Specifications for Peer-to-Peer Argumentation Dialogues

Bas Testerink and Floris J. Bex

Department of Information and Computing Sciences, Utrecht University, The Netherlands

Abstract. In this paper, we propose a generic specification framework for argumentation dialogue protocols in an open multi-agent system. The specification framework is based on reusable elements – *dialogue templates* – which are realized as an open-source implementation. We provide operational semantics and show formally how templates can be used to determine the possible dialogues. Furthermore, for open multi-agent systems we need to be able to specify *peer-to-peer dialogues*, where the agents themselves are in a position to know whether their dialogue actions are legal according to the protocol without relying on central entities, institutes or middleware. We prove that all protocols that can be specified in our framework are peer-to-peer suitable.

1 Introduction

Dialogue games attempt to capture aspects of structured, mixed-initiative communication with the aim of understanding, improving, and automatically recreating such communication. Particularly for more complex tasks such as software design discussions [7], argument-based negotiation [23], collaborative learning [17] or legal argument [22], it is imperative that the agents' behaviour is constrained by so-called dialogue protocols, so that the agents can argue, inform, investigate and negotiate in a regulated way. We believe that argumentation dialogue games [15] are particularly suited for more complex dialogues, as they nicely straddle the divide between the naturalistic dialogues from the intelligent user interface community [12] and the more rigid multi-agent communication protocols [18]. Furthermore, argumentation dialogue games allow for many features of realistic dialogues, such as agent dialogue strategies and the construction of underlying argument structures, to be captured.

There are several protocol proposals for argumentation dialogue games, many of which have been analysed for their formal properties (see e.g. [1, 21, 9]). The wide range of possible dialogue protocols has given rise to an effort to define generic, executable specification languages for notating protocols [25, 5, 6]. The motivation behind these generic specification languages is twofold. First, these specification languages allow us to represent the syntax and update semantics of any arbitrary dialogue game we care to define in a common language. This allows us to move away from the case-by-case approach common in the literature on computational argumentation and move towards a class-of-systems approach. Second, generic specification languages allow for the development of general purpose execution engines for dialogue games (such as DGEP [3], which has been proposed as an execution engine for DGDL [25]). Such execution engines not only allow us to empirically analyse and compare different protocols, but also take dialogue games out of the academic lab and deploy them in realistic, large-scale settings. Unfortunately, existing specification languages either have no formal operational semantics that allow us to formally study protocols [25, 5], or they lack an implemented execution engine for handling protocol execution [6].

A more general problem of existing argumentation dialogue games in the literature is that, from the perspective of open multi-agent systems, dialogue games are too rigid: dialogues are mostly one-on-one where it is assumed that the participants see all the communication, there is a strict turntaking order and communication is instantaneous, agents are assumed to never fail and always behave according to the protocol, and it is assumed that the internal states of the agents participating in the dialogues is known. These assumptions do not hold in realistic systems, where different dialogues may run in parallel, agents can only be assumed to observe the messages that they themselves send and receive and communication is peer-to-peer rather than governed by a central entity, institute or middleware [16, 10, 8, 3].

As a case-in-point, consider the prototype multi-agent system for the Dutch National Police that we are currently developing [4]. In this system, various agents work together to build a case regarding internet trade fraud (e.g. scammers on eBay or fake online stores). We have agents that interact with human users (victims, police detectives), agents that exchange information with external services (banks, trade sites) and agents that automatically combine and reason with information from these different sources. This distributed, open, peer-to-peer multi-agent system is needed because (i) the police uses strict privacy policies that make it undesirable to centrally gather and reason with data; and (ii) not all participating agents in the system are known beforehand: for instance, the human users that file online complaints are initially unknown.

Removing the usual assumptions of argumentation dialogue games brings extra challenges in the design of these games. Because individual agents each have their view on a dialogue that only includes the messages they themselves sent or received, and because the agents cannot rely on a third-party or control mechanism to determine which dialogue moves are legal, the agent itself then needs to be in a position to know whether its actions break protocol in order to plan a dialogue or formulate strategies. In order for this to work, the protocols need to be *peer-to-peer suitable*. Take, as an example from our implementation domain, the situation where two software agents, A and B, are talking to a human user about one or more criminal complaints. What we do not want is for A to make claims that contradict something B has said , as this would make the police, as an organization, to come across as inconsistent to the human user. If we simply specify the protocol to be ‘A may not contradict a claim that B made towards the user and vice versa’, then this is not peer-to-peer suitable. When A wants to make a claim, then it is not guaranteed to know which messages B sent to the user, and hence whether A’s claim contradicts an earlier claim from B. For the protocol regulating the communication between A, B and the user to be peer-to-peer suitable it would need to include explicit mechanisms to ensure that A and B are consistent (e.g. only allow a claim to be made in the dialogue with the user of which one agent is certain that the other agent agrees). Note that peer-to-peer suitability of a protocol implies that the protocol can be followed without violations by agents, but does not guarantee that there will be no violations. Hence in open multi-agent systems where some agents are not trusted, we may still want to realize a controlling proxy interface to check those untrusted agents.

To summarize, we need a generic, formally underpinned framework for expressing dialogues, protocols and the ways in which those protocols update dialogue structures. Importantly, the protocols expressed in this framework need to be peer-to-peer suitable, that is, the protocols should allow agents to, given their view on a dialogue, determine their legal actions themselves. Inspired by [5], we propose a specification framework for dialogue protocols based on reusable elements called *dialogue templates*[1], and we show formally how templates can be used to determine the possible dialogues. We furthermore define peer-to-peer suitable protocols for dialogues between individual agents, and show that all protocols that can be specified using dialogue templates are in fact peer-to-peer suitable. Our framework includes an execution engine and is part of an open-source multi-agent system.

The rest of this paper is organized as follows. Section 2 discusses related research. In Section 3.1 we formalize the notion of protocol and peer-to-peer suitability. Our specification framework itself is presented in Section 3.2. Finally, in Section 4 we provide conclusions and future work.

2 Related Work

Conversational agents have experienced a resurgence as of late, with much of the work focusing on (spoken) language interpretation issues [12] or on dialogue learning policies for more simple recommendation or negotiation tasks [24]. However, these dialogues are lacking in formal (logical) semantics, which is needed if we want to regulate and formally verify the behaviour of agents in dialogues about more complex and information-rich tasks.

Among the most prominent proposals for the specification of formal protocols for multi-agent systems is Agent UML [2, 18] which has been formally analysed with coloured petri-nets [14, 20]. Formal semantics for these protocols investigate the properties of protocols when they are followed or how agents can interpret locutions, but to our knowledge have not been analysed for determining whether agents are guaranteed to know whether their own behaviour violates a protocol. The work by Pitt and Mamdani [19] focuses on the agent perspective of communication and specifies protocols using finite state machines, but does not address and analyse the protocol specifications directly.

There have recently been various proposals for generic specification languages for argumentation dialogue protocols, such as Dialogue Interaction Diagrams (DID) [13], which can be verified with coloured Petri nets, and the Framework for Dialogical Argumentation (FDA) [6], which defines protocols in an executable logic and simulates dialogues using finite state machines. Whilst these approaches thus allow for the specification and theoretical analysis of different argumentation dialogue protocols, they are not suitable for peer-to-peer argumentation dialogue specifications in implemented open multi-agent systems, for various reasons. First, no mention is made of peer-to-peer

[1] Black and Hunter [6] use a similar concept which they call action rules.

suitability in either approach, and hence they cannot guarantee that for a protocol designed in their respective specification language an agent can determine what the legal moves are given its view on the dialogue. Second, both DID and FDA lack implemented engines for execution protocols, and do not concern themselves with apects such as, for example, connections to software agents and human-computer interfaces.

Recent work that does provide an execution engine for argumentation dialogues is the Dialogue Game Execution Platform (DGEP) [3]. However, this execution platform is not suited for peer-to-peer dialogues in open multi-agent systems, as it uses middleware to provide the agents participating in a dialogue the next possible legal moves, from which the agents have to pick a choice. This 'black box' approach makes it hard for an individual agent to plan ahead: it has to learn a model of the black box or it has to run 'simulations' of dialogues on the black box (if possible) to see possible outcomes of certain strategies. The first option requires a learning phase and offers no guarantees whereas the second option will likely have to deal with an exponential blow-up of simulations. Furthermore, DGEP uses a central knowledge base, the AIFdb [11], and provides the agents with arguments from this central knowledge base; an agent thus cannot use its own private knowledge base to determine which types of utterances can be made in a dialogue. A second shortcoming of DGEP is that as its specification language, DGEP uses the dialogue game description language DGDL [25] and DGDL+ [3], the latter of which is based on the dialogue templates of [5]. While DGDL(+) is very rich and allows for the specification of many different types of dialogue protocols, it has no formal operational semantics. Hence, the outcomes of executing a dialogue protocol with certain knowledge bases and parameters cannot be formally checked or analysed. In this paper we present a template-based protocol specification language inspired by [5], provide operational semantics, and show that all protocols that can be specified in this language allow agents to determine their legal actions themselves.

3 Protocol Specification

In this section we specify our framework. We first model dialogue systems and protocols, and then define the specification of protocols through reusable dialogue templates.

3.1 Peer-to-Peer Protocols

A dialogue system is an abstract description of agents, modelled by atoms, and the locutions, also modelled by atoms, that they may utter.

Definition 1 (Dialogue System, D). *A dialogue system* D *is specified by* (A, L)*, where* A *and* L *are sets of agent and locution atoms, respectively.*

Example 1 (Dialogue System). We will use the same dialogue system throughout our examples. The example scenario is an interaction between some user and expert agent. We will make the scenario quite simple in order to not over complicate full formal specifications of all the aspects of the scenario. We have made a demonstration of a more advanced example publicly available[2]. This demonstration contains implementation examples of the proposed framework, a dialogue among nine different agents, visual representations, and more expanded argumentation-specific aspects of dialogues such as commitments and bookkeeping of arguments that were made. For the remainder of this paper, our example dialogue system D is specified by (A, L) where $\mathsf{A} = \{user, expert\}$ and $\mathsf{L} = \{claim(x)|x \in P\} \cup \{why(x)|x \in P\} \cup \{support(x, y)|x, y \in P\}$ such that $P = \{p, q, r, s\}$. The elements of P are logical atoms, $claim(x)$ can be read as that x is stated to be true, $why(x)$ can be read as a question why x was stated to be true and finally $support(x, y)$ can be read that x follows from y and y is true.

A dialogue system allows for dialogue events, which are the events of sending or receiving a message. An event is specified by its type (sending/receiving), the subject (the sender/recipient), the message (a locution) and the object (the recipient/sender).

Definition 2 (Dialogue Event, e). *Let* $\mathsf{D} = (\mathsf{A}, \mathsf{L})$ *be a dialogue system. A dialogue event* e *is specified by either* $\mathsf{s}(\mathsf{a}, \mathsf{l}, \mathsf{a}')$ *or* $\mathsf{r}(\mathsf{a}, \mathsf{l}, \mathsf{a}')$*, where* $\mathsf{a}, \mathsf{a}' \in \mathsf{A}$ *and* $\mathsf{l} \in \mathsf{L}$.

Example 2 (Dialogue Event). An example event for our scenario is $\mathsf{s}(expert, claim(p), user)$ which can be read as "The expert sends locution $claim(p)$ to the user". Similarly, $\mathsf{r}(user, claim(p), expert)$ can be read as "The user receives the locution $claim(p)$ from the expert".

Dialogue events do not need to occur in sequential order, because agents may reason and communicate in parallel with other agents. We model a dialogue as a sequence of sets of events (we name these sets moments). If multiple events are in the same moment, then we interpret this as those events happening simultaneously. Non-instantaneous communication is captured by the constraint that a moment in which a message is received has to be preceded by some moment in the past where the message was sent. The constraint takes into account that a message might be sent at different moments. Therefore, a message should not be received more often than that it was sent.

Definition 3 (Dialogue, d). *Let* $\mathsf{D} = (\mathsf{A}, \mathsf{L})$ *be a dialogue system and* $\mathbb{E}$ *be all possible dialogue events given* D*. A dialogue* $\mathsf{d} = \mathsf{M}_1 ... \mathsf{M}_k$ *is a finite sequence of sets of events, referred to as moments, such that for each moment* $\mathsf{M}_i \subseteq \mathbb{E}$*,* $i \in [1, k]$*, if* $\mathsf{r}(\mathsf{a}, \mathsf{l}, \mathsf{a}') \in \mathsf{M}_i$ *then the number of moments* M_j*,* $j \in [1, i-1]$*, such that* $\mathsf{r}(\mathsf{a}, \mathsf{l}, \mathsf{a}') \in \mathsf{M}_j$ *is less than the number of moments* M_n*,* $n \in [1, i-1]$*, such that* $\mathsf{s}(\mathsf{a}', \mathsf{l}, \mathsf{a}) \in \mathsf{M}_n$.

Example 3 (Dialogue). Consider a dialogue where the expert claims p, the user asks why the expert thinks p is true, and the expert responds with a support for p by arguing that "p because q". This dialogue could be captured by the moments/dialogue: $\mathsf{M}_1\mathsf{M}_2\mathsf{M}_3$ where $\mathsf{M}_1 = \{\mathsf{s}(expert, claim(p), user)\}$, $\mathsf{M}_2 = \{\mathsf{r}(user, claim(p), expert), \mathsf{s}(user, why(p), expert)\}$, $\mathsf{M}_3 = \{\mathsf{r}(expert, why(p), user), \mathsf{s}(expert, support(p, q), user)\}$.

Consider an alternative sequence of moments $\mathsf{M}'_1\mathsf{M}'_2$ where $\mathsf{M}'_1 = \{\mathsf{s}(expert, claim(p), user), \mathsf{r}(user, claim(p), expert)\}$, $\mathsf{M}'_2 = \{\mathsf{r}(user, claim(p), expert)\}$. This is not a dialogue for two reasons: (1) the send action in moment one was simultaneously

[2] https://git.science.uu.nl/B.J.G.Testerink/OO2APL-P2PArgumentationDialogDemo

received, and (2) the second moment contains a receive event without there being a message that still had to be received.

A protocol specifies how a dialogue ought to be conducted. Hence we can view a protocol as a specification that divides a set of dialogues into those dialogues that follow the protocol and those dialogues that violate it. We note that a protocol violation in a dialogue means that any extension of that dialogue is also a dialogue with a protocol violation. This constraint is captured by requiring that if a dialogue obeys a protocol then all its prefixes do so too (item one in Definition 4). Finally, we add a constraint that receiving a message never violates the protocol. An agent has no control over the messages that it receives, and no control over when a sent message is received. If receiving a message would violate the protocol then the sending agent therefore potentially cannot determine beforehand whether the arrival of the message will break protocol, and the receiving agent cannot avoid the protocol violation since it cannot prevent the message from being received. Hence allowing the receipt of messages to break protocol can bring about situations where the legality of actions is uncertain. If a dialogue violates the protocol, then it must have a smallest prefix (possibly the dialogue itself) that violates the protocol. The final moment of that prefix contains the dialogue events that brought about the violation. If receiving a message cannot violate a protocol, then removing all the send actions from that final moment should give us a dialogue which does not violate the protocol (item two in Definition 4).

Definition 4 (Protocol, P). *Let $\mathbb{D}$ be all possible dialogues given a dialogue system D. A protocol $\mathsf{P} \subseteq \mathbb{D}$ for D is a set of dialogues, such that:*

1. $\forall \mathsf{M}_1 ... \mathsf{M}_k \in \mathsf{P}, i \in [1, k] : \mathsf{M}_1 ... \mathsf{M}_i \in \mathsf{P}$.
2. *For each $\mathsf{d} \in \mathbb{D}$ if $\mathsf{d} \notin \mathsf{P}$ and $\mathsf{M}_1 ... \mathsf{M}_k$ is the shortest prefix of d that is not in P, then $\mathsf{d}' \in \mathsf{P}$ where $\mathsf{d}' = \mathsf{M}_1 ... \mathsf{M}_{k-1}\mathsf{M}'_k$ such that $\mathsf{M}'_k = \{\mathsf{r}(\mathsf{a}, \mathsf{l}, \mathsf{a}') \in \mathsf{M}_k\}$.*

Example 4 (Protocol). As an example protocol we want to specify that a $why(x)$-locution for some propositional atom $x \in \{p, q, r, s\}$ can only be uttered by the user and only as a response to some earlier claim or support that involved x. Similarly, the protocol also specifies that support locutions can only be uttered for supporting earlier claim or support locutions (which in turn are only allowed for the expert). Our example protocol P therefore contains a dialogue d iff for each agent $\mathsf{a}, \mathsf{a}' \in \mathsf{A}$: (a) each send action $\mathsf{s}(\mathsf{a}, why(x), \mathsf{a}')$ in the dialogue is preceded by some previous moment in which $\mathsf{r}(\mathsf{a}, claim(x), \mathsf{a}')$ or $\mathsf{r}(\mathsf{a}, support(y, x), \mathsf{a}')$ is contained and $\mathsf{a} = user$ and (b) each send action $\mathsf{s}(\mathsf{a}, support(x, y), \mathsf{a}')$ in the dialogue is preceded by some previous moment in which $\mathsf{s}(\mathsf{a}, claim(x), \mathsf{a}')$ or $\mathsf{s}(\mathsf{a}, support(z, x), \mathsf{a}')$ is contained, where $x, y, z \in \{p, q, r, s\}$ and $\mathsf{a} = expert$. In summary, protocol P allows the expert to make claims at any time, the user to request support of previously made claims and the expert to support past claims/supports of itself.

To illustrate the second item of Definition 4, consider the dialogue $\mathsf{M}_1\mathsf{M}_2$ such that $\mathsf{M}_1 = \{\mathsf{s}(expert, claim(p), user)\}$ and $\mathsf{M}_2 = \{\mathsf{r}(user, claim(p), expert), \mathsf{s}(user, claim(r), expert)\}$. This dialogue is illegal because the user may not send the claim. The shortest prefix of this dialogue that is not in the protocol is $\mathsf{M}_1\mathsf{M}_2$ itself. Let M'_2 be M_2, except that we remove the send actions (i.e., $\mathsf{s}(user, claim(r), expert)$). Clearly $\mathsf{M}_1\mathsf{M}'_2$ satisfies the protocol.

Agents do not have a total view of the dialogue. In fact, only those events where they send or receive a message can be assumed to be observable by them. We model this view of an agent on a dialogue as a function that filters the moments of the dialogue to those events that concern that agent. Hence, the view of an agent on a dialogue is a sequence of moments where each event has that agent as the subject.

Definition 5 (View, v_{a}). *Let $\mathbb{D}$ be all possible dialogues given a dialogue system $\mathsf{D} = (\mathsf{A}, \mathsf{L})$, $\mathsf{a} \in \mathsf{A}$ be an agent and $\mathsf{d} = \mathsf{M}_1 ... \mathsf{M}_k \in \mathbb{D}$ be a dialogue. For a moment M and agent a let $\mathsf{M}{\downarrow_{\mathsf{a}}} = \{x(\mathsf{a}, \mathsf{l}, \mathsf{a}') \in \mathsf{M} | x \in \{\mathsf{s}, \mathsf{r}\}\}$. The view of a on a moment M in d, notated $v_{\mathsf{a}}(\mathsf{M})$, is the single-moment sequence $\mathsf{M}{\downarrow_{\mathsf{a}}}$ if $\mathsf{M}{\downarrow_{\mathsf{a}}} \neq \emptyset$ and the empty sequence ε otherwise. The view of a on d, notated $v_{\mathsf{a}}(\mathsf{d})$, is the concatenation of the views of each moment: $v_{\mathsf{a}}(\mathsf{d}) = v_{\mathsf{a}}(\mathsf{M}_1) ... v_{\mathsf{a}}(\mathsf{M}_k)$.*

Example 5 (View). The view of the user on the dialogue $\mathsf{M}_1\mathsf{M}_2\mathsf{M}_3$ from Example 3 is $v_{user}(\mathsf{M}_1)v_{user}(\mathsf{M}_2)v_{user}(\mathsf{M}_3) = \varepsilon\mathsf{M}_2'\varepsilon = \mathsf{M}_2'$, where $\mathsf{M}_2' = \{\mathsf{r}(user, claim(p), expert), \mathsf{s}(user, why(p), expert)\}$. Hence, from the user's point of view only one moment has passed in the dialogue, where it responded to a claim with a why question.

We will next specify the class of peer-to-peer suitable protocols. These protocols guarantee that an agent is always in a position to know that its send actions are not violating the protocol. If an agent wants to check whether a dialogue is violating the protocol, then it has to do this using its view on the dialogue. Many different dialogues will look the same to an agent given its view. For each specific agent we want to specify a personal set of dialogues such that the agent can distinguish with its view whether a dialogue is in or out of that set. If a dialogue is out of an agent's dialogue set, then the dialogue violates the protocol (but not necessarily the other way around). A protocol is peer-to-peer suitable if we can define such a set for each agent and furthermore, if a dialogue violates the protocol then there must be at least one agent that can detect this (i.e. the dialogue falls out of that agent's personal set). The latter constraint is captured by requiring that the intersection of all the personal dialogue sets is the protocol itself.

Definition 6 (Peer-to-Peer Suitable). *Let $\mathbb{D}$ be all possible dialogues given a dialogue system $\mathsf{D} = (\mathsf{A}, \mathsf{L})$ and P be a protocol for D. P is peer-to-peer suitable iff for each agent $\mathsf{a} \in \mathsf{A}$ there exists a subset $\mathsf{P}_{\mathsf{a}} \subseteq \mathbb{D}$ such that $\mathsf{P} = \bigcap_{\mathsf{a} \in \mathsf{A}} \mathsf{P}_{\mathsf{a}}$ and for each $\mathsf{d}, \mathsf{d}' \in \mathbb{D}$ if $v_{\mathsf{a}}(\mathsf{d}) = v_{\mathsf{a}}(\mathsf{d}')$ then $\mathsf{d}, \mathsf{d}' \in \mathsf{P}_{\mathsf{a}}$ or $\mathsf{d}, \mathsf{d}' \notin \mathsf{P}_{\mathsf{a}}$.*

Example 6 (Peer-to-Peer Suitable). Our example protocol P from Example 4 is peer-to-peer suitable. We can make P_{user} such that a dialogue is in P_{user} iff each send action $\mathsf{s}(user, why(x), \mathsf{a})$ in the dialogue is preceded by some previous moment in which $\mathsf{r}(user, claim(x), \mathsf{a})$ or $\mathsf{r}(user, support(y, x), \mathsf{a})$ is contained, and there is no occurence of $\mathsf{s}(user, claim(x), \mathsf{a})$ or $\mathsf{s}(user, support(x, y), \mathsf{a})$, for $\mathsf{a} \in \{user, expert\}$. We can make P_{expert} such that a dialogue is in P_{expert} iff each send action $\mathsf{s}(expert, support(x, z), \mathsf{a})$ in the dialogue is preceded by some previous moment in which $\mathsf{s}(expert, claim(x), \mathsf{a})$ or $\mathsf{s}(expert, support(y, x), \mathsf{a})$ is contained, and there is no occurence of $\mathsf{r}(expert, why(x), \mathsf{a})$, for $\mathsf{a} \in \{user, expert\}$. The intersection $\mathsf{P}_{user} \cap \mathsf{P}_{expert}$ forms the protocol. A violation is detected when either agent sends illegally a message. For instance, if the sender violates a protocol, then it sent a why question, which clearly is not allowed by P_{expert}.

Consider another protocol in which the agents must take turns but there are no further restrictions. This protocol could be defined by specifying that each dialogue in the protocol contains only moments where at most one agent sends messages. This protocol is not peer-to-peer suitable. Intuitively, the view of an agent does not allow the agent to distinguish between moments where it alone sends a message and moments where others send messages as well. Therefore, no agent can detect a violation. A solution would be to introduce a special turn-yielding locution in the dialogue system, and specify that (a) an agent may only send messages if it has received the yield-turn locution and has not yet send it after the last time that the locution was received (or if the agent is the starting agent and no messages have been sent yet), (b) sending the yield-turn locution is only allowed if the agent has not sent it since the last time the locution was received (note that this implies that the agent can only yield the turn to a single peer, and it is not restricted which peer this is). We note that turn taking brings with it the risk of agents not yielding their turn and thereby threatening the liveness of a dialogue application. Hence, unlike many protocol specification frameworks for argumentation, we do not assume that turn taking is always part of the protocol.

We show that a peer-to-peer suitable protocol violation can always be detected by one of the agents, that is, one of the agents can, given its view on the violating dialogue, determine that the protocol is violated. At the same time, if the dialogue is not violated, then it is not violated given the views of all the agents. We show this by constructing a set of views per agent that is based on the personal set of dialogues in the definition of peer-to-peer suitability. Given that the protocol is the intersection of the personal sets, we know that a violating dialogue is outside of at least one personal dialogue set. Given further that the personal sets of dialogues are closed under the views of agents, we know that at least one agent can distinguish a bad dialogue from the good ones. If a dialogue is not violating the protocol then it is contained in each personal dialogue set. If we define the set of views per agent as the views on the dialogues in the personal sets, then it follows immediately that a correct dialogue is always in all the sets of views per agent.

Proposition 1. *Let* $\mathbb{D}$ *be all possible dialogues given a dialogue system* $\mathsf{D} = (\mathsf{A}, \mathsf{L})$ *and* P *be a peer-to-peer suitable protocol for* D*. For each agent* $\mathsf{a} \in \mathsf{A}$ *there exists a set of views* $\mathsf{V_a} \subseteq \{v_\mathsf{a}(\mathsf{d}) | \mathsf{d} \in \mathbb{D}\}$ *such that (a) if* $\mathsf{d} \notin \mathsf{P}$ *then there is an agent* $\mathsf{a}' \in \mathsf{A}$ *and* $v_{\mathsf{a}'}(\mathsf{d}) \notin \mathsf{V}_{\mathsf{a}'}$ *and (b) if* $\mathsf{d} \in \mathsf{P}$ *then for each agent* $\mathsf{a} \in \mathsf{A} : v_\mathsf{a}(\mathsf{d}) \in \mathsf{V_a}$.

Proof. From the definition of peer-to-peer suitable we know that for each agent $\mathsf{a} \in \mathsf{A}$ there exists a subset $\mathsf{P_a} \subseteq \mathbb{D}$ such that $\mathsf{P} = \bigcap_{\mathsf{a} \in \mathsf{A}} \mathsf{P_a}$ and for each $\mathsf{d}, \mathsf{d}' \in \mathbb{D}$ if $v_\mathsf{a}(\mathsf{d}) = v_\mathsf{a}(\mathsf{d}')$ then $\mathsf{d}, \mathsf{d}' \in \mathsf{P_a}$ or $\mathsf{d}, \mathsf{d}' \notin \mathsf{P_a}$. Let for each agent $\mathsf{a} \in \mathsf{A}$ the set $\mathsf{V_a}$ be defined as $\{v_\mathsf{a}(\mathsf{d}) | \mathsf{d} \in \mathsf{P_a}\}$. For a dialogue $\mathsf{d} \in \mathbb{D}$ if $\mathsf{d} \notin \mathsf{P}$ and there is no agent $\mathsf{a} \in \mathsf{A}$ such that $\mathsf{d} \notin \mathsf{P_a}$, then it means that d is in each $\mathsf{P}_{\mathsf{a}'}$, $\mathsf{a}' \in \mathsf{A}$, i.e., $\mathsf{d} \in \bigcap_{\mathsf{a} \in \mathsf{A}} \mathsf{P_a}$ (d would be deemed legal by all agents). This cannot be the case as $\mathsf{P} = \bigcap_{\mathsf{a} \in \mathsf{A}} \mathsf{P_a}$. Therefore, if $\mathsf{d} \notin \mathsf{P}$ then there is an agent $\mathsf{a} \in \mathsf{A}$ such that $\mathsf{d} \notin \mathsf{P_a}$. From the peer-to-peer suitability definition we also know that there cannot be a dialogue $\mathsf{d}' \in \mathsf{P_a}$ such that $v(\mathsf{d}) = v(\mathsf{d}')$ if $\mathsf{d} \notin \mathsf{P_a}$. Therefore if $\mathsf{d} \notin \mathsf{P_a}$ then $v_\mathsf{a}(\mathsf{d}) \notin \mathsf{V_a}$, thus proving (a). If $\mathsf{d} \in \mathsf{P}$, then $\mathsf{d} \in \mathsf{P_a}$ for each agent $\mathsf{a} \in \mathsf{A}$, and therefore $v_\mathsf{a}(\mathsf{d}) \in \mathsf{V_a}$ for each agent $\mathsf{a} \in \mathsf{A}$, thus proving (b).

We note that there exist peer-to-peer suitable protocols where only the agent that causes the violation can see the violation. The notion of peer-to-peer suitability could be expanded such that it guarantees that if some send action causes a violation that then the receipt of that message allows the recipient to determine the protocol violation as well. Such an expansion falls out of the scope of this paper.

3.2 Templates for Argumentation Dialogues

We observe that most dialogue systems, such as argumentation dialogue systems, require information about whether a received locution is a response to some past locution. This is not always as straightforward as saying that the last locution is a response to the penultimate locution. In an argumentation dialogue an agent can, for instance, utter an extra support locution in a response to some past question. Since the coherence relation among locutions cannot be derived solely from the legality conditions of a protocol, we include the coherence relation as part of our protocol specification framework. We opt to capture dialogue coherence with graphs in order to accommodate a wide variety of coherence structures (linear lists, trees, and arbitrary graphs). We refer to such a graph as a dialogue graph and assume that a dialogue graph is what agents use to decide upon their next locution(s). As always, a graph consists of nodes and edges. We do not assume any structure on nodes. We add the agent that belongs to a graph to the graph's specification.

Definition 7 (Dialogue Graph, g). *Let* A *be a set of agents. A dialogue graph* g *is specified by* $(\mathsf{a}, \mathsf{N}, \mathsf{E})$*, where* $\mathsf{a} \in \mathsf{A}$*,* N *is a set of nodes and* $\mathsf{E} \subseteq \mathsf{N} \times \mathsf{N}$ *are the edges. We use* $\mathsf{g}^{\emptyset}_{\mathsf{a}}$ *for the empty graph* $(\mathsf{a}, \emptyset, \emptyset)$.

Example 7 (Dialogue Graph). For our example scenario we use dialogue events as nodes in dialogue graphs. As an example graph consider $\mathsf{g} = (user, \mathsf{N}, \mathsf{E})$ where $\mathsf{N} = \{\mathsf{e}_1, \mathsf{e}_2, \mathsf{e}_3, \mathsf{e}_4\}$ and $\mathsf{E} = \{(\mathsf{e}_1, \mathsf{e}_3), (\mathsf{e}_3, \mathsf{e}_4)\}$ such that $\mathsf{e}_1 = \mathsf{r}(user, claim(p), expert)$, $\mathsf{e}_2 = \mathsf{r}(user, claim(q), expert)$, $\mathsf{e}_3 = \mathsf{s}(user, why(p), expert)$, $\mathsf{e}_4 = \mathsf{r}(user, support(p, q), expert)$. This graph represents the user's view on a dialogue where the expert has claimed both p and q in the dialogue. The user has also asked why p is the case, which is why this event is connected to the claim of p. Then, the answer of the expert, that q supports p, is connected to the why question.

As reusable high-level elements for dialogue system specifications we propose dialogue templates [5]. A dialogue template tells for a specific locution under which circumstances it might be sent or received and how this updates the dialogue graph of the sender (or receiver, respectively). The condition of a template is specified by a function that takes a dialogue graph and returns the agents to which the locution might be sent (or from whom it might be received, respectively). The update of a graph is conditioned on the graph that is being updated and the agent to which the locution is sent (or from whom it is received). An update must furthermore produce a graph for the same agent that belongs to the input graph of the function. We can interpret the update function as a specification for how new locutions are related to previous locutions in a dialogue.

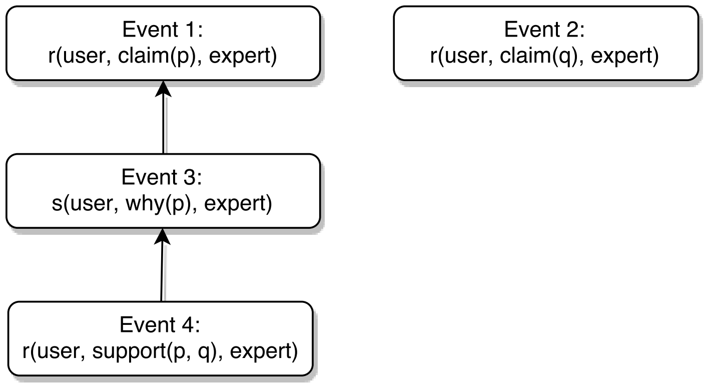
Event 1:
r(user, claim(p), expert)
Event 2:
r(user, claim(q), expert)
Event 3:
s(user, why(p), expert)
Event 4:
r(user, support(p, q), expert)

Fig. 1. Example dialogue graph. See Example 7.

Definition 8 (Dialogue Template, t). *Let* L *be a set of locutions,* A *be a set of agent atoms, and* G *be a set of dialogue graphs. A dialogue template* t *for* L*,* A *and* G *is specified by either* $\mathsf{t_s}(\mathsf{l}, c, u)$ *or* $\mathsf{t_r}(\mathsf{l}, c, u)$*, where* $l \in \mathsf{L}$*,* $c : \mathsf{G} \to 2^{\mathsf{A}}$ *returns to which agents* l *might be sent (or received, respectively) and* $u : \mathsf{G} \times \mathsf{A} \to \mathsf{G}$ *specifies a dialogue graph update given a dialogue graph and recipient (or sender, respectively) such that* $u((\mathsf{a}, \mathsf{N}, \mathsf{E})) = (\mathsf{a}, \mathsf{N}', \mathsf{E}')$.

Example 8 (Dialogue Templates). For all possible dialogue events $\mathbb{E}$ given our example scenario we define the set of all dialogue graphs G to be $\mathsf{G} = \{(\mathsf{a}, \mathsf{N}, \mathsf{E}) | \mathsf{a} \in \mathsf{A} \wedge \mathsf{N} \subseteq \mathbb{E} \wedge \mathsf{E} \subseteq \mathsf{N} \times \mathsf{N}\}$. In the following we use x and y as placeholders for values in $\{p, q, r, s\}$. We also abbreviate the expert and user to e and u, respectively.

For our example scenario we specify the protocol from Example 4 with a set of six dialogue templates. We specify a receive and a send template for each of the locutions. In general there can be multiple templates for receiving/sending the same locution. The templates T for our example scenario are given by:

- $\mathsf{t_s}(claim(x), c^4, u_{\mathsf{s},claim(x)})$, $\mathsf{t_r}(claim(x), c^3, u_{\mathsf{r},claim(x)})$
- $\mathsf{t_s}(why(x), c^1_x, u_{\mathsf{s},why(x)})$, $\mathsf{t_r}(why(x), c^4, u_{\mathsf{r},why(x)})$
- $\mathsf{t_s}(support(x, y), c^2_{x,y}, u_{\mathsf{s},support(x,y)})$, $\mathsf{t_r}(support(x, y), c^3, u_{\mathsf{r},support(x,y)})$

Formally, for a graph $\mathsf{g} = (\mathsf{a}, \mathsf{N}, \mathsf{E})$ the condition functions are defined as:

$$c^1_x(\mathsf{g}) = \begin{cases} \{e\} & \exists \mathsf{r}(u, claim(x), e) \in \mathsf{N} \\ \{e\} & \exists \mathsf{r}(u, support(y, x), e) \in \mathsf{N} \\ \emptyset & \text{otherwise} \end{cases}$$

$$c^2_{x,y}(\mathsf{g}) = \begin{cases} \{u\} & \exists \mathsf{s}(e, claim(x), u) \in \mathsf{N} \\ \emptyset & \text{otherwise} \end{cases}$$

$$c^3(\mathsf{g}) = \{e\}$$

$$c^4(\mathsf{g}) = \{u\}$$

So for instance $\mathsf{t_r}(support(x,y), c^3, u_{\mathsf{r},support(x,y)})$ says that $support(x,y)$ can be received from the expert and if it is received, then the update function $u_{\mathsf{r},support(x,y)}$ is applied to the agent's dialogue graph (specified below).

Intuitively, for a specific agent the update functions specify that sending a why question is a response to either an earlier received claim or received support utterance, receiving a why question is a response to an earlier sent claim or sent support utterance, sending a support utterance is a response to a received why question and receiving a support utterance is a response to an earlier sent why question. We parametrize the update functions with a communication mode (send or receive) and a locution. Formally, for a locution l, a graph $\mathsf{g} = (\mathsf{a}, \mathsf{N}, \mathsf{E})$ and an agent a' the update functions are defined as:

$u_{\mathsf{s},\mathsf{l}}(\mathsf{g}, \mathsf{a}') = (\mathsf{a}, \mathsf{N} \cup \{\mathsf{s}(\mathsf{a}, \mathsf{l}, \mathsf{a}')\}, \mathsf{E}')$
$u_{\mathsf{r},\mathsf{l}}(\mathsf{g}, \mathsf{a}') = (\mathsf{a}, \mathsf{N} \cup \{\mathsf{r}(\mathsf{a}, \mathsf{l}, \mathsf{a}')\}, \mathsf{E}')$
such that $(\mathsf{e}_1, \mathsf{e}_2) \in \mathsf{E}'$ iff:
$\mathsf{e}_1 = \mathsf{r}(\mathsf{a}, claim(x), \mathsf{a}') \in \mathsf{N}$ and $\mathsf{e}_2 = \mathsf{s}(\mathsf{a}, why(x), \mathsf{a}') \in \mathsf{N}$,
or $\mathsf{e}_1 = \mathsf{r}(\mathsf{a}, support(y,x), \mathsf{a}') \in \mathsf{N}$ and $\mathsf{e}_2 = \mathsf{s}(\mathsf{a}, why(x), \mathsf{a}') \in \mathsf{N}$,
or $\mathsf{e}_1 = \mathsf{s}(\mathsf{a}, claim(x), \mathsf{a}') \in \mathsf{N}$ and $\mathsf{e}_2 = \mathsf{r}(\mathsf{a}, why(x), \mathsf{a}') \in \mathsf{N}$,
or $\mathsf{e}_1 = \mathsf{s}(\mathsf{a}, support(y,x), \mathsf{a}') \in \mathsf{N}$ and $\mathsf{e}_2 = \mathsf{r}(\mathsf{a}, why(x), \mathsf{a}') \in \mathsf{N}$,
or $\mathsf{e}_1 = \mathsf{r}(\mathsf{a}, why(x), \mathsf{a}') \in \mathsf{N}$ and $\mathsf{e}_2 = \mathsf{s}(\mathsf{a}, support(x,y), \mathsf{a}') \in \mathsf{N}$,
or $\mathsf{e}_1 = \mathsf{s}(\mathsf{a}, why(x), \mathsf{a}') \in \mathsf{N}$ and $\mathsf{e}_2 = \mathsf{r}(\mathsf{a}, support(x,y), \mathsf{a}') \in \mathsf{N}$.

A template-based dialogue system is a dialogue system that is extended with possible dialogue graphs and templates.

Definition 9 (Template-Based System, D^+). *A template-based dialogue system D^+ is specified by $(\mathsf{A}, \mathsf{L}, \mathsf{G}, \mathsf{T})$ where A and L are sets of agent and locution atoms, respectively, G is a set of dialogue graphs such that $\mathsf{g}^{\emptyset}_{\mathsf{a}} \in \mathsf{G}$ for each $\mathsf{a} \in \mathsf{A}$ and T is a set of dialogue templates for L, A and G.*

We will next specify the operational semantics of a template-based dialogue system with Plotkin style transition rules. These rules give insight in how exactly a dialogue system can transition and help us and agents to reason about possible dialogues that may result from the specification. The runtime configuration of a dialogue system consists of the history of moments that have already passed, the messages that are to be received now or in the future (notated τ_1, these messages were sent in earlier moments), the messages that are to be received in the future (notated τ_2, these messages are being sent in the current moment) and the dialogue graphs of each agent. There are four possible transition rules that apply for a transition: an agent receives a message for which a template is specified, an agent receives a message for which no template is specified (in which case the message has no effect), an agent sends a message or the moment passes. An agent can receive a message if that message was sent in an earlier moment. If there is a template specified for the reception of the message, then this template is used to update the dialogue graph. An agent can only send a message if there is a template that allows this. Finally, if the current moment passes, then all sent messages of the current moment can be received during the following moments.

Definition 10 (Dialogue Operational Semantics, $\mathsf{c}_{\mathsf{D}^+}$). *Let $\mathsf{D}^+ = (\mathsf{A}, \mathsf{L}, \mathsf{G}, \mathsf{T})$ be a template-based dialogue system where $\mathsf{A} = \{\mathsf{a}_1, ..., \mathsf{a}_k\}$. A runtime configuration of*

D^+ *is specified by* $\mathsf{c}_{\mathsf{D}^+} = (\mathsf{d}, \mathsf{M}, \tau_1, \tau_2, \mathsf{g}_1 ... \mathsf{g}_k)$*, where* $\mathsf{d} \in (2^{\mathbb{E}})^*$ *is a sequence of sets of dialogue events,* $\mathsf{M} \subseteq \mathbb{E}$ *is a set of dialogue events,* τ_1 *and* τ_2 *are sets of receive events and* g_i*,* $i \in [1, k]$*, is a dialogue graph* $(\mathsf{a}_i, \mathsf{N}_i, \mathsf{E}_i)$*. The initial runtime configuration of* D^+ *is* $(\varepsilon, \emptyset, \emptyset, \emptyset, \mathsf{g}^{\emptyset}_{\mathsf{a}_1} ... \mathsf{g}^{\emptyset}_{\mathsf{a}_k})$*. The operational semantics of* D^+ *is given by:*

$$\frac{\mathsf{r}(\mathsf{a}_i, \mathsf{l}, \mathsf{a}') \in \tau_1 \;\&\; \exists \mathsf{t}_{\mathsf{r}}(\mathsf{l}, c, u) \in \mathsf{T} : \mathsf{a}' \in c(\mathsf{g}_i) \;\&\; u(\mathsf{g}_i, \mathsf{a}') = \mathsf{g}'_i}{(\mathsf{d}, \mathsf{M}, \tau_1, \tau_2, \mathsf{g}_1 ... \mathsf{g}_i ... \mathsf{g}_k) \xrightarrow{\mathsf{r}(\mathsf{a}_i, \mathsf{l}, \mathsf{a}')}_d (\mathsf{d}, \mathsf{M}', \tau'_1, \tau_2, \mathsf{g}_1 ... \mathsf{g}'_i ... \mathsf{g}_k)} \quad (\textit{Dialogue Receive 1})$$

where $\mathsf{M}' = \mathsf{M} \cup \{\mathsf{r}(\mathsf{a}_i, \mathsf{l}, \mathsf{a}')\}$ *and* $\tau'_1 = \tau_1 \setminus \{\mathsf{r}(\mathsf{a}_i, \mathsf{l}, \mathsf{a}')\}$

$$\frac{\mathsf{r}(\mathsf{a}_i, \mathsf{l}, \mathsf{a}') \in \tau_1 \;\&\; \not\exists \mathsf{t}_{\mathsf{r}}(\mathsf{l}, c, u) \in \mathsf{T} : \mathsf{a}' \in c(\mathsf{g}_i)}{(\mathsf{d}, \mathsf{M}, \tau_1, \tau_2, \mathsf{g}_1 ... \mathsf{g}_i ... \mathsf{g}_k) \xrightarrow{\mathsf{r}(\mathsf{a}_i, \mathsf{l}, \mathsf{a}')}_d (\mathsf{d}, \mathsf{M}', \tau'_1, \tau_2, \mathsf{g}_1 ... \mathsf{g}_i ... \mathsf{g}_k)} \quad (\textit{Dialogue Receive 2})$$

where $\mathsf{M}' = \mathsf{M} \cup \{\mathsf{r}(\mathsf{a}_i, \mathsf{l}, \mathsf{a}')\}$ *and* $\tau'_1 = \tau_1 \setminus \{\mathsf{r}(\mathsf{a}_i, \mathsf{l}, \mathsf{a}')\}$

$$\frac{\exists \mathsf{t}_{\mathsf{s}}(\mathsf{l}, c, u) \in \mathsf{T} : \mathsf{a}' \in c(\mathsf{g}_i) \;\&\; u(\mathsf{g}_i, \mathsf{a}') = \mathsf{g}'_i}{(\mathsf{d}, \mathsf{M}, \tau_1, \tau_2, \mathsf{g}_1 ... \mathsf{g}_i ... \mathsf{g}_k) \xrightarrow{\mathsf{s}(\mathsf{a}_i, \mathsf{l}, \mathsf{a}')}_d (\mathsf{d}, \mathsf{M}', \tau_1, \tau'_2, \mathsf{g}_1 ... \mathsf{g}'_i ... \mathsf{g}_k)} \quad (\textit{Dialogue Send})$$

where $\mathsf{M}' = \mathsf{M} \cup \{\mathsf{s}(\mathsf{a}_i, \mathsf{l}, \mathsf{a}')\}$ *and* $\tau'_2 = \tau_2 \cup \{\mathsf{r}(\mathsf{a}'_i, \mathsf{l}, \mathsf{a}_i)\}$

$$\frac{\tau'_1 = \tau_1 \cup \tau_2}{(\mathsf{d}, \mathsf{M}, \tau_1, \tau_2, \mathsf{g}_1 ... \mathsf{g}_k) \xrightarrow{moment}_d (\mathsf{dM}, \emptyset, \tau'_1, \emptyset, \mathsf{g}_1 ... \mathsf{g}_k)} \quad (\textit{Dialogue Moment})$$

Example 9 (Operational Semantics). For our scenario we specify D^+ as $(\mathsf{A}, \mathsf{L}, \mathsf{G}, \mathsf{T})$, where the elements are drawn from the previous examples. In Table 1 we show the transitions that take place in the dialogue $\mathsf{M}_1\mathsf{M}_2\mathsf{M}_3$ such that $\mathsf{M}_1 = \{\mathsf{s}(e, claim(p), u)\}$, $\mathsf{M}_2 = \{\mathsf{r}(u, claim(p), e), \mathsf{s}(u, why(p), e)\}$, $\mathsf{M}_3 = \{\mathsf{r}(e, why(p), u), \mathsf{s}(e, support(p, q), u)\}$. The transitions are enabled by the dialogue templates T. In the initial state the graphs of the expert and user are their empty graphs (g^0_e and g^0_u). Then the sender sends a claim to the user. This causes its graph to be updated with the $u_{\mathsf{s},claim(x)}$ function, resulting in g^1_e (where the locution is added to its nodes). The send event is added to the current moment and the corresponding receive event to τ_2. This means that somewhere in the future the message will be received. The second transition passes the moment. The receive event switches to τ_1, meaning that the message can be received in the current moment. The other transitions further handle the other messages and moments. The result is intuitively that the operational semantics allow for a dialogue to take place. In the rest of this section we show that the transitions indeed allow for dialogues and that the set of all possible allowed dialogues specifies a peer-to-peer suitable protocol.

Given the operational semantics of a template-based dialogue system we may specify the set of sequences of sets of dialogue events that may result from those transitions, which is called the language of that template-based dialogue system. A sequence of sets of dialogue events d is in the language of a template-based dialogue system if and only if from the initial runtime configuration we can make transitions such that we reach a runtime configuration where d is the first argument of the state.

D	M	τ_1	τ_2	g_e	g_u	Transition
ε	$\emptyset$	$\emptyset$	$\emptyset$	g_e^0	g_u^0	Send
ε	$\{e_1\}$	$\emptyset$	$\{e_2\}$	g_e^1	g_u^0	Moment
M_1	$\emptyset$	$\{e_2\}$	$\emptyset$	g_e^1	g_u^0	Receive1
M_1	$\{e_2\}$	$\emptyset$	$\emptyset$	g_e^1	g_u^1	Send
M_1	$\{e_2, e_3\}$	$\emptyset$	$\{e_4\}$	g_e^1	g_u^2	Moment
M_1M_2	$\emptyset$	$\{e_4\}$	$\emptyset$	g_e^1	g_u^2	Receive1
M_1M_2	$\{e_4\}$	$\emptyset$	$\emptyset$	g_e^2	g_u^2	Send
M_1M_2	$\{e_4, e_5\}$	$\emptyset$	$\{e_6\}$	g_e^3	g_u^2	Moment
$M_1M_2M_3$	$\emptyset$	$\{e_6\}$	$\emptyset$	g_e^3	g_u^2	-

Table 1. Example transitions (see Example 9). A row indicates the left hand side state of a transition and ends with the name of the transition rule that is used to transition to the next row. The events are $e_1 = s(e, l_1, u)$, $e_2 = r(u, l_1, e)$, $e_3 = s(u, l_2, e)$, $e_4 = r(e, l_2, u)$, $e_5 = s(e, l_3, u)$, $e_6 = r(u, l_3, e)$, where the locutions are $l_1 = claim(p)$, $l_2 = why(p)$ and $l_3 = support(p, q)$.

Definition 11 (Dialogue Language, $\mathcal{L}_{D^+}$). *Let* $D^+ = (A, L, G, T)$ *be a template-based dialogue system. The language of* D^+*, notated* $\mathcal{L}_{D^+}$ *are all sequences of sets of dialogue events* d *such that there is a sequence of transitions* $c_{D^+} \rightarrow_d \ldots \rightarrow_d c'_{D^+}$ *possible where* c_{D^+} *is the initial runtime configuration of* D^+ *and* $c'_{D^+} = (d, M, \tau_1, \tau_2, g_1 \ldots g_k)$.

As intended, each element in the language of a template-based dialogue system is a dialogue.

Proposition 2. *Let* $D^+ = (A, L, G, T)$ *be a template-based dialogue system. Each element in* $\mathcal{L}_{D^+}$ *is a dialogue.*

Proof. By definition 3, a sequence of sets of dialogue events $d = M_1 \ldots M_k$ is a dialogue iff for each moment M_i, $i \in [1, k]$, if $r(a, l, a') \in M_i$ then the number of moments M_j, $j \in [1, i-1]$, such that $r(a, l, a') \in M_j$ is less than the number of moments M_n, $n \in [1, i-1]$, such that $s(a', l, a) \in M_n$. The operational semantics only allow a receive transition for an event $r(a, l, a')$ if a corresponding send transition took place for the event $s(a', l, a)$ in a previous moment. Furthermore, if a receive transition takes place, then the event is removed from the runtime configuration. Hence for a new receive event it is required that first a new send transition for that event takes place. This means that a receive event $r(a, l, a')$ can only be in the current moment M_i if the corresponding send event $s(a', l, a)$ is in a past moment M_j, $j < i$, and in between those two moments no receive event took place: $\forall n \in [j+1, i-1] : r(a, l, a') \notin M'_n$. Hence any possible sequence of sets of dialogue events of role based dialogue system fits the definition of a dialogue, and therefore the language of a role based dialogue system is a set of dialogues.

The language of a template-based dialogue system is a protocol. This means that a template-based dialogue system can be seen as a protocol specification.

Proposition 3. *Let* $D^+ = (A, L, G, T)$ *be a template-based dialogue system.* $\mathcal{L}_{D^+}$ *is a protocol.*

Proof. By definition 4, $\mathcal{L}_{\mathsf{D}^+}$ is a protocol iff:

1. $\forall \mathsf{M}_1 ... \mathsf{M}_k \in \mathcal{L}_{\mathsf{D}^+}, i \in [1,k] : \mathsf{M}_1 ... \mathsf{M}_i \in \mathcal{L}_{\mathsf{D}^+}$.
2. For each $\mathsf{d} \in \mathbb{D}$ if $\mathsf{d} \notin \mathcal{L}_{\mathsf{D}^+}$ and $\mathsf{M}_1 ... \mathsf{M}_k$ is the shortest prefix of d that is not in $\mathcal{L}_{\mathsf{D}^+}$, then $\mathsf{d}' \in \mathcal{L}_{\mathsf{D}^+}$ where $\mathsf{d}' = \mathsf{M}_1 ... \mathsf{M}_{k-1}\mathsf{M}'_k$ such that $\mathsf{M}'_k = \{\mathsf{r}(\mathsf{a}, \mathsf{l}, \mathsf{a}') \in \mathsf{M}_k\}$.

Point 1 is easily verifiable; a dialogue $\mathsf{d} \in \mathcal{L}_{\mathsf{D}^+}$ is possible due to a number of consecutive transitions. Hence, all prefixes of d are also in $\mathcal{L}_{\mathsf{D}^+}$ since we can obtain each prefix by executing only an initial segment of the consecutive transitions that create the prefix.

Point 2 follows from the fact that a receive transition can always be made, even if there is no template specified (rule Dialogue Receive 2). The transitions that create $\mathsf{M}_1 ... \mathsf{M}_{k-1}$ result in a configuration $\mathsf{c}_{\mathsf{D}^+} = (\mathsf{M}_1 ... \mathsf{M}_{k-1}, \mathsf{M}, \tau_1, \tau_2, \mathsf{g}_1 ... \mathsf{g}_k)$ where necessarily all the receive events from M_k are in τ_1. Therefore, we can make receive transitions until a configuration $\mathsf{c}'_{\mathsf{D}^+} = (\mathsf{M}_1 ... \mathsf{M}_{k-1}, \mathsf{M}', \tau'_1, \tau_2, \mathsf{g}'_1 ... \mathsf{g}'_k)$ is reached such that $\mathsf{M}' = \{\mathsf{r}(\mathsf{a}, \mathsf{l}, \mathsf{a}') \in \mathsf{M}_k\}$. If we follow this transition with a moment transition, then $\mathsf{M}_1 ... \mathsf{M}_{k-1}\mathsf{M}'$ is created, which is thus in $\mathcal{L}_{\mathsf{D}^+}$, which proves point 2.

Finally our main theorem formulates the goal of this paper: the language of a template-based dialogue system is a peer-to-peer suitable protocol. This means that we can use a specification of a template-based dialogue system as a protocol specification that is suitable for open multi-agent systems.

Theorem 1. *Let* $\mathsf{D}^+ = (\mathsf{A}, \mathsf{L}, \mathsf{G}, \mathsf{T})$ *be a template-based dialogue system.* $\mathcal{L}_{\mathsf{D}^+}$ *is a peer-to-peer suitable protocol.*

Proof. $\mathcal{L}_{\mathsf{D}^+}$ is a protocol, hence we only need to prove that we can make for each agent $\mathsf{a} \in \mathsf{A}$ a set of dialogues $\mathsf{P}_\mathsf{a} \subseteq \mathbb{D}$ such that $\mathcal{L}_{\mathsf{D}^+} = \bigcap_{\mathsf{a} \in \mathsf{A}} \mathsf{P}_\mathsf{a}$ and for each $\mathsf{d}, \mathsf{d}' \in \mathbb{D}$ if $v_\mathsf{a}(\mathsf{d}) = v_\mathsf{a}(\mathsf{d}')$ then $\mathsf{d}, \mathsf{d}' \in \mathsf{P}_\mathsf{a}$ or $\mathsf{d}, \mathsf{d}' \notin \mathsf{P}_\mathsf{a}$. First, let P_a for each agent $\mathsf{a} \in \mathsf{A}$ be defined as $\mathsf{P}_\mathsf{a} = \{\mathsf{d} \in \mathbb{D} | \exists \mathsf{d}' \in \mathcal{L}_{\mathsf{D}^+} : v_\mathsf{a}(\mathsf{d}) = v_\mathsf{a}(\mathsf{d}')\}$. By this construction it follows immediately that for each agent $\mathsf{a} \in \mathsf{A}$ if $v_\mathsf{a}(\mathsf{d}) = v_\mathsf{a}(\mathsf{d}')$ then $\mathsf{d}, \mathsf{d}' \in \mathsf{P}_\mathsf{a}$ or $\mathsf{d}, \mathsf{d}' \notin \mathsf{P}_\mathsf{a}$.

A dialogue d is not in $\mathcal{L}_{\mathsf{D}^+}$ iff there is no sequence of transitions $\mathsf{c}_{\mathsf{D}^+} \rightarrow_d ... \rightarrow_d \mathsf{c}'_{\mathsf{D}^+}$ where $\mathsf{c}_{\mathsf{D}^+}$ is the initial runtime configuration of D^+ and $\mathsf{c}'_{\mathsf{D}^+} = (\mathsf{d}, \mathsf{M}, \tau_1, \tau_2, \mathsf{g}_1 ... \mathsf{g}_k)$. Necessarily at least one unallowed send transition for some event $\mathsf{s}(\mathsf{a}, \mathsf{l}, \mathsf{a}')$ is required to make d (receive and moment transitions are always possible). This means that given the view of a there cannot be a dialogue $\mathsf{d}' \in \mathcal{L}_{\mathsf{D}^+}$ such that $v_\mathsf{a}(\mathsf{d}) = v_\mathsf{a}(\mathsf{d}')$. And thus if $\mathsf{d} \notin \mathcal{L}_{\mathsf{D}^+}$ then there is an agent $\mathsf{a} \in \mathsf{A}$ such that $\mathsf{d} \notin \mathsf{P}_\mathsf{a}$. Furthermore, for each $\mathsf{d} \in \mathcal{L}_{\mathsf{D}^+}$ and agent $\mathsf{a} \in \mathsf{A}$ it holds that $v_\mathsf{a}(\mathsf{d}) = v_\mathsf{a}(\mathsf{d})$, hence if $\mathsf{d} \in \mathcal{L}_{\mathsf{D}^+}$ then $\mathsf{d} \in \mathsf{P}_\mathsf{a}$ for each $\mathsf{a} \in \mathsf{A}$. Therefore $\mathcal{L}_{\mathsf{D}^+} = \bigcap_{\mathsf{a} \in \mathsf{A}} \mathsf{P}_\mathsf{a}$.

We can see a template-based dialogue system as a protocol specification because the language of a template-based dialogue system is a peer-to-peer suitable protocol. The protocol handles the legality of locutions, but does not address the 'low-level' concerns such as ensuring that messages will arrive. The templates ought to be notated in a human/machine readable format in order to make them interpretable for the agents. An agent can, given the protocol, determine all the views on all the dialogues that are allowed by the protocol and use this to strategise. For instance, in a strict turn-based dialogue system it may build a game tree and apply classic decision mechanisms for deciding upon a locution. We note that the templates can formally be quite involved, but at the same time might be quite straightforward in their manifestation as programs. Similarly, the maintenance of dialogue graphs in the formal semantics implies that the graph can grow indefinitely as the dialogue grows, but in an implementation this might not always be the case.

4 Conclusion

In this paper we have presented a formal framework for specifying communication protocols for open multi-agent systems. Our main reusable building blocks for specifying protocols are dialogue templates, which we have provided with a formal operational semantics. In open multi-agent systems we cannot assume that agents have knowledge about the messages that are exchanged among other agents, and an agent can also not assume any order on which messages are received when it sends messages. We formalised such concerns as peer-to-peer dialogues and presented a specification method to specify protocols that guarantee that each agent is in a position to know whether an action from itself causes a protocol violation.

With respect to argumentation, future work consists of further developing the framework to specify argumentation dialogue protocols which specify illocutionary force, that is, how argumentation dialogues can be used to build and analyse logical argument structures. Further work includes adapting the multitude of dialogue protocols specified in DGDL [25] to the current specification framework, representing common elements such as turntaking, commitments, and so on in our framework.

Not enforcing a protocol through a proxy agent or middleware brings up numerous interesting situations for argumentation dialogue systems. In this paper we focus on ensuring that agents can determine whether their own behaviour causes a violation of a protocol. In the future we want to investigate under which conditions it can be guaranteed that an agent can determine that another agent violated the protocol. The current work also assumes that agents adhere to the same protocol, but how does an agent determine the protocol under which it is communicating with other agents when there is no middleware? For which types of protocols/moves and under which conditions are agents guaranteed to pick up protocol violations from other agents? Is it possible to induce which protocols other agents are using from their behaviour? These are all questions we want to answer in future research.

Acknowledgements

This research is part of the project *Intelligence Amplification for Cybercrime* which has been funded by the Dutch National Police Innovation Programme.

References

1. L. Amgoud, N. Maudet, and S. Parsons. Modelling dialogues using argumentation. In *Proceedings of the 4th Conference on Multi-Agent Systems*, pages 31–38. IEEE, 2000.

2. B. Bauer, J. P. Müller, and J. Odell. Agent UML: A formalism for specifying multi-agent software systems. *International journal of software engineering and knowledge engineering*, 11(03):207–230, 2001.
3. F. Bex, J. Lawrence, and C. Reed. Generalising argument dialogue with the dialogue game execution platform. In *Proceedings of COMMA 2014*, volume 266 of *Frontiers in Artificial Intelligence and Applications*, pages 141 – 152. IOS Press, 2014.
4. F. Bex, J. Peters, and B. Testerink. A.I. for online criminal complaints: From natural dialogues to structured scenarios. In *Workshop A.I. for Justice - Proceedings of ECAI 2016*, pages 22–29, 2016.
5. F. Bex and C. Reed. Dialogue templates for automatic argument processing. In *Proceedings of COMMA 2012*, volume 245 of *Frontiers in Artificial Intelligence and Applications*, pages 366–377. IOS Press, 2012.
6. E. Black and A. Hunter. Executable logic for dialogical argumentation. In *Proceedings of ECAI*, pages 15–20. IOS Press, 2012.
7. E. Black, P. McBurney, and S. Zschaler. Towards agent dialogue as a tool for capturing software design discussions. In *International Workshop on Theory and Applications of Formal Argumentation*, pages 95–110, 2013.
8. M. Dastani, D. Grossi, J. C. Meyer, and N. A. M. Tinnemeier. Normative multi-agent programs and their logics. In *First International Workshop on Knowledge Representation for Agents and Multi-Agent Systems*, pages 16–31, 2008.
9. X. Fan and F. Toni. A general framework for sound assumption-based argumentation dialogues. *Artificial Intelligence*, 216:20–54, 2014.
10. M. Hannoun, O. Boissier, J. S. Sichman, and C. Sayettat. Moise: An organizational model for multi-agent systems. In *Advances in Artificial Intelligence*, pages 156–165. Springer, 2000.
11. J. Lawrence, F. Bex, C. Reed, and M. Snaith. Aifdb: Infrastructure for the argument web. In *Proceedings of COMMA 2012*, pages 515–516, 2012.
12. O. Lemon and O. Pietquin, editors. *Data-Driven Methods for Adaptive Spoken Dialogue Systems: Computational Learning for Conversational Interfaces*. Springer-Verlag New York, 2012.
13. A. Maghraby, D. Robertson, A. Grando, and M. Rovatsos. Automated deployment of argumentation protocols. In *Proceedings of COMMA 2012*, volume 245 of *Frontiers in Artificial Intelligence and Applications*, pages 197–204, 2012.
14. H. Mazouzi, A. E. F. Seghrouchni, and S. Haddad. Open protocol design for complex interactions in multi-agent systems. In *Proceedings of the first international joint conference on Autonomous agents and multi-agent systems: part 2*, pages 517–526. ACM, 2002.
15. P. McBurney and S. Parsons. Dialogue games for agent argumentation. In *Argumentation in artificial intelligence*, pages 261–280. Springer, 2009.
16. N. Minsky and V. Ungureanu. Law-governed interaction: A coordination and control mechanism for heterogeneous distributed systems. *TOSEM, ACM Transactions on Software Engineering and Methodology*, 9:273–305, 2000.
17. O. Noroozi, A. Weinberger, H. J. Biemans, M. Mulder, and M. Chizari. Argumentation-based computer supported collaborative learning (abcscl): A synthesis of 15 years of research. *Educational Research Review*, 7(2):79–106, 2012.
18. J. J. Odell, H. V. D. Parunak, and B. Bauer. Representing agent interaction protocols in UML. In *Agent-oriented software engineering*, pages 121–140. Springer, 2001.
19. J. Pitt and A. Mamdani. A protocol-based semantics for an agent communication language. In *Proceedings of IJCAI 99*, pages 486–491, 1999.
20. D. Poutakidis, L. Padgham, and M. Winikoff. Debugging multi-agent systems using design artifacts: The case of interaction protocols. In *Proceedings of the first international joint*

conference on Autonomous agents and multiagent systems: part 2, pages 960–967. ACM, 2002.

21. H. Prakken. Coherence and flexibility in dialogue games for argumentation. *Journal of logic and computation*, 15(6):1009–1040, 2005.
22. H. Prakken. A formal model of adjudication dialogues. *Artificial Intelligence and Law*, 16(3):305–328, 2008.
23. I. Rahwan, S. D. Ramchurn, N. R. Jennings, P. Mcburney, S. Parsons, and L. Sonenberg. Argumentation-based negotiation. *The Knowledge Engineering Review*, 18(4):343–375, 2003.
24. J. Schatzmann, K. Weilhammer, M. Stuttle, and S. Young. A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies. *The knowledge engineering review*, 21(2):97–126, 2006.
25. S. Wells and C. Reed. A domain specific language for describing diverse systems of dialogue. *Journal of Applied Logic*, 10(4):309–329, 2012.