outside argument inquiry dialogues are maintained and attack relations inside the argument inquiry dialogue exist merely to construct the argument that we see in the dialectical tree.

Additionally, the last relevant move, which determines the outcome of the dialogue in the adapted system, is also present in the dialectical tree if the dialogue is well-formed.

Lemma 16. If a dialogue is well-formed, the last relevant move is an *assert* move and the argument inside this assert move is present in the dialectical tree (possibly inside a larger argument).

Thus, the last relevant move might also be determined using the dialectical tree before and after the move, which resembles the way Black and Hunter determine the outcome.

Lemma 17. When a dialogue is translated to or from the adapted system, the original and the resulting dialogue have the same outcome.

Proof From Lemma 15 and 16 we know that that determining the outcome on the dialectical tree gives the same result as determining outcome on the last relevant move, and since the dialectical trees are the same according to Lemma

13, the outcome must be the same. □

Proposition 3. If D^t_1 is a well-formed dialogue in Black and Hunter's inquiry dialogue system and $dialogueToFramework(D^t_1) = d_v$, then d_v and D^t_1 are equivalent according to Definition 64. Similarly, if d is a well-formed dialogue in the adapted system and $dialogueToInquiry(d) = D^u_1$, then D^u_1 and d are equivalent according to Definition 64.

Proof Proposition 1 and 2 have shown that translation of a well-formed dialogue produces a well-formed dialogue. From Lemma 13, 14 and 17, it has been proven that dialogues before and after the conversion process (in either direction) are equivalent according to the criteria defined in Definition 64. □

5.12 Generating dialogues

Aside from a system for inquiry dialogues, Black and Hunter have also provided an agent strategy for use in generating dialogues. They have proven that the dialogues generated by agents using this exhaustive strategy are always sound and complete, based on the agents' belief bases. This strategy can be used with minor changes in the adapted system. Three changes are required: since *close* moves are not possible, these are removed from the strategy; *query* moves must be added using protocol rule R_6 and the format of the moves must be changed.

Definition 65. The **exhaustive strategy** $\Omega_x(d)$, for player x participating in a well-formed dialogue d is a function $\Omega_x : \mathcal{P}(M) \mapsto M$ and is defined as:

$$\Omega_x(d) = \begin{cases} Pick_q(Queries_x(d)) & \text{iff } Queries_x(d) \neq \emptyset \\ Pick_a(Asserts_x(d)) & \text{iff } Queries_x(d) = \emptyset \\ & \quad \text{and } Asserts_x(d) \neq \emptyset \\ Pick_o(Proposes_x(d)) & \text{iff } Queries_x(d) = \emptyset \\ & \quad \text{and } Asserts_x(d) = \emptyset \\ & \quad \text{and } Proposes_x(d) \neq \emptyset \end{cases}$$

where

$$
\left\|
\begin{array}{lll}
Pick_a(\Xi) & = & \text{as defined in Definition 26, with altered move notation;} \\
Pick_o(\Xi) & = & \text{as defined in Definition 26, with altered move notation;} \\
Pick_q(\Xi) & = & \left\{
\begin{array}{l}
m_i \quad \text{iff } m_i \in \Xi \\
\qquad \text{and } s(m_i) = query\left(\left[\bigwedge \phi\right] \to \psi, p\right) \\
\qquad \text{or} \\
\qquad \quad \left[
\begin{array}{l}
s(m_i) = query(\alpha) \\
\text{and } \nexists m_j \in \Xi \text{ such that} \\
\quad \left[
\begin{array}{l}
s(m_j) = query\left(\left[\bigwedge \phi\right] \to \psi, p\right) \\
\text{or } s(m_j) = query(\beta) \\
\ \text{and } \beta < \alpha \\
\ \text{when lexicographically sorted}
\end{array}
\right.
\end{array}
\right.
\end{array}
\right.
\end{array}
\right.
$$

and

$Asserts_x(d) =$

$$
\left\{ m \;\middle|\;
\begin{array}{l}
m \in Pr(d) \text{ and} \\
s(m) = assert(A) \text{ and} \\
\quad \left[
\begin{array}{l}
pl(m) \in Roles \text{ and} \\
A \in \mathcal{A}(\Sigma^x \cup C_d(P) \cup C_d(O))
\end{array}
\right.
\end{array}
\right\}
$$

and

$Proposes_x(d) =$

$$
\left\{ m \;\middle|\;
\begin{array}{l}
m \in Pr(d) \text{ and} \\
s(m) = propose(R) \text{ and} \\
\exists p \in \mathbb{N}_1 \text{ such that } (R, p) \in \Sigma^x
\end{array}
\right\}
$$

and

$Queries_x(d) =$

$$
\left\{ m \;\middle|\;
\begin{array}{l}
m \in Pr(d) \text{ and} \\
s(m) = query(\phi) \\
\text{where } \phi \in \mathcal{S} \cup \mathcal{R}
\end{array}
\right\}
$$

Note that, as in Black and Hunter's exhaustive strategy, the strategy is assumed to be used after the first open move in a dialogue has been made (in the adapted system, this could be either the first *open* or *propose* move).

5.12.1 Proof of equivalence

The following Lemmas are used to construct the proof for Proposition 4.

In order to show that the adapted exhaustive strategy produces the same dialogues as the original exhaustive strategy, the dialogue equivalence conditions from Definition 64 is used.

Lemma 18. The $Pick_a$ and $Pick_o$ functions are identical to Black and Hunter's definition, except for notation. The $Asserts_x$ function has been rewritten for the adapted system to reflect the changes in role notation, but is otherwise unchanged.

Arguments The only differences in the generated dialogues are the result of differences in the protocol, as the protocol of the adapted system requires that literals are moved individually before they are moved inside larger arguments. These moves are added to the *assert* moves generated by $Asserts_x$, but since these were moved inside larger arguments in Black and Hunter's exhaustive strategy, the arguments do not differ.

Lemma 19. $Asserts_x$ provides every argument the original in Black and Hunter's system does, and does not add arguments that are visible in the dialectical tree.

Committed facts On the basis of the equivalence condition of arguments, this must be the same.

Lemma 20. Given the same belief bases, the committed facts in a dialogue generated by the adapted exhaustive strategy and those in a dialogue generated by Black and Hunter's exhaustive strategy are identical.

Proof From Lemma 19 and the commitment functions in Definition 50 and 20. □

Outcome Black and Hunter prove soundness and completeness on the outcome of the dialogue when the exhaustive strategy is used. This must also be possible with the adapted exhaustive strategy. Fortunately, this can also be based on the fact that the set of arguments is roughly equivalent and the fact that the $Proposes_x$ is equivalent to $Opens_x$. This ensures that a dialogue is created with the same structure as the one that would be created by the original exhaustive strategy.

The possible differences are based on the protocol and difference in dialogue structure between the two systems. As it has been shown that the structures can be converted to each other, and the original dialogue is equivalent to the converted dialogue, the dialogue structure has no influence on the outcome.

Lemma 21. The outcome of a dialogue generated by the adapted exhaustive strategy is equivalent to the outcome of a dialogue generated by Black and Hunter's exhaustive strategy, given the same belief bases.

Proof From Lemma 18, 19 and 17. □

Proposition 4. $\forall d$ that is a dialogue produced by the exhaustive strategy defined in Definition 65, $\exists D_1^t$ such that D_1^t is generated by Black and Hunter's exhaustive strategy and $dialogueToInquiry(d) = D_1^t$ and $\forall D_1^v$ that is a dialogue produced by Black and Hunter's exhaustive strategy, $\exists d'$ such that d' is generated by the exhaustive strategy from Definition 65 and $dialogueToFramework(D_1^v) = d'$.

Proof Lemma 18 through 21 show that the conditions in Definition 64 are met by dialogues generated by Black and Hunter's exhaustive strategy and the strategy defined in Definition 65, so that given the same belief bases, equivalent dialogues are generated. $\square$

5.13 Fundamental properties

The adapted system can now be compared to the fundamental properties of Black and Hunter's system for inquiry dialogues. These fundamental properties were determined on page 39. First, the distinction between agents and roles has been made clear in the adapted system, so this property has been preserved. Similarly, since the exhaustive strategy has been converted successfully, the predetermined result has also been preserved and soundness and completeness can be benchmarked on the belief bases if this strategy is used.

The major differences are in the simplicity of the moves and the absence of targets for moves. Every move now has a target; this was a necessary change for the system to fit with Prakken's framework, it could not be avoided without changing the framework. The simplicity of moves has been reduced, increasing from three performatives to four and, more importantly, some of these are in forced relationship, i.e. the *query* moves following a *propose* or *open* move.

Lastly, the cooperative nature of the dialogue has been preserved when viewed from an agent's perspective, but not when viewed from a role. There are specific Proponent and Opponent roles, so the competitive element present in persuasion dialogues has been incorporated into the adapted system. From an agent's perspective, however, these are merely tools which are used when they are convenient. They make explicit whether a move aims to prove or disprove the topic of the dialogue, a property that was implicit in Black and Hunter's system.

5.13.1 Removing *propose*

Given the fact that the simplicity of moves has been reduced, one might wonder why the *propose* move is necessary, given the similarity with the *assert* move. Why not just assert the rule, instead of using a special move?

The most important reason this was not done for the adapted system is the fact that the *propose* move and *assert* move represent significantly different dialogue actions: the *propose* move proposes a dialogue about the possibility of a rule being used for the dialogue, while an assert move states that the agent believes something to be true.

A few example trees are shown where the *propose* was replaced, all alterations of the tree below, and each will be discussed.

$$\begin{tikzcd}
 & open(a) \arrow[d, dash] & \\
 & query(a, \infty) \arrow[d, dash] & \\
 & propose(b \rightarrow a, \infty) \arrow[dl, dash] \arrow[dr, dash] & \\
query(b \rightarrow a, \infty) \arrow[d, dash] & & query(b, \infty) \arrow[d, dash] \\
a \text{ since } (b, 2), (b \rightarrow a, 2) & & b \text{ since } (b, 2)
\end{tikzcd}$$

Assert the entire argument instead of *propose* :

$$\begin{tikzcd}
 & open(a) \arrow[d, dash] & \\
 & query(a, \infty) \arrow[d, dash] & \\
 & a \text{ since } (b, 2), (b \rightarrow a, 2) \arrow[dl, dash] \arrow[dr, dash] & \\
query(b \rightarrow a, \infty) \arrow[d, dash] & & query(b, \infty) \arrow[d, dash] \\
b \rightarrow a \text{ since } (b \rightarrow a, 2) & & b \text{ since } (b, 2)
\end{tikzcd}$$

This first example is problematic for several reasons: First, assuming agents only move things they believe, the a since $(b, 2), (b \rightarrow a, 2)$ can only be moved if both premises are in the beliefs of a single agent (or the commitments of the players, which in this case don't contain the necessary beliefs, as the argument inquiry is opened to determine these). Second, it would mean that the entire argument can be moved, and the tree below it will be fixed and not allow any alterations; if it could contain different premises than those that were used in the replaced *propose* move, the extending of arguments would have to be allowed. It has been shown before that this is inconsistent, given the preference levels.

Assert the rule instead of *propose* :

$$\begin{tikzcd}
 & \mathit{open}(a) \arrow[d, dash] & \\
 & \mathit{query}(a, \infty) \arrow[d, dash] & \\
 & b \rightarrow a \text{ since } (b \rightarrow a, 2) \arrow[dl, dash] \arrow[dr, dash] & \\
\mathit{query}(b \rightarrow a, \infty) \arrow[d, dash] & & \mathit{query}(b, \infty) \arrow[d, dash] \\
a \text{ since } (b, 2), (b \rightarrow a, 2) & & b \text{ since } (b, 2)
\end{tikzcd}$$

This second tree is more reasonable; it is not problematic for the commitment store function, as the agent basically states that he beliefs that this defeasible rule is true, nor does it require one agent to believe everything in the argument. They replying *query* move is somewhat odd, but a *query*(a) is not an option either, as this has been moved higher up in the tree. Unfortunately, it is also not clear how the ∞ weight of the *query* move above the replaced *propose* move is reflected in the *query* moves below the replaced *propose* move, nor is it clear how the moves attack each other.

Assert the rule instead of *propose* move, and only the premises queried:

$$\begin{tikzcd}
\mathit{open}(a) \arrow[d, dash] \\
\mathit{query}(a, \infty) \arrow[d, dash] \\
b \rightarrow a \text{ since } (b \rightarrow a, 2) \arrow[d, dash] \\
\mathit{query}(b, \infty) \arrow[d, dash] \\
b \text{ since } (b, 2)
\end{tikzcd}$$

This last example is perhaps the most similar to Prakken's dialogues; the entire argument is never moved, but it can be read from the moves in the tree. In this case, the *query* move has lost its special function and will function as the *why* move does in Prakken's example liberal dialogues. Like the first example, this proves to be an inconsistent system, as extension of an argument with a certain preference level may actually make it weaker, breaking the defeat relations.

To conclude, the option to replace the *propose* move with the *assert* move might make the moves more simple, but this comes at the cost of the concept of argument inquiry dialogues and it might have a significant impact on either the structure and/or the consistency of the dialogues.

5.14 Summary

This chapter has shown that it is possible to adapt Black and Hunter's system for inquiry dialogues to fit in Prakken's framework. The cost of the change was not significant, but there are considerable differences between the original system as defined by Black and Hunter, and the adapted version.

First, there are now more speech acts, with different, more complex conditions. Second, moves can now be moved several times, something that was not a possibility in Black and Hunter's system. Third, arguments must be built up using argument inquiry dialogues, in order to ensure a readable tree. Fourth, since close moves are no longer available to skip turns, turntaking restrictions have been loosened to allow any player to make a move in a given turn.

Prakken's framework remains unchanged except for some notational changes. This means that any statement made about Prakken's framework is applicable to this adapted system for inquiry dialogues.

Functions have been provided to convert dialogues to and from the adapted system, and while it is not a 1 to 1 mapping, it is a restricted mapping: converting a dialogue from a set of dialogues in Black and Hunter's system (with the same dialectical tree) to the adapted system and back to Black and Hunter's system will result in a dialogue in that set.

The exhaustive strategy defined by Black and Hunter has also been defined for the adapted system, and the differences are small, reflecting the changes in the speech acts.

Finally, no mention has been made about restrictions on the number of players in $\mathcal{I}$, and the players have been effectively disconnected from the dialogue; they make the moves, but the fact that a move is made by $i \in \mathcal{I}$ has no consequence whatsoever, since Prakken's proofs use the roles, not the agents[28].

[28]Recall that his definition of agents was renamed roles.

6 Adapted version of Prakken's framework that is compatible with Black and Hunter's system for inquiry dialogues

The previous chapter examined the changes needed to Black and Hunter's system for inquiry dialogues, in order to fit its dialogues into Prakken's framework without changing anything that was formally defined in Prakken's framework definition. This chapter defines a system the other way around, adapting Prakken's framework where necessary to allow Black and Hunter's inquiry dialogues to be formulated in it without making significant changes to the structure of Black and Hunter's system. Functions of Prakken's that were not explicitly defined, such as the definition for the turntaking function will be filled in where necessary.

When the adapted system is defined, any statement made about Black and Hunter's system is applicable to it.

The necessary adaptions to Prakken's framework and the functions that have to be defined in Prakken's framework are given, including the protocol for the system. Next, two functions are defined to provide the outcome of the different dialogues. This is followed by some example dialogues, and functions to convert dialogues from and to Black and Hunter's system for inquiry dialogues. After that, the exhaustive strategy is defined for the adapted system. Finally, the fundamental properties are evaluated for the adapted system.

6.1 Framework definition

The basic framework is identical to the one from the previous chapter, in Definitions 39 through 41; the protocol, moves and speech acts are different. The modus ponens rule is used here as well.

The following sections will define the system more specifically. Note that comments made about notation on page 11 apply here as well. Additionally, as in the previous chapter, dialogues are assumed to be well-ordered sets of moves, allowing notation such as $m_t \in D_r^t$ and $m_t \in d_t$.

6.2 Argumentation theory

The set *Args* in Definition 40 is defined as $\mathcal{A}(\mathcal{S}^* \cup \mathcal{R}^*)$. The $\Rightarrow$ relation, which is the defeat relation between arguments, is defined from the definition of the defeat relation between arguments given by Black and Hunter. It is repeated below for convenience.

Definition 9, repeated[29]. Given arguments A, B and SB, with SB a sub-

[29]Originally defined on page 13

argument of B, where A attacks B at SB, if $pLevel(A) \leq pLevel(SB)$ then A **defeats** B, specifically if $pLevel(A) < pLevel(B)$ then A is a **proper defeater** of B, otherwise A is a **blocking defeater** of B and vice versa.

6.3 Communication language

Since performatives are not explicitly defined, but defined in the table of moves, a separate set is defined from this table of moves to match with the *Perf* set in Prakken's framework.

The framework defines a performative as unary. While it is true that the performatives in Black and Hunter's system do receive only a single parameter, this parameter is often of a complex form: $dialogue(type, topic)$ or $\langle \Phi, \phi \rangle$. Additionally, given the type of the dialogue, certain restrictions apply to the topic of the dialogue. The open and close moves require some modification to fit the style of the framework.

Definition 66. The **performatives** for the adapted system for inquiry dialogues are
$Perf =$

$$\begin{cases} assert & \text{for asserting arguments,} \\ open_{ai} & \text{to open an argument inquiry,} \\ close_{ai} & \text{to close an argument inquiry,} \\ open_{wi} & \text{to open a warrant inquiry, and} \\ close_{wi} & \text{to close a warrant inquiry} \end{cases}$$

Each is a unary function, with restrictions on the parameters to be defined in the communication language.

It is clear that, using the newly defined performatives, the communication language can be defined in accordance with Prakken's structure. An adapted set for the L_c is defined below, using Black and Hunter's notation for $\mathcal{S}^*$ and $\mathcal{R}^*$[30].

Definition 67. The **communication language** for the adapted system for inquiry dialogues is defined as the set
$L_c =$

$$\begin{array}{l} \{\ assert(c) \mid c \in Args\ \} \\ \cup \\ \{\ open_{ai}(c) \mid c \in \mathcal{R}^*\ \} \\ \cup \\ \{\ close_{ai}(c) \mid c \in \mathcal{R}^*\ \} \\ \cup \\ \{\ open_{wi}(c) \mid c \in \mathcal{S}^*\ \} \\ \cup \\ \{\ close_{wi}(c) \mid c \in \mathcal{S}^*\ \} \end{array}$$

[30] Which are equal to sets defined by Prakken, respectively L_t and R

Note that these changes are not an adaptation of Black and Hunter's system for inquiry dialogues, this is merely a notational alteration. For example the move $\langle x, open, dialogue(\theta, \gamma)\rangle$ would have the speech act $open_\theta(\gamma)$.

6.3.1 Attacking reply relation R_a and surrendering reply relation R_s

No equivalent of Prakken's R_a and R_s are defined in Black and Hunter's system. While the lack of R_s is not a significant loss, since Black and Hunter's system really only attacks using counterarguments, and has no *why* moves, it is necessary to define an equivalent of R_a.

Given the communication language defined in Definition 67 above, this relation can be established based on the only comparable relation, the attack relation between arguments. Definition 8 shows that arguments in Black and Hunter's system are in an attack relation if the attacker's claim conflicts with either the claim of the object of the attack, or one of the subarguments in the object of the attack. The winner is determined by preference level. Based on this, a definition for R_a is defined[31].

Definition 68. The **attack relation between speech acts** for the adapted system for inquiry dialogues is defined as

$$R_a = \left\{ (assert(c), assert(d)) \;\middle|\; \begin{array}{l} assert(c), assert(d) \in L_c, \\ c \text{ defeats } d \end{array} \right\}$$

Note that, since there are no performatives that can be surrendered to, the set R_s can be empty.

Definition 69. The **surrender relation between speech acts** for the adapted system for inquiry dialogues is defined as
$R_s = \emptyset$

6.3.2 Move targeting

Many of the moves in the newly defined L_c, defined in Definition 67, are not compatible with Prakken's idea of move targets. Therefore, the definition of move targets needs to be modified slightly, to allow for a correct definition of targets for moves. Moves have a set of targets; for some moves the set is $\emptyset$, some have only $\{0\}$, the rest have one or multiple targets.

Definition 70. Moves that require targets are called **operative moves**, the set of which is denoted by $\mathcal{O}(D_r^t)$ and defined as
$\mathcal{O}(D_r^t) =$

[31]Again, using Prakken's notation *Args*

$$\left\{ m_i \in D_r^t \;\middle|\; \begin{array}{l} \left[\begin{array}{l} s(m_i) = assert(A) \text{ and} \\ Type(Current(D_r^i)) = wi \end{array}\right. \\ \text{or} \\ \left[\begin{array}{l} s(m_i) = assert(A) \text{ and} \\ Type(Current(D_r^i)) = ai \\ \text{and } \nexists D_u^v \text{ such that} \\ \quad \left[\begin{array}{l} D_r^i \text{ sub-dialogue of } D_u^v \\ \text{and } Type(D_u^v) = ai \end{array}\right. \\ \text{and the consequent of} \\ Topic(Current(D_r^i)) = conc(A) \end{array}\right. \end{array} \right\}$$

Operative moves are assert moves that are inside a warrant inquiry dialogue, or inside the topmost argument inquiry dialogue, if they make a claim about the topic of that dialogue. Deeper argument inquiry dialogues do not provide any assertion that can attack earlier operative moves. With this definition, every argument inquiry dialogue inside a warrant inquiry dialogue provides one, multiple, or possibly no operative moves, but any operative move provided has a valid target.

For the operative moves, the targets are determined using the attack relation between the arguments asserted in the moves. The first operative move has target 0, as in Prakken's definition. Any move not in $\mathcal{O}$ has no target, so the $t(m)$ function returns *null*.

The definition below ensures that any move that can be attacked by an operative move is in the target set of this operative move. If there is an operative move for which there are no targets, the result is a singular set containing only 0.

Definition 71. The function $determineTarget(D_1^t)$, of type $determineTarget : \mathcal{D} \mapsto \mathcal{P}(\mathbb{N}_0)$, returns the set of targets of move m_t in dialogue D_1^t, where $D_1^t = m_1, \ldots, m_t$.

$$determineTarget(D_1^t) =$$
$$\left\{ id(m_i) \;\middle|\; \begin{array}{l} m_t, m_i \in \mathcal{O}(D_1^t) \text{ and} \\ (m_t, m_i) \in R_a \end{array} \right\}$$
$$\cup$$
$$\left\{ 0 \;\middle|\; \begin{array}{l} m_t \in \mathcal{O}(D_1^t) \text{ and} \\ \nexists m_i \in \mathcal{O}(D_1^t) \text{ where} \\ (m_t, m_i) \in R_a \end{array} \right\}$$
$$\cup$$
$$\left\{ null \;\middle|\; m_t \notin \mathcal{O}(D_1^t) \right\}$$

where $id(m_i)$ returns the id of a move, as defined by Prakken.

Note that this definition allows that several moves in an argument inquiry dialogue have target $\{0\}$. As a matter of fact, any operative move in a top-level argument inquiry dialogue has target $\{0\}$.

Since the operative moves have targets, they can be structured into a tree structure. This tree is neither a dialogue tree nor a dialectical tree; the nodes are moves, so it is somewhat a dialogue tree, yet not all the moves of the dialogue are in the tree. Since the only moves in the tree are operative moves (moves that are in the set $\mathcal{O}$ defined in Definition 70), this type of tree will be refered to as the operative dialogue tree, $\mathcal{O}T$.

A move may appear at multiple locations in the operative dialogue tree if it has multiple targets. Also, since top-level argument inquiry dialogues provide multiple moves with target $\{0\}$, they provide a operative dialogue graph, as there are several disconnected nodes and no real tree structure; this fact is mostly ignored because the operative dialogue graph is neither a useful nor an interesting structure, merely a graph containing every operative move and no edges.

Definition 72. The **operative dialogue tree** of a warrant inquiry dialogue D_1^t (or **operative dialogue graph** of an argument inquiry dialogue, as $Root(\mathcal{O}T(D_1^t))$ is undefined) is denoted $\mathcal{O}T(D_1^t)$ and defined as

$\mathcal{O}T(D_1^t) =$

```
if $Type(D_1^t) = wi$
    $Root(\mathcal{O}T(D_1^t)) = m_i$ such that
        $m_i \in D_r^t$
        and $0 \in t(m_i)$
else if $Type(D_1^t) = ai$
    $\forall m_i$ such that
        $m_i \in D_r^t$
        and $0 \in t(m_i)$
    make a node with label $m_i$
and $\forall m_j \in D_1^t$ such that $t(m_j) \neq null$ and $0 \notin t(m_j)$ :
$m_j$ is a child of each node with label $l \in t(m_j)$
```

6.3.3 Player roles

As was done in Definition 44, the adapted notation for roles is used here.

Since dialogues in Prakken's framework are "assumed to be for two parties arguing about a single dialogue topic", a function is given to assign a role to a move. This way, moves are made by a specific role, but this need not be the same player; any player can make a move using either role.

In order to determine the role of a move, the content of the entire dialogue must be considered. Given a top-dialogue from Black and Hunter's inquiry system, the role of a move in the dialogue is determined by looking at the moves made before it. The move is considered to be made by the *Proponent* if

- the top-level dialogue is an argument inquiry dialogue (these contain a single role, as there are no counterarguments)
- the sub-dialogue the move is in is an argument inquiry dialogue and the topic of the sub-dialogue supports the topic of the top-level dialogue
- the move closes the top-level dialogue (always opened by *Proponent*)
- the move asserts an argument that supports the topic of the top-level dialogue (directly or indirectly)

Definition 73. The function $\mathit{determineRole}(D_1^t)$, of type $\mathit{determineRole} : \mathcal{D} \mapsto \mathit{Roles}$, receives an inquiry dialogue and returns the role of move m_t, where $D_1^t = m_1, \ldots, m_t$.

$$\mathit{determineRole}(D_1^t) = \begin{cases} P(x) & \text{iff } \mathit{Type}(D_1^t) = ai \\ & \text{or } p_t = \mathit{close}_{wi} \\ & \text{or} \\ & \quad \left[\begin{array}{l} \mathit{Type}(\mathit{Current}(D_1^t)) = ai \\ \text{and } \mathit{proRoot}(\mathit{Arg}, D_1^t) \\ \text{where} \\ \mathit{Arg} = \mathit{ruleToArg}(\mathit{Topic}(\mathit{Current}(D_1^t), 1) \end{array} \right. \\ & \text{or} \\ & \quad \left[\begin{array}{l} p_t = \mathit{assert} \\ \text{and } \mathit{proRoot}(c_t, D_1^t) \end{array} \right. \\ & \text{where} \\ & \quad \left[\begin{array}{l} x = \text{player moving } m_t \\ p_t = \text{speech act moved in } m_t \\ c_t = \text{parameter of the speech act} \end{array} \right. \\ O(x) & \text{otherwise} \\ & \text{where } x = \text{player moving } m_t \end{cases}$$

where $\mathit{ruleToArg}$ turns a rule into an argument, as defined in Definition 58.

And *proRoot*, of type $\mathit{proRoot} : \mathcal{A} \mapsto \mathcal{D} \mapsto \mathit{Boolean}$, determines whether an argument supports or opposes the root argument.

```
proRoot(Arg, $D_1^t$) :
    $T$ = operative dialogue tree of $D_1^t$
    if $Arg$ is not in $T$
        $\forall$ node $N$ in $T$ with label $l$ such that
        $(assert(Arg), s(l)) \in R_a$
        add a node with $Arg$ to $T$ as child of $N$
    return $true$ if $\exists\Gamma$ such that
        $\Gamma$ = argumentation line from $Root(T)$
        to a node with $Arg$
        and $\Gamma$ has an odd number of elements
    return $false$ otherwise
```

6.4 Moves and dialogues

Moves are defined as they were in Definition 49. The same role notation is used, and the same functions are used.

Definition 74. A **dialogue** is a sequence of moves from m_r to m_t, denoted D_r^t where $r, t \in \mathbb{N}_0$. The empty dialogue is D_0^0, and for any nonempty dialogue it is true that $1 \leq r \leq t$.

The **first move of a dialogue** D_r^t, move m_r, always contains an $open_\theta(\gamma)$ speech act, as this determines both the **type** and **topic** of the dialogue. The functions $Type$ and $Topic$ that give the type and topic of a dialogue, defined as
$Type(D_r^t) =$

$$\{\ \theta \mid \text{iff } s(m_r) = open_\theta(\gamma)\ \}$$

$Topic(D_r^t) =$

$$\{\ \gamma \mid \text{iff } s(m_r) = open_\theta(\gamma)\ \}$$

6.5 Commitment function

Definition 75. The **commitment store update** function gives the new commitment store of a role $r \in Roles$ after a move m_t.
$C_r^t(r) =$

$$\begin{cases} \emptyset & \text{iff } t \leq 1 \\ C_r^{t-1}(r) \cup \Phi & \text{iff } s(m_t) = assert(\langle \Phi, \phi \rangle) \\ & \text{and } Role(pl(m_t)) = r \\ C_r^{t-1}(r) & \text{otherwise.} \end{cases}$$

6.6 Turntaking function

In Black and Hunter's system, the turntaking function is integrated into the protocol. In order to fit this to the T function in Prakken's framework, it is now explicitly defined, using the adapted notation for roles defined in Definition 44.

Definition 76. $T(D_r^t) =$

$$\begin{cases} \{P(x), O(x)\} & \text{where } P(x), O(x) \in \mathit{Roles} \\ & \text{and } x \neq \mathit{Player}(pl(m_t)) \end{cases}$$

Note that this function is made to be as flexible about the number of players as possible.

6.7 Protocol

Several changes and additions are necessary to Prakken's protocol rules. First, Prakken's protocol conditions are repeated below for convenience. Note that while the notation of the protocol P_{Pr} was changed to Π above, the protocol function Pr will still be used.

Definition 34, repeated[32]**.** $\forall d, m$ if $m \in Pr(d)$ then

R_1 $pl(m) \in T(d)$

R_2 if $d \neq d_0$ and $m \neq m_1$ then $s(m)$ is a reply to $s(t(m))$ according to L_c

R_3 $\forall m' \in d$ if $t(m) = m'$ then $pl(m) \neq pl(m')$

R_4 $\forall m' \in d$ if $t(m) = t(m')$ then $s(m) \neq s(m')$

R_5 $\forall m' \in d$ if $t(m) = t(m')$ and $s(m')$ is a surrendering reply, then m is not an attacking counterpart of m'

As moves are allowed to have multiple targets in this adapted system, rule R_2 has to be altered. Additionally, the protocols in [3] for argument and warrant inquiry need to be merged into a single protocol, and fit to the adapted system with as little changes as possible.

First, rule R_4 is altered; R_4 states that no moves with the same speech act may target the same move. Since moves without target are allowed in this adapted system, R_4 has to be adapted to fit this change. Additionally, the only move that can be repeated is a close move, all others can only be moved once, regardless of target (note that since a move can have multiple targets, this is essentially the same as moving it multiple times with a single target).

Next, *assert* moves have to be restricted, as without a target any *assert* move with any content would be valid. Therefore, inside a warrant inquiry dialogue, an *assert* move needs to change the operative tree to be allowed to move; inside an argument inquiry dialogue, it must assert an argument of which the conclusion contains a literal that is in the topic of the argument inquiry dialogue.

Similarly, a subdialogue can be opened only when the topic of the subdialogue would either change the operative dialogue tree or is a rule that concludes a

[32]Originally defined on page 31

literal that is in the topic of the current dialogue. Additionally, both *open*$_{wi}$ and *open*$_{ai}$ moves are allowed to be moved when the dialogue is empty. Finally, *close* moves may only be moved when they close the current dialogue.

To summarise, the following change and additions are made to the protocol rules:

Replacement for rule 2 Every move is allowed to have multiple targets, or no target at all;

Replacement for rule 4 No move may be moved multiple times, except *close* moves;

Assert moves *assert* moves can be made when they change the operative tree (in a warrant inquiry dialogue), or when they assert a literal that is in the topic of the current dialogue (in an argument inquiry);

Open moves for argument inquiry *open*$_{ai}$ moves can be moved if the rule would change the operative tree if it were moved as an argument (in a warrant inquiry dialogue), or when the rule moved ends with a literal that is in the topic of the current dialogue (in an argument inquiry);

Opening the dialogue *open*$_{wi}$ and *open*$_{ai}$ moves can be moved when the dialogue is empty;

Closing a dialogue *close* moves can only be moved if they close the current dialogue.

The changes to the protocol are now formally defined.

Definition 77. The **protocol** from Black and Hunter's system adds the following protocol rules to those of Prakken's framework.

If $m_v \in Pr(D_r^t)$, then:

R_2 $t(m_v) = \emptyset$, $t(m_v) = \{0\}$ or $t(m_v) = \mathcal{P}(\mathbb{N}_1)$
as determined by $determineTarget(D_r^v)$;

R_4 if $\exists m_u \in D_r^t$ such that $s(m_v) = s(m_u)$ then $Performative(s(m_v)) = close$;

R_7 if $Performative(s(m_v)) = assert$
then $\mathcal{OT}(D_1^t) \neq \mathcal{OT}(D_1^t \cup \{m_v\})$ or

$$\left[\begin{array}{l} Topic(Current(D_1^t) = \Phi \rightarrow \psi \\ conc(Content(s(m_v))) \in \Phi \\ \text{or } conc(Content(s(m_v))) = \psi \end{array}\right.$$

R_8 if $Performative(s(m_v)) = open_{ai}$ and $A = ruleToArg(Content(s(m_v)), 1)$
then $\mathcal{OT}(D_1^t) \neq \mathcal{OT}(D_1^t \cup \{assert(A)\})$ or

$$\left[\begin{array}{l} Topic(Current(D^t_1) = \Phi \rightarrow \psi \\ conc(A) \in \Phi \\ \text{or } conc(A) = \phi \end{array}\right.$$

R_9 if $D^t_r = D^0_0$, then $Performative(s(m_v)) = open_{ai}$ or
$Performative(s(m_v)) = open_{wi}$

R_{10} if $Performative(s(m_v)) = close_{ai}$ or $Performative(s(m_v)) = close_{wi}$
then $Content(s(m_v)) = Topic(Current(D^t_1))$

6.7.1 Termination

Termination in the adapted system is identical to termination in Black and Hunter's system, as described in Definition 18, with one exception: the matched close in Definition 17, which is defined for specifically 2 players, is expanded to encompass all players in $\mathcal{I}$.

Definition 78. Given a dialogue D^t_r and i participants in the set $\mathcal{I}$, a **matched close** for D^t_r occurs at t when

$$\left[\begin{array}{l} \forall m_j \in D^t_{t-i+1} \\ s(m_j) = close_\theta(\gamma) \\ \text{and } \theta = Type(Current(D^{t-i+1}_1)) \\ \text{and } \gamma = Topic(Current(D^{t-i+1}_1)) \\ \text{and } \left[\begin{array}{l} \forall m_k \in D^t_{t-i+1} \\ \text{if } Player(pl(m_j)) = Player(pl(m_k)) \\ \text{then } m_j = m_k. \end{array}\right. \end{array}\right.$$

6.7.2 Well-formed dialogues

The protocol already determines which dialogues are correct. Black and Hunter also define that a dialogue must be terminated or that there is a dialogue that extends the current dialogue and is terminated (meaning it will be terminated in the future).

Definition 79. Any dialogue D^t_r is **well-formed** iff

$$\left[\begin{array}{l} D^t_r \in \Pi \\ \text{and} \\ \exists v \text{ such that} \\ \quad \left[\begin{array}{l} t \leq v \\ \text{and } D^v_r \text{ extends } D^t_r \\ \text{and } D^v_r \text{ terminates at } v \end{array}\right. \end{array}\right.$$

6.7.3 Protocol additions based on underlying logic

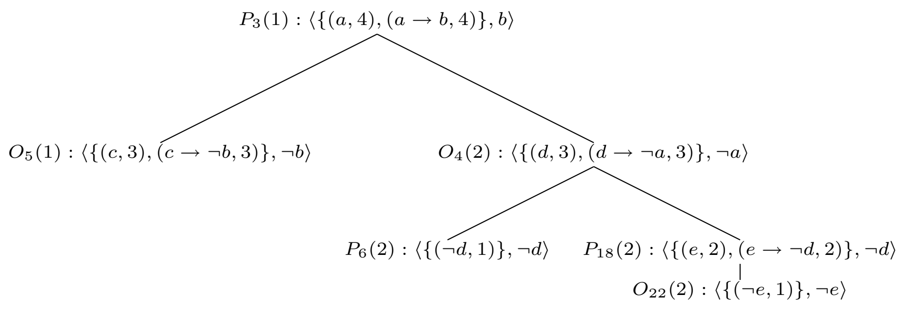
As was discussed in the previous chapter, this adapted system does not specifically consider DeLP or alternate underlying logics, as any underlying logic can be used with the adapted system, simply by adding protocol rules to match the logic. For example, if DeLP were to be used, a protocol rule would be added to enforce acceptable argumentation lines inside the operative dialogue tree.

6.8 Outcome and dialectical tree and graph

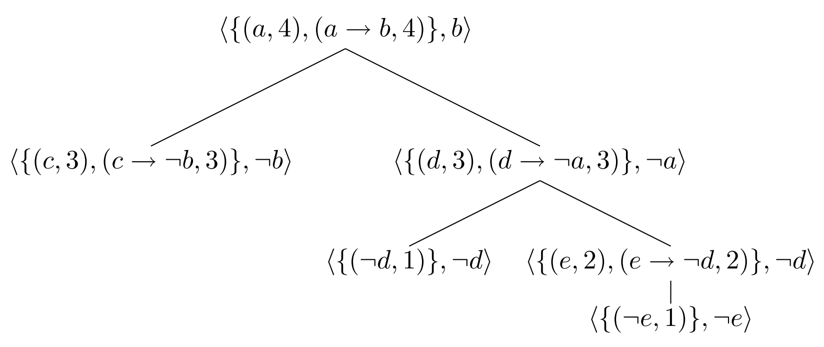
Now that the flow of dialogues has been worked out, the only thing remaining is determining the outcome of a dialogue. Black and Hunter base the outcome on the dialectical tree, while Prakken bases it on the dialogue tree.

6.8.1 Dialectical tree and graph

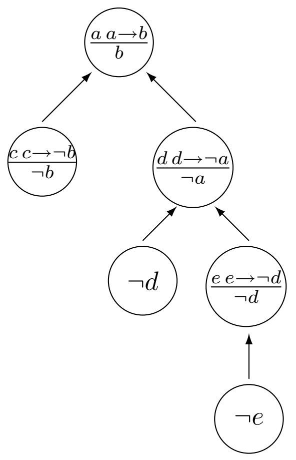
Since not all moves have a target in the dialogue, it is impossible to structure all the moves in a tree shape. However, the operative dialogue tree defined in Definition 72, a tree constructed using the moves that have targets, can be used. This process gives us the following tree for Table 10 (performatives have been omitted as these are all *assert* moves):

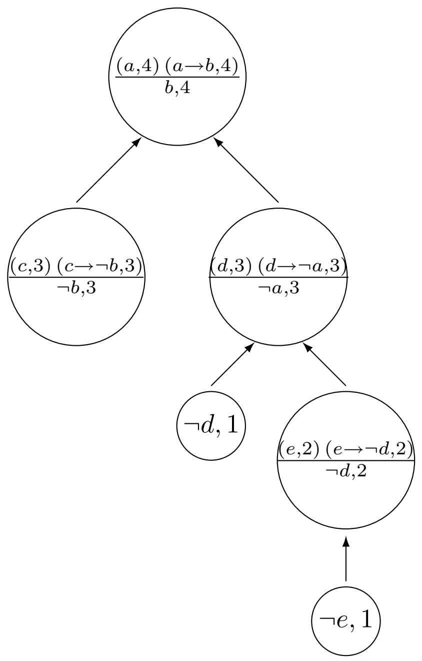

$P_3(1) : \langle\{(a,4),(a \rightarrow b,4)\}, b\rangle$
$O_5(1) : \langle\{(c,3),(c \rightarrow \neg b,3)\}, \neg b\rangle$
$O_4(2) : \langle\{(d,3),(d \rightarrow \neg a,3)\}, \neg a\rangle$
$P_6(2) : \langle\{(\neg d,1)\}, \neg d\rangle$
$P_{18}(2) : \langle\{(e,2),(e \rightarrow \neg d,2)\}, \neg d\rangle$
$O_{22}(2) : \langle\{(\neg e,1)\}, \neg e\rangle$

The operative dialogue tree is very similar to the dialectical tree used by Black and Hunter to determine the outcome of their dialogues; if every node was labelled with the content of the locution of the move, instead of the move itself, the dialectical tree would be produced.


$\langle\{(a,4),(a \to b,4)\},b\rangle$
$\langle\{(c,3),(c \to \neg b,3)\},\neg b\rangle$
$\langle\{(d,3),(d \to \neg a,3)\},\neg a\rangle$
$\langle\{(\neg d,1)\},\neg d\rangle$
$\langle\{(e,2),(e \to \neg d,2)\},\neg d\rangle$
$\langle\{(\neg e,1)\},\neg e\rangle$

Similarly, if the nodes of the tree were labelled with a dialectical representation of the argument that is the content of the locution, the dialectical graph as described in Prakken's framework would be produced, either with or without preference levels. Since it was shown earlier that Black and Hunter's system for inquiry dialogues does not allow argument extension, the dialectical graph differs from the dialectical tree only in notation of the arguments.


$\frac{a \; a\to b}{b}$
$\frac{c \; c\to\neg b}{\neg b}$
$\frac{d \; d\to\neg a}{\neg a}$
$\neg d$
$\frac{e \; e\to\neg d}{\neg d}$
$\neg e$


$\frac{(a,4) \; (a\to b,4)}{b,4}$
$\frac{(c,3) \; (c\to\neg b,3)}{\neg b,3}$
$\frac{(d,3) \; (d\to\neg a,3)}{\neg a,3}$
$\neg d,1$
$\frac{(e,2) \; (e\to\neg d,2)}{\neg d,2}$
$\neg e,1$

Definition 80. The functions $graph(D_1^t)$, $graph_{pLevel}(D_1^t)$ returns a dialectical graph $g_{D_1^t}$ or graph $g_{D_1^t}^{pLevel}$, respectively without and with preference levels. $graph(D_1^t) =$

```
for every node $N \in \mathcal{OT}(D_1^t)$
    replace label of node $N$ with its tableaux notation
    without $pLevels$
```

$graph_{pLevel}(D_1^t) =$

```
for every node $N \in \mathcal{OT}(D_1^t)$
    replace label of node $N$ with its tableaux notation
```

Replacing the labels of node N with their tableaux notation work as follows:

```
if $l$ is the label of $N$ and $l \in \mathcal{S}$, no change is required.
if $l$ is the label of $N$ and $l \notin \mathcal{S}$,
then $l = \langle \Phi, \phi \rangle$ and is converted to $\frac{\Phi}{\phi}$
```

6.8.2 Outcome

Definition 81. The **outcome of an argument inquiry dialogue** D_1^t is defined as a function $Outcome_{ai} : \mathcal{D}_{ai} \mapsto \mathcal{P}(Args)$.

$$Outcome_{ai}(D_1^t) = \text{if } Type(D_1^t) \neq ai, \text{ then } null, \text{ otherwise } \left\{ \; Content(s(m_i)) \;\middle|\; m_i \in \mathcal{O}(D_1^t) \; \right\}$$

Definition 82. The **winner of an argument inquiry dialogue** D_1^t is defined as a function $Outcome_{ai} : \mathcal{D}_{wi} \mapsto Roles$.

$$Winner_{ai}(D_1^t) = \begin{cases} P & \text{iff } Outcome_{ai}(D_1^t) \neq \emptyset \\ O & \text{iff } Outcome_{ai}(D_1^t) = \emptyset \\ null & \text{otherwise} \end{cases}$$

For a warrant inquiry dialogue, this is a more complex procedure; it requires a check for relevance. Luckily, this was already defined in Definition 54, which can be used here as well with a few small changes.

Definition 83. The **last relevant move** *lastRelevant* in a dialogue D_1^t is defined as:

$$lastRelevant(D_1^t) = \begin{cases} m_i & \text{iff } m_i \in D_1^t \text{ and} \\ & m_i \in \mathcal{O}(D_1^t) \\ & \text{the status of } Root(\mathcal{OT}(D_1^t)) \\ & \neq \text{the status of } Root(\mathcal{OT}(D_1^i)) \\ & \text{and } \nexists m_j \in d \text{ such that} \\ & \quad \left[\begin{array}{l} m_j \in \mathcal{O}(D_1^t) \text{ and} \\ id(m_j) > id(m_i) \\ \text{and the status of } Root(\mathcal{OT}(D_1^t)) \\ \neq \text{the status of } Root(\mathcal{OT}(D_1^j)) \end{array} \right. \end{cases}$$

Using the last relevant move, we can determine whether the Proponent role or the Opponent role is winning. If the last relevant move was made using the Proponent role, Proponent is winning.

Definition 84. The **winner of a warrant inquiry dialogue** D_1^t is defined as a function $Outcome_{wi} : \mathcal{D} \mapsto Roles$.

$$Winner_{wi}(D_1^t) = \begin{cases} P & \text{iff } Role(pl(lastRelevant(D_1^t))) = P \\ O & \text{otherwise.} \end{cases}$$

Definition 85. The **outcome of a warrant inquiry dialogue** D_1^t with an operative dialogue tree $\mathcal{OT}(D_1^t)$ is defined as a function $Outcome_{wi} : \mathcal{D} \mapsto Args$.

$$Outcome_{wi}(D_1^t) = \begin{cases} Root(\mathcal{OT}(D_1^t)) & \text{iff } Winner_{wi}(D_1^t) = P \\ null & \text{otherwise.} \end{cases}$$

Basically, if the last relevant move was made using the role of Proponent, the Proponent is currently winning, so the topic of the dialogue is warranted.

6.9 Example dialogues

Several examples of a dialogue in the framework are now given. These are conversions from examples from [3] to a dialogue in the framework. Conversion from the framework back to inquiry can be shown later, as currently, no example dialogue has been provided with the above definitions.

Since the warrant inquiry dialogue in Table 2 on page 27 is the most similar to the framework, it provides a clear example of the conversion process and the structure of a dialogue in the framework.

Role$_{\text{Turn}}$(Player)	Performative(Content)	Target
$P_1(1)$:	$open_{wi}(b)$	
$P_2(2)$:	$close_{wi}(b)$	
$P_3(1)$:	$assert(\langle\{(a,4),(a \rightarrow b,4)\}, b\rangle)$	0
$O_4(2)$:	$assert(\langle\{(d,3),(d \rightarrow \neg a,3)\}, \neg a\rangle)$	3
$O_5(1)$:	$assert(\langle\{(c,3),(c \rightarrow \neg b,3)\}, \neg b\rangle)$	3
$P_6(2)$:	$assert(\langle\{(\neg d,1)\}, \neg d\rangle)$	4
$P_7(1)$:	$open_{ai}(a \rightarrow b)$	
$P_8(2)$:	$close_{ai}(a \rightarrow b)$	
$P_9(1)$:	$close_{ai}(a \rightarrow b)$	
$O_{10}(2)$:	$open_{ai}(d \rightarrow \neg a)$	
$O_{11}(1)$:	$close_{ai}(d \rightarrow \neg a)$	
$O_{12}(2)$:	$close_{ai}(d \rightarrow \neg a)$	

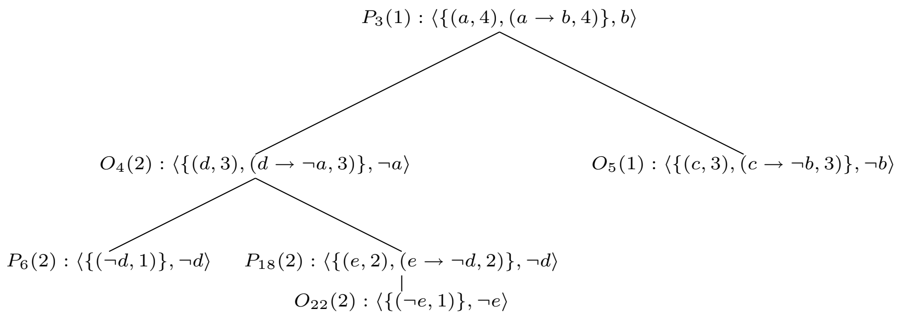
Continued on Next Page…

$\text{Role}_{\text{Turn}}(\text{Player})$	Performative(Content)	Target
$O_{13}(1):$	$open_{ai}(c \rightarrow \neg b)$	
$O_{14}(2):$	$close_{ai}(c \rightarrow \neg b)$	
$O_{15}(1):$	$close_{ai}(c \rightarrow \neg b)$	
$P_{16}(2):$	$open_{ai}(e \rightarrow \neg d)$	
$P_{17}(1):$	$assert(\langle\{(e,2)\}, e\rangle)$	
$P_{18}(2):$	$assert(\langle\{(e,2),(e \rightarrow \neg d,2)\}, \neg d\rangle)$	4
$P_{19}(1):$	$close_{ai}(e \rightarrow \neg d)$	
$P_{20}(2):$	$close_{ai}(e \rightarrow \neg d)$	
$P_{21}(1):$	$close_{wi}(b)$	
$O_{22}(2):$	$assert(\langle\{(\neg e,1)\}, \neg e\rangle)$	18
$P_{23}(1):$	$close_{wi}(b)$	
$P_{24}(2):$	$close_{wi}(b)$	

Table 10: A warrant inquiry dialogue in an adapted version of Prakken's framework

As can be seen clearly from the dialogue above, the difference with the original dialogue is minimal. The structure of the moves is identical, the indexes of the moves would be identical as well, had it not been for an error in the indexing of the moves in the original example.

This dialogue has the following operative dialogue tree:


$P_3(1) : \langle\{(a,4),(a \rightarrow b,4)\}, b\rangle$
$O_4(2) : \langle\{(d,3),(d \rightarrow \neg a,3)\}, \neg a\rangle$
$O_5(1) : \langle\{(c,3),(c \rightarrow \neg b,3)\}, \neg b\rangle$
$P_6(2) : \langle\{(\neg d,1)\}, \neg d\rangle$
$P_{18}(2) : \langle\{(e,2),(e \rightarrow \neg d,2)\}, \neg d\rangle$
$O_{22}(2) : \langle\{(\neg e,1)\}, \neg e\rangle$

Next, the argument inquiry example from the same section as the example above is translated, to show that argument inquiry poses no problem for the adapted framework. As was noted when this example was translated in Table 8, the original example seems to be incomplete; two moves are possible at different stages in the dialogue but are not present in the original example. These two moves have been added to the dialogue below as move 12 and 16, and as a result, a *close* has been dropped, as there was no player that needed to skip a turn at that point.

Role$_{\text{Turn}}$(Player)	Performative(Content)	Target
$P_1(1)$:	$open_{ai}(c \to d)$	
$P_2(2)$:	$close_{ai}(c \to d)$	
$P_3(1)$:	$open_{ai}(b \to c)$	
$P_4(2)$:	$assert(\langle\{(b,1)\}, b\rangle)$	
$P_5(1)$:	$assert(\langle\{(b,1),(b \to c,1)\}, c\rangle)$	
$P_6(2)$:	$close_{ai}(b \to c)$	
$P_7(1)$:	$open_{ai}(a \to b)$	
$P_8(2)$:	$assert(\langle\{(a,1)\}, a\rangle)$	
$P_9(1)$:	$assert(\langle\{(a,1),(a \to b,1)\}, b\rangle)$	
$P_{10}(2)$:	$close_{ai}(a \to b)$	
$P_{11}(1)$:	$close_{ai}(a \to b)$	
$P_{12}(2)$:	$assert(\langle\{(a,1),\ (a \to b,1),\ (b \to c,1)\}, c\rangle)$	
$P_{13}(1)$:	$close_{ai}(b \to c)$	
$P_{14}(2)$:	$close_{ai}(b \to c)$	
$P_{15}(1)$:	$assert(\langle\{(a,1),(a \to b,1),(b \to c,1),(c \to d,1)\}, d\rangle)$	0
$P_{16}(2)$:	$assert(\langle\{(b,1),\ (b \to c,1),\ (c \to d,1)\}, d\rangle)$	0
$P_{17}(1)$:	$close_{ai}(c \to d)$	
$P_{18}(2)$:	$close_{ai}(c \to d)$	

Table 11: An argument inquiry dialogue in an adapted version of Prakken's framework

This dialogue clearly shows how it is possible for an argument inquiry dialogue to provide multiple arguments, and the resulting operative dialogue graph, containing 15 and 16:

$P_{15}(1) : \langle\{(a,1),(a \to b,1),(b \to c,1),(c \to d,1)\}, d\rangle \quad P_{16}(2) : \langle\{(b,1),(b \to c,1),(c \to d,1)\}, d\rangle$

Finally, an example medical dialogue from [3] is translated.

Σ^1	$\{(pain,1),(nonCyc,1),(preg,2)\}$
Σ^2	$\{(pain \land nonCyc \to mam,3),(preg \to \neg mam,1)\}$

Role$_{\text{Turn}}$(Player)	Performative(Content)	Target
$P_1(2)$:	$open_{wi}(mam)$	
$P_2(1)$:	$close_{wi}(mam)$	
$P_3(2)$:	$open_{ai}(pain \land nonCyc \to mam)$	
$P_4(1)$:	$assert(\langle\{(pain,1)\}, pain\rangle)$	
$P_5(2)$:	$close_{ai}(pain \land nonCyc \to mam)$	
$P_6(1)$:	$assert(\langle\{(nonCyc,1)\}, nonCyc\rangle)$	

Continued on Next Page...

Role$_{\text{Turn}}$(Player)	Performative(Content)	Target
$P_7(2)$:	$assert(\langle\{(pain,1),(nonCyc,1),(pain \wedge nonCyc \rightarrow mam,3)\},mam\rangle)$	0
$P_8(1)$:	$close_{ai}(pain \wedge nonCyc \rightarrow mam)$	
$P_9(2)$:	$close_{ai}(pain \wedge nonCyc \rightarrow mam)$	
$P_{10}(1)$:	$close_{wi}(mam)$	
$O_{11}(2)$:	$open_{ai}(preg \rightarrow \neg mam)$	
$O_{12}(1)$:	$assert(\langle\{(preg,2)\},preg\rangle)$	
$O_{13}(2)$:	$assert(\langle\{(preg,2),(preg \rightarrow \neg mam,1)\},\neg mam\rangle)$	7
$O_{14}(1)$:	$close_{ai}(preg \rightarrow \neg mam)$	
$O_{15}(2)$:	$close_{ai}(preg \rightarrow \neg mam)$	
$P_{16}(1)$:	$close_{wi}(mam)$	
$P_{17}(2)$:	$close_{wi}(mam)$	

Table 12: An example from the medical domain in an adapted version of Prakken's framework

This example shows how argument inquiry can be used to build an argument for and against the topic. The tree for this dialogue looks like this:

$$\begin{array}{c} P_7(2) : \langle\{(pain,1),(nonCyc,1),(pain \wedge nonCyc \rightarrow mam,3)\},mam\rangle \\ | \\ O_{13}(2) : \langle\{(preg,2),(preg \rightarrow \neg mam,1)\},\neg mam\rangle \end{array}$$

6.10 Converting dialogues to and from Black and Hunter's system for inquiry dialogues

6.10.1 Move conversion

With the above functions for moves and dialogues defined, moves from Black and Hunter system for inquiry dialogues can now be converted into moves in Prakken's framework, with the adaptions from earlier definitions applied; speech acts are assumed to be elements of the L_c defined in Definition 67 and role notation as defined in Definition 44. The functions *determineRole* and *determineTarget* are defined in respectively Definitions 73 and 71.

Definition 86. Given a dialogue D_1^t consisting of moves $m_1, \ldots, m_t$ in the form of $m_i = \langle x_i,\ performative_i,\ content_i\rangle$ $(0 \leq i \leq t)$, the function **moveToFramework** of type $moveToFramework : \mathcal{D} \mapsto \mathcal{P}(M)$ is defined as

$moveToFramework(D_1^t) = (id, role(player), perf(content), target)$

$$\text{where}\quad \left\| \begin{array}{ll} id & = t \\ role(player) & = determineRole(D_1^t) \\ perf & = performative_t \\ content & = content_t \\ target & = determineTarget(D_1^t) \end{array} \right.$$

The conversion from moves in the framework to moves in Black and Hunter's system is significantly more simple.

Definition 87. Given a move $(id, role(player), perf(content), t)$ using the adaptions from earlier definitions, **moveToInquiry** of type $moveToInquiry : \mathcal{P}(M) \mapsto \mathcal{M}$ is defined as
$moveToInquiry((id, role(player), perf(content), t)) = \langle player, perf, content \rangle$

6.10.2 Dialogue conversion

With the above functions defined, defining the function to convert a dialogue from Prakken's framework to Black and Hunter's system for inquiry dialogues, and vice versa, is a trivial matter. Since dialogues are nothing more than a sequence of moves, to convert a dialogue its moves must simply be converted.

Definition 88. Given a well-formed dialogue D_1^t consisting of moves m_1, $\ldots$, m_t, the function **dialogueToFramework** of type $dialogueToFramework : \mathcal{D} \mapsto \mathcal{P}(M)$ is defined as
$dialogueToFramework(D_1^t) =$
$$\left\{ \; moveToFramework(D_1^i) \;\middle|\; m_i \in D_1^t \; \right\}$$

Definition 89. Given a well-formed dialogue d_t consisting of moves m_1, $\ldots$, m_t, the function **dialogueToInquiry** of type $dialogueToInquiry : \mathcal{P}(M) \mapsto \mathcal{D}$ is defined as
$dialogueToInquiry(d_t) =$
$$\left\{ \; moveToInquiry(m_i) \;\middle|\; m_i \in d_t \; \right\}$$

The following Lemmas are used to construct the proof for Proposition 5.

First, a lemma is provided to show that the structure of dialogues in unaffected by conversion.

Lemma 22. Dialogue conversion makes no changes in the order of moves.

Proof This can be seen from Definition 88 and 89, as dialogue conversion is merely move conversion applied to every move. □

Next, a lemma is provided to show that move conversion to the adapted system only adds information that the adapted system needs, and does not remove anything, and vice versa.

Lemma 23. No information is removed from a move when it is converted to the adapted system. The only information that is added is necessary for the adapted system, and is derived from the dialogue itself.
No information is added to a move when it is converted to Black and Hunter's system. The only information that is removed is unnecessary in Black and Hunter's system.

Proof From Definition 86 and 87. □

Proposition 5. If D_1^t is a well-formed dialogue in Black and Hunter's system and $dialogueToFramework(D_1^t) = d$, then d is a well-formed dialogue in the adapted system. Similarly, if d is a well-formed dialogue in the adapted system and $dialogueToInquiry(d_t) = D_1^v$, then D_1^v is a well-formed dialogue in Black and Hunter's system.

Proof From Lemma 22 and 23 we can conclude that given a well-formed dialogue, the conversion functions provide a well-formed dialogue. □

In order to prove that the conversion provides an equivalent dialogue, but in a different system, the measure of equivalence from Definition 64 can be used.

A proof is constructed to show that a dialogue before the conversion and the dialogue after the conversion are equivalent according to the above criteria. This proof is very similar to the proof constructed for the adapted system in the previous chapter.

The following Lemmas are used to construct the proof for Proposition 6.

Arguments By definition of the move conversion functions, the content of moves is not altered in any way and by definition of the dialogue conversion functions, nothing is done except the conversion of moves.

Lemma 24. The content of assert moves is not altered when moves are converted.

Proof From Definition 86 and 87. □

From this, we can conclude that no argument has been changed, added or removed during conversion.

Lemma 25. Arguments cannot change during dialogue conversion.

Proof From Lemma 24 and 22. □

A function has been provided to create the dialectical tree from a dialogue, using only the operative moves.

Lemma 26. Every move in the dialectical tree in Black and Hunter's system is an operative move in the adapted system.

Proof From Definition 70. □

From the definitions of the dialectical tree and the operative dialogue tree, we can conclude that the operative dialogue tree and the dialectical tree are the same, except for move notation.

Lemma 27. Every operative move must also be in the dialectical tree, and anything in the dialectical tree is in the operative dialogue tree.

Proof From Definition 72. □

Thus, the dialectical trees are identical.

Committed facts In order to prove that the same committed facts have been used, the commitment functions are examined. Both Black and Hunter's system and the adapted system only make changes to the commitment stores when an assert move is made. It was shown above that these are the same, and that there are no new beliefs introduced in these arguments.

Lemma 28. The commitment store of the dialogue before and after translation contains the same facts.

Proof From Lemma 25 we know that the assert moves must be the same. Thus, by Definitions 75 and 20, the same facts will be in the commitment stores. □

We can conclude that there cannot be more committed facts than there are in the original dialogue, but we can also conclude that there cannot be less, by looking at the commitment functions. Since these function are nearly identical (except for the difference of players and roles, which is ignored as it has no consequence for the content of the combined commitment stores), it is safe to say there are no difference in this area either.

Outcome It has already been shown that the dialectical trees are equal. Black and Hunter base the outcome of a dialogue on the status of nodes in the dialectical tree. While the adapted system does not, the outcome is designed to give a similar result: a set of arguments when the dialogue is an argument inquiry dialogue, or a fact that is deemed warranted if the dialogue is a warrant inquiry dialogue.

The outcome of the adapted system is based on the last relevant move, which is based on the operative dialogue tree. Since the operatives dialogue tree has the same structure as the dialectical tree, this is identical to basing the outcome on the dialectical tree.

Lemma 29. The outcome of the dialogue before and after translation is unchanged.

Proof Definition 54 shows that the last relevant move is based on the operative dialogue tree. Lemma 27 shows that this is equal to the dialectical tree, so the outcome is based on the same measure. □

Proposition 6. If D_1^t is a well-formed dialogue in Black and Hunter's system and $dialogueToFramework(D_1^t) = d$, then d and D_1^t are equivalent dialogues. Similarly, if d is a well-formed dialogue in the adapted system and $dialogueToInquiry(d_t) = D_1^v$, then D_1^v and d are equivalent dialogues.

Proof From Lemma 26, 27, 28 and 29, we can conclude that the dialogues are equivalent. □

6.11 Generating dialogues

Aside from a system for inquiry dialogues, Black and Hunter have also provided an agent strategy for use in generating dialogues. They have proven that the dialogues generated by agents using this exhaustive strategy are always sound and complete, based on the agent's belief bases. This strategy can be used with minor changes in the adapted system.

The following minor changes are required, however: the rules for *open* must be fit to $open_{ai}$ and $open_{wi}$ and the same must be done for close moves, and the moves must have targets.

Definition 90. The **exhaustive strategy** $\Omega_x(d)$, for player x participating in a dialogue D_1^t is a function $\Omega_x : \mathcal{D} \mapsto \mathcal{M}$ and is defined as:

$$\Omega_x(D_1^t) = \begin{cases} Pick_a(Asserts_x(D_1^t)) & \text{iff } Asserts_x(d) \neq \emptyset \\ Pick_o(Opens_x(D_1^t)) & \text{iff } Asserts_x(d) = \emptyset \\ & \text{ and } Opens_x(d) \neq \emptyset \\ Close_x(D_1^t)) & \text{iff } Asserts_x(d) = \emptyset \\ & \text{ and } Opens_x(d) = \emptyset \end{cases}$$

where

$$\left\|\begin{array}{ll} Pick_a(\Xi) & = \text{as defined in Definition 26, with altered move notation;} \\ Pick_o(\Xi) & = \text{as defined in Definition 26, with altered move notation;} \end{array}\right.$$

and

$Asserts_x(D_1^t) =$

$$\left\{ \begin{array}{l|l} m & m \in Pr(D_1^t) \text{ and} \\ & s(m) = assert(A) \text{ and} \\ A \in \mathcal{A}(\Sigma^x \cup C_r^t(P) \cup C_r^t(O)) & \end{array} \right\}$$

and

$Opens_x(D_1^t) =$

$$\left\{ \begin{array}{l|l} m & m \in Pr(D_1^t) \text{ and} \\ & s(m) = open_{ai}(R) \text{ and} \\ & \exists p \in \mathbb{N}_1 \text{ such that } (R,p) \in \Sigma^x \\ & \text{and } (R,p) \in \mathcal{R} \end{array} \right\}$$

$\cup$

$$\left\{ \begin{array}{l|l} m & m \in Pr(D_1^t) \text{ and} \\ & s(m) = open_{wi}(\phi) \text{ and} \\ & \phi \in \mathcal{S}^* \end{array} \right\}$$

and

$Close_x(D_1^t) =$

$$\left\{ \begin{array}{ll} m & \text{such that } m \in Pr(D_1^t) \text{ and} \\ & s(m) = close_\theta(\gamma) \text{ such that} \\ & \theta = Type(Current(D_1^t)) \text{ and} \\ & \gamma = Topic(Current(D_1^t)) \end{array} \right.$$

6.11.1 Proof of equivalence

The following Lemmas are used to construct the proof for Proposition 7.

As in the previous chapter, in order to show that the adapted exhaustive strategy produces the same dialogues as the original exhaustive strategy, the dialogue equivalence conditions from Definition 64 is used.

Lemma 30. The $Pick_a$ and $Pick_o$ functions are identical to Black and Hunter's definition, except for notation. The $Asserts_x$ function has been rewritten for the adapted system to reflect the changes in role notation, but is otherwise unchanged.

Arguments Since the only changes in $Asserts_x$ were made to account for the change in notation, $Asserts_x$ generates the same arguments Black and Hunter's $Asserts_x$ function generates.

Lemma 31. $Asserts_x$ provides every argument the original in Black and Hunter's system does.

Committed facts On the basis of the equivalence condition of arguments, this must be the same.

Lemma 32. Given the same belief bases, the committed facts in a dialogue generated by the adapted exhaustive strategy and those in a dialogue generated by Black and Hunter's exhaustive strategy are identical.

Proof From Lemma 31 and the commitment functions in Definition 75 and 20. □

Outcome Black and Hunter prove soundness and completeness on the outcome of the dialogue when the exhaustive strategy is used. This must also be possible with the adapted exhaustive strategy. Fortunately, this can also be based on the fact that the set of arguments is equivalent and the fact that $Opens_x$ in Definition 90 is equivalent to $Opens_x$ defined by Black and Hunter, except for the first move of a dialogue, which is not present in the strategy defined by Black and Hunter. This ensures that a dialogue is created with the same structure as the one that would be created by the original exhaustive strategy.

Lemma 33. The outcome of a dialogue generated by the adapted exhaustive strategy from Definition 90 is equivalent to the outcome of a dialogue generated by Black and Hunter's exhaustive strategy, given the same belief bases.

Proof From Lemma 30, 31 and 29. □

Proposition 7. $\forall d$ that is a dialogue produced by the exhaustive strategy defined in Definition 65, $\exists D_1^t$ such that D_1^t is generated by Black and Hunter's exhaustive strategy and $dialogueToInquiry(d) = D_1^t$ and $\forall D_1^v$ that is a dialogue produced by Black and Hunter's exhaustive strategy, $\exists d'$ such that d' is generated by the exhaustive strategy from Definition 65 and $dialogueToFramework(D_1^v) = d'$.

Proof Lemma 30 through 33 show that the conditions in Definition 64 are met by dialogues generated by Black and Hunter's exhaustive strategy and the strategy defined in Definition 90, so that given the same belief bases, equivalent dialogues are generated. □

6.12 Fundamental properties

As in the previous chapter, the adapted system can be compared to the fundamental properties of Black and Hunter's system for inquiry dialogues. The distinction between agents and roles is clear in the adapted system; this fundamental property has been preserved. The exhaustive strategy has been successfully converted to the adapted system, so the predetermined result and soundness and completeness, benchmarked on the belief bases has also been preserved.

Unlike the previously described adapted system, the simplicity of moves has been preserved; the distinction between *open* and *close* moves for argument and warrant inquiry has been made more explicit, and targets and roles have been added to the moves, but the simplicity is the same.

Black and Hunter use the dialectical tree to determine the targeting of the moves. In a way, the targeting that has been added in this adapted system is similar; it uses the operative dialogue tree, which can be converted to the dialectical tree. By definition, the moves in the operative dialogue tree contain the arguments that appear in the dialectical tree of a dialogue in Black and Hunter's system, and from the definition of the function that assigns target to moves it must be that they are structured in the same way. Argument inquiry dialogues do not produce such a tree, rather they produce a graph of operative moves. This is not problematic, as Black and Hunter do not use the dialectical tree for argument inquiry dialogues either.

Lastly, the cooperative nature of the dialogue has been preserved when viewed from an agent's perspective, but not when viewed from a role. There are specific Proponent and Opponent roles, so the competitive element present in persuasion dialogues has been incorporated into the adapted system. From an agent's perspective, however, these are merely tools which are used when they are convenient. They make explicit whether a move aims to prove or disprove the topic of the dialogue, a property that was implicit in moves in Black and Hunter's system.

6.12.1 Compatibility with Prakken's original framework

Unfortunately, by changing the targeting of moves and allowing zero to many targets, compatibility with proofs for Prakken's framework may be problematic. These proofs assume the basic structure of moves that have a single target.

However, when the operative dialogue tree is compared to a dialogue tree in Prakken's framework there are not only clear similarities; the entire issue of moves without target or moves with multiple targets is not present in the operative dialogue tree. If only the operative moves (as these are the only moves that have targets) are considered to be in the dialogue for Prakken's proofs, and moves with multiple targets are considered to be moved multiple times, the resulting dialogue is compatible with Prakken's proofs.

For example, if the dialogue in Table 10 were reduced as described above, the following dialogue would remain:

Role$_{\text{Turn}}$(Player)	Performative(Content)	Target
$P_1(1)$:	$assert(\langle\{(a,4),(a \rightarrow b,4)\},b\rangle)$	0

Continued on Next Page...

Role$_{\text{Turn}}$(Player)	Performative(Content)	Target
$O_2(2)$:	$assert(\langle\{(d,3),(d \rightarrow \neg a,3)\},\neg a\rangle)$	1
$O_3(1)$:	$assert(\langle\{(c,3),(c \rightarrow \neg b,3)\},\neg b\rangle)$	1
$P_4(2)$:	$assert(\langle\{(\neg d,1)\},\neg d\rangle)$	2
$P_5(2)$:	$assert(\langle\{(e,2),(e \rightarrow \neg d,2)\},\neg d\rangle)$	2
$O_6(2)$:	$assert(\langle\{(\neg e,1)\},\neg e\rangle)$	5

Table 13: A warrant inquiry dialogue in an adapted version of Prakken's framework

It is likely that any dialogue in this adapted system that is reduced as described above results in a dialogue that is not only fully compatible with Prakken's proofs, but even Prakken's original framework. This can be easily explained[33]:

Only fully realised arguments Since only fully realised arguments are present in this form of dialogue, there are no problems with incomplete arguments, sub-dialogues to construct arguments, etc. Of course, this means that this form of dialogue is inadequate for constructing these arguments as is done in the adapted system, but that is not the aim here.

Only a single type of move Since the only type of move left in this form of dialogue is the *assert* move, attack relations become extremely simple, and most proofs that are concerned with move relations could be reduced to examining the logical content of the moves (i.e. proving that the dialogue tree provides the same conclusion as would be reached when reasoned from all asserted arguments; the *assert* moves attack every move they can by definition).

6.13 Summary

This chapter has shown that it is possible to adapt Prakken's framework to fit Black and Hunter's system for inquiry dialogues without losing the fundamental properties of Black and Hunter's system. The cost of the change is significant, as targeting rules have been changed, which has some influence on Prakken's proofs. This means that any statement made about Prakken's framework needs to be verified in the new system to determine whether it is applicable to this adapted system.

If only the operative moves are considered (since the *open* and *close* moves are not relevant for outcome of a dialogue), and moves with multiple targets are duplicated, the dialogue is effectively a normal dialogue in Prakken's framework, and any statement made about the framework is applicable to this dialogue. This way, it is likely that most of Prakken's proofs are applicable to the adapted

[33]Exact proof will be left for future research.

system.

Functions have been provided to convert dialogues to and from the adapted system, with a 1 to 1 conversion. This means that any statement made about a Black and Hunter's system is applicable to this adapted system.

The exhaustive strategy defined by Black and Hunter has also been defined for the adapted system, and the differences are small, reflecting the changes in the speech acts.

Finally, no mention has been made about restrictions on the number of players in $\mathcal{I}$, and the players have been effectively disconnected from the dialogue; they make the moves, but the fact that a move is made by $i \in \mathcal{I}$ has no consequence whatsoever, since Prakken's proofs use the roles, not the agents[34].

[34] Recall that his definition of agents was renamed roles.

7 Conclusion

In the previous chapters, two systems for inquiry dialogues have been described that respectively adapt Black and Hunter's system for inquiry dialogues to Prakken's framework and adapt Prakken's framework to Black and Hunter's system. Both of these are able to describe dialogues that are equivalent[35] to the dialogues described by Black and Hunter's system for inquiry dialogues; any well-formed dialogue that can be formulated in Black and Hunter's system has an equivalent dialogue in the adapted systems, and vice versa.

Additionally, the exhaustive strategy Black and Hunter provided as a deterministic method to construct inquiry dialogues has been translated to both systems, and has been shown to produce dialogues that are equivalent to the dialogues produced by the original exhaustive strategy.

Finally, while the first adapted system is by definition compatible with Prakken's framework, for the second adapted system the proofs and properties that are shown for Prakken's framework can only be applied to a modified version of the dialogue, considering only the operative moves.

$$\begin{tikzcd}
\text{Prakken's framework} & \text{Adapted system} & \text{Black and Hunter's system} \\
d_i \arrow[r, leftrightarrow] \arrow[d] & D_1^x \arrow[r, leftrightarrow] \arrow[d] & D_1^p \arrow[d] \\
d_j \arrow[r, leftrightarrow] \arrow[d] & D_1^y \arrow[r, leftrightarrow] \arrow[d] & D_1^q \arrow[d] \\
d_k \arrow[r, leftrightarrow] & D_1^z \arrow[r, leftrightarrow] & D_1^r
\end{tikzcd}$$

From the results described above, the bisimulation can be concluded. The bidirectional arrows between the adapted system and Black and Hunter's system have been proven[36], as well as the bidirectional arrows between the adapted system and Prakken's framework.

The cost and limitations of this translation will be discussed using the research questions as a guideline.

7.1 Research question

What are the consequences of adapting Black and Hunter's inquiry dialogues to Prakken's persuasion framework, and vice versa, and what properties can we derive from this? What can we say about the system for inquiry dialogues on the basis of its similarities and differences with persuasion dialogues?

[35]According to the definition of equivalence in Definition 64

[36]Fully proven for the first adapted system and partially for the second adapted system.

First, it has been shown that Prakken's framework can indeed be applied to inquiry dialogues, and specifically to Black and Hunter's system for inquiry dialogues. The modification required was significant, however it must be noted that most of the modifications were required because of the structure of dialogues and subdialogues that Black and Hunter use. Using subdialogues to produce an argument is quite useful, as they avoid Prakken's backward extension of arguments, and allow agents to jointly construct these arguments.

The most important alteration was the separation of roles (in the persuasion dialogue) and players (in the inquiry dialogue). This separation is what allows the cooperative nature of inquiry dialogues to be reflected while using persuasion dialogues, allowing players to move as *Proponent* or *Opponent* to the dialogue topic, and even attack moves they made themselves, using a different role.

Since it was shown that dialogues in the adapted system are equivalent to dialogues in Black and Hunter's system, and the same equivalence has been shown for the dialogues generated by the exhaustive strategy provided by Black and Hunter, any property that Black and Hunter have shown for their system also applies to the adapted systems. For example, the proof they provide for soundness and completeness when using the exhaustive strategy can also be applied to the adapted systems.

Proofs and properties from systems in Prakken's framework with compatible protocols can also be applied to the first adapted system without difficulty, as no change has been made to the framework in the first adapted system. In the second adapted system, properties can be applied to the set of operative moves, but this is not proven formally.

Given the above, the differences between Black and Hunter's system for inquiry dialogues and Prakken's framework for persuasion dialogues can be resolved with limited changes to the fundamental properties of both systems. The important properties like the cooperative nature of the dialogue have been preserved, and the fundamental properties that were abandoned have not proved to be problematic; like the absence of move targeting in Black and Hunter's system that was abandoned in the first adapted system, or the requirement of exactly one target per move in Prakken's framework that was abandoned in the second adapted system.

The adapted systems allow for an even more significant conclusion: if, in the adapted systems, a maximum of two players were allowed and these players were directly tied to their role (so for example only $P(1)$ and $O(2)$ are allowed), both systems would describe persuasion dialogues. The differences that are left are directly the consequence of the choices made with respect to the method of argument construction, rather than specific for inquiry dialogues; though the choice of dialogue type might influence the type of argument construction, this is a consideration of the creator of the system, and both methods are practical possibilities in both inquiry and persuasion dialogues.

With this insight, we can have another look at the table in Figure 1. We have concluded above that the difference between persuasion and inquiry dialogues, or Walton and Krabbe's initial states of 'Conflict' and 'Open problem', can be reduced to whether players are tied to roles in the dialogue. It is likely the same might be applied to negotiation and deliberation dialogues.

7.2 Relevance

The field of dialogue games is broad, but research is not equally distributed amongst the different types of dialogue games. Persuasion is the most popular type of dialogue of the types defined by Walton and Krabbe; more properties have been proven of persuasion dialogues and more situations have been described using persuasion dialogues than have been proven of or described with inquiry dialogues.

This thesis has shown that it is possible to overcome the differences between persuasion and inquiry dialogues, allowing inquiry dialogue to be formulated inside a framework for persuasion. Typically, this will only require a separation of players from the roles in the dialogue.

The majority of research into persuasion dialogues can therefore be applied to inquiry dialogues as well, increasing the volume of research that relates to inquiry dialogues significantly.

7.3 Proposed research

Many lines of research can be constructed from this conclusion. First, the proof that the properties of Prakken's framework can be applied to the operative moves of a dialogue in the second adapted system has not been formally given; this provides a possible avenue for additional research. It might also be interesting to see how these results compare to other dialogue types.

In [4] it was shown that Prakken's framework can be adapted to express deliberation dialogues, however Prakken's framework needed to be modified significantly. Perhaps further research might produce similar results for deliberation and negotiation dialogues as have been shown here for persuasion and inquiry dialogues.

Additionally, Walton and Krabbe have provided more types of dialogue for which the same might be investigated. 'Information seeking' is similar to inquiry in terms of goal of the dialogue, but the initial state is determined to be an 'Unsatisfactory spread of information'. This might not require any changes to be formulated inside an inquiry dialogue, where one player provides most of the information.

Another interesting avenue for further research is complexity and performance. The complexity and performance of the systems described or constructed in previous chapters have not been considered. It is possible that the adapted systems produce significantly larger trees than the individual systems by Black and Hunter and Prakken. No measure of complexity has been given for the exhaustive strategy, nor for the exhaustive strategies of the adapted systems. A difference in complexity for the exhaustive strategy might make either system more or less preferred for practical usage.

Finally, the practical applications for these systems are an interesting topic of research. Several scenarios can be constructed in which we could compare one of the inquiry systems in this thesis to other methods for jointly constructing new knowledge.

References

[1] Leila Amgoud, Simon Parsons, and Nicolas Maudet. Arguments, dialogue, and negotiation. *Journal of Artificial Intelligence Research*, 23:2005, 2000.

[2] Elizabeth Black and Anthony Hunter. A generative inquiry dialogue system. In *AAMAS '07: Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1010–1017. ACM, 2007.

[3] Elizabeth Black and Anthony Hunter. An inquiry dialogue system. *Autonomous Agents and Multi-Agent Systems*, 19(2):173–209, 2009.

[4] Henry Prakken Eric M. Kok, John-Jules Ch. Meyer and Gerard A. W. Vreeswijk. A formal argumentation framework for deliberation dialogues. *Argumentation in Multi-Agent Systems: Sixth International Workshop*, 2010.

[5] Alejandro J. García and Guillermo R. Simari. Defeasible logic programming: an argumentative approach. *Theory and Practice of Logic Programming*, 4(2):95–138, 2004.

[6] Peter McBurney and Simon Parsons. Representing epistemic uncertainty by means of dialectical argumentation. *Annals of Mathematics and Artificial Intelligence*, 32(1-4):125–169, 2001.

[7] Simon Parsons, Michael Wooldridge, and Leila Amgoud. On the outcomes of formal inter-agent dialogues. In *AAMAS '03: Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 616–623, New York, NY, USA, 2003. ACM.

[8] Henry Prakken. Coherence and flexibility in dialogue games for argumentation. *Journal of Logic and Compututation*, 15(6):1009–1040, 2005.

[9] Henry Prakken and Giovanni Sartor. Argument-based logic programming with defeasible priorities. *Journal of Applied Non-classical Logics, special issue on 'Handling inconsistency in knowledge systems'*, 7:25–75, 1997.

[10] Douglas N. Walton and Erik C. W. Krabbe. *Commitment in dialogue: Basic concepts of interpersonal reasoning.* State University of New York Press, 1995.