Commitment-Based Negotiation Semantics

Phillip Sloan and Nirav Ajmeri

University of Bristol, Bristol BS8 1UB, UK
{phillip.sloan,nirav.ajmeri}@bristol.ac.uk

Abstract. Negotiation is a key form of interaction in multi-agent systems. Negotiation enables agents to come to a mutual agreement about some goal or plan of action. Current negotiation approaches use traditional interaction protocols which do not capture the normative meaning of interactions and often restrict agent autonomy. These traditional negotiation approaches also have difficulty capturing accountability — a key requirement for creating an ethical agent. This paper seeks to address this gap in maintaining autonomy and establishing accountability requirements during negotiation. We propose Nala, a commitment-based negotiation semantics. Nala uses commitments to help provide normative meaning to agent interactions. The nature of commitments establish accountability requirements between agents in negotiation. We illustrate Nala's usage via a case study using a game scenario where agents participate in negotiation to bring about their goals in a research constrained environment.

1 Introduction

Negotiation is a process that allows multi-agent system (MAS) agents to achieve mutual agreement about a goal or a plan [7]. Agents undertake negotiation by sending offers and counter offers between each other. This process helps manage potential conflicts between agents [30]. For agents to negotiate, they must be able to communicate with one another in an agreed manner. To help with this, interaction protocols can be specified that demonstrate the permitted interactions between agents. If an agent follows the rules of the protocol, they are considered protocol compliant, otherwise they are considered non-compliant.

Contract Net Protocol (CNP) is a specification for negotiation in MAS that is frequently used [25]. CNP specifies two roles, the initiator and the participant. An agent can play either role at any given time, initiators send out offers and participants can send back a responding bid [40]. Interactions in the CNP are modelled with Unified Modeling Language (UML) sequence diagrams which are an operational approach to specifying agent interactions. These methods focus on specifying the message sequence and ordering but these specifications are quite rigid and miss the meaning behind the messages that have been transmitted [10].

Ethical guidelines [24] outline the requirements for creating ethical agents. One of the key suggestions is for agents to be accountable for their actions [24]. Accountability means that an agent is responsible for their actions, and are

expected to explain them when they are asked [6,15]. One of the limitations of the traditional interaction protocols implemented by negotiation protocols, is that the designs are not able to establish meaning and agent accountability [14], they simply define the permitted interactions that agents are allowed to make. Traditional protocols do not clearly outline who is accountable when the protocols are not adhered to [5].

Commitments are a directed conditional relationship between two agents. By creating a commitment, agents are creating a social expectation to satisfy the created commitment [3]. NALA attempts to establish accountability requirements through the commitments created during agent interactions. An accountability requirement is where one agent is accountable to another regarding some expectation that they have [15]. Commitments support agent reasoning and are able to be manipulated which can support desired accountability processes [6]. Commitments are often used to describe business transactions [10] which can often be classed as a form of negotiation [30].

Traditional research into commitments attempts to combine the operational and meaning properties into one specification [4,35,44]. However, more recent approaches consider the operational and meanings of agent interactions to be distinct but complementary from each other [11,12,13]. More recent, information-based approaches look to separate structure from meaning [16,38,22,9], providing a declarative and operational base to model the causality of agent interactions, from which a meaning based protocol such as commitments can be layered on top to understand the normative implications of an agent actions [9].

Contributions We make the following contributions:

- We develop NALA, a commitment-based negotiation protocol. Architecturally, NALA brings the two strands of operational and meaning based protocols together. The agent's low level meanings are specified using an information-based interaction protocol called the Blindingly Simple Protocol Language [38], normative meanings are then provided to the messages by using commitment operators to create and manipulate commitments.
- We present a case study to demonstrate NALA operationally. We use the Coloured Trails (CT) environment [21] to help demonstrate NALA operationally.

Structure The rest of the paper is organised as below. Section 2 reviews relevant related work. Section 3 introduces the relevant technical concepts required to understand this paper and formalises the specification of the commitment-based negotiation protocol, NALA. Section 4 provides a case study in a developed CT test-bed which demonstrates NALA in practice. Section 5 concludes the paper, and discusses potential future work.

2 Related Works

Research on formalising commitments and formalising protocols are related to our contribution. We now discuss relevant works in these areas.

2.1 Formalising Commitments

Literature on norms is well established. Previous literature in multi-agent systems often accepts the social definition of norms [18,33], defining a norm as a behavior that is considered a widely accepted practise within society. This research uses the deontic definition of norms, where an agent holds an expectation towards another agent, this definition was adopted as it more closely mirrors a negotiation and has been used by recent literature which uses negotiation [2,29].

The relationship between communication and commitments has also been examined extensively, with Singh [37] and Colombetti [17] among the first to research commitments and Agent Communication Languages. Various commitment lifecycles have been suggested by researchers [19,31,39]. The formalisation of commitments are often based on life cycles by using operational semantics. We follow Kafali et al.'s [28] definition of commitment life cycle which is based on Singh's [39] formalism of norms.

2.2 Formalising Protocols

Negotiation protocols use interaction protocols to help specify the interactions between agents and their control flow, the legal ordering of messages that can occur. The Contract Net Protocol [40] is a commonly used protocol which is formalised operationally through the use of UML Sequence Diagrams [26], UML sequence diagrams are a commonly used interaction protocol for formalising MAS negotiation protocols [41,20,36] which was also expanded into Agent UML [42,23,34]. UML is a graphical notation which focuses on message interchange between life lines. Petri-Nets have also been used to model the interactions of MAS negotiation protocols [8,27] which also uses graphical notation. The problem with these tools is that there is no clear way to define the meaning of interactions and they are rigid in their implementation [10]. Instead of modelling with these protocols, this research uses BSPL [38] and commitments [43,37] to enable a flexible negotiation protocol to be composed, facilitating agent autonomy.

Aydogan et al. [2] specify an agent-based automated negotiation framework called Nova, where agents negotiate over a set of norms. The norms used within Nova are commitments, authorisations and prohibitions. The negotiation protocol used within this framework is the stacked alternating offers protocol which gives agents the ability to accept, reject or counter offers. Counter offers are revised by norms. Nova is similar to Nala because both use norms to model the interactions, allowing for agent accountability to be established. Nova's research was focused on utilising one agent that they designed and used this agent to play against multiple human agents. This is opposed to Nala which focuses on the low level messages.

3 Nala

3.1 Preliminaries

Blindingly Simple Protocol Language (BSPL) BSPL [38] is a text based, declarative protocol language that focuses on modularity and information-based interactions between agents. BSPL does not have any control-flow abstractions, instead BSPL protocols are messages between two agents specifying the flow of information through its parameters. An atomic BSPL protocol is simply a message between two agents, and any higher BSPL protocol are composed protocols, comprised of multiple messages or other smaller protocols [38]. There are several components to a BSPL protocol schema, each protocol has a name, a set of roles, at least one information parameter and at least one reference to a protocol. A role is a conceptual position that an agent can play during an enactment of the protocol. A protocol can also contain a reference to another protocol.

Information parameters are placeholders in BSPL for the information being exchanged between agents. They can be public or private, with private parameters being used internally by a protocol [16]. At least one parameter must be labeled as the key parameter, uniquely identifying each enactment of the protocol. Public parameters are adorned with in or out, which specifies if a parameter must either be known or generated by the protocol when a message is sent, respectively. Parameters can not be bound more than once during an enactment of a protocol, but all public parameters must be bound during an enactment for it to be considered a safe and complete enactment.

One of the main principles of BSPL is that it separates the structure of interactions away from its meaning [38]. BSPL abstracts away the information being passed into parameters and focuses entirely on the operational details of a protocol which allows meaning to be overlayed on top of the protocol, usually in the form of meaning-based protocols such as commitments [38].

Commitments Following Singh and Chopra's [10] definition, a commitment is an expression in the form of C(creditor, debtor, antecedent, consequent). In this expression, the debtor and creditor are agents while the antecedent and consequent are propositions related to some task or resource. A commitment expresses a conditional and directed relationship between the two agents [28]. A commitment means that the debtor commits to the creditor, that if the creditor bring about the antecedent proposition then the debtor will bring about the consequent proposition [10].

Example 1 (Example of a commitment). Consider the commitment C(Alex, Becka, £2, cup of tea). In this commitment, Alex is committing to Becka, that if Becka pays £2, Alex will provide Becka with a cup of tea. The accountability requirements are demonstrated through commitment violations, if Becka pays £2, Alex becomes accountable for providing a cup of tea to Becka. Alex can decide to provide a cup of tea and satisfy his commitment to Becka or not provide a cup of tea and violate this commitment, causing him to become accountable [15].

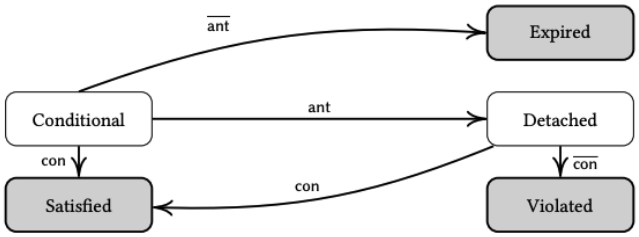

Fig. 1. Illustrating the commitment norm lifecycle. White rectangles represent non-terminal states while grey rectangles are terminal states for the lifecycle [28]

In a commitment, the accountable party is the debtor [15,28]. A commitment lifecycle generally starts in a conditional state where neither the antecedent nor consequent are true. When the antecedent becomes true, the commitment is then considered to be in force, and will move into the detached state. A commitment can start with its antecedent being true, transitioning the commitment into the detached state. If the creditor of a commitment cannot bring about the antecedent, then the commitment will enter the expired state. From the detached state, if the consequent becomes true, then the commitment will move into the satisfied state, with the debtor of the commitment being released from the commitment. If the debtor cannot bring about the consequent, then the commitment will move into the violated state of the lifecycle.

Commitments are created and manipulated via operations which provides flexibility [10] and helps to better reflect real life. Here, x and y signifies agents, while r and u are the antecedent and consequent propositions.

- **CREATE**(x,y,r,u): x establishes a commitment toward y, a CREATE operation can only be performed by the debtor of the commitment [39].
- **CANCEL**(x,y,r,u): x the debtor cancels the commitment, causing it to not hold.
- **RELEASE**(x,y,r,u): y the creditor withdraws the commitment, causing it to not hold.
- **DECLARE**(x,y,r): is performed by x to let y know that the proposition r holds.

By using these commitment operations, the meaning of interactions can be can observed and defined in terms of how they create and manipulate the commitments. Commitment based protocols are an effective way of specifying the normative meanings of interactions [14,28]. In the example above, Alex can violate the commitment by not providing a cup of tea after being paid. By doing this, Alex uses the CANCEL operation to violate the commitment. By violating his commitment, Alex becomes accountable to Becka. In Nala, commitments will be mapped on top of the operational messages which have been specified in

BSPL to provide normative meaning which is something lacking from current approaches [10].

3.2 Nala Lifecycle

We develop Nala by formalising the operational and meaning based aspects of the protocol using BSPL and commitments. Consider Example 1 in Section 3.1 where Alex offers Becka a cup of tea for £2 and Becka accepts the offer. This is the most basic example of a negotiation, called a one-shot or ultimatum negotiation. There is a single offer that can be accepted or rejected [7]. In Nala, we extend this one shot protocol to allow counter-offers, where the agent rejects the previous offer and makes a new offer that replaces it [1]. For example, Becka could think that £2 is too expensive, and offer £1 instead. Based on this form of negotiation this paper proposes a negotiation life cycle which can be seen in Figure 2. For consistency, we use the terms of the commitment life cycle in our definition.

Definition 1. *A negotiation N is a tuple in the form of N(ID, debtor, creditor, antecedent, consequent, state). ID* $\in \mathbb{N}$ *is an identifier which uniquely identifies each negotiation; Debtor and creditor are roles agents use during a negotiation. Antecedent and consequent are propositions which are being negotiated over. State is the current state of the negotiation which can be one of the following:* (inactive, active, successful, failed, expired).

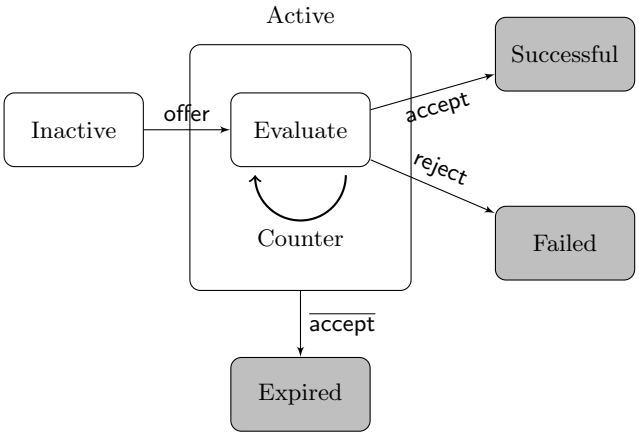

Fig. 2. Negotiation Lifecycle – White states are non-terminal states and Grey states are terminal states.

Figure 2 demonstrates lifecycle of a negotiation. The nodes of the lifecycle represent states that the lifecycle can be in at any point, edges are messages that are sent between agents that can manipulate the current state of a negotiation. The states of a negotiation describe what is happening at any point in time.

Inactive State: The inactive state is the starting state for a negotiation where no agent has attempted any kind of negotiation. A negotiation can remain in an inactive state if no agents attempt a negotiation.
Active State: If an offer has been sent then the negotiation will transition into an active state. In the active state, agents receive offers or counter offers and actively evaluate the offers.
Evaluate State: Evaluate is a sub-state of the active state, which implies that agents can only be evaluating offers and counter offers when a negotiation is in an active state. An agent can create three types of messages from the evaluate state. An accept, reject or counter message.
Failed State: A failed state in the life cycle means that no commitments were made between the agents, they were unable to come to mutual agreement, ending the round of negotiations.
Expired State: If the agents fail to come to an agreement after a defined amount of time has passed then the negotiation will enter the expired state, this means that no agreement was made between the two agents.
Successful State: A successful state means that the agents have negotated successfully, and a pair of opposite commitments (C(x,y,r,u) and C(y,x,u,r)) have been established, these opposite commitments will now be referred to as reciprocal commitments. The agents are both conditionally committed to one another to perform what they agreed in the negotiation. These reciprocal commitments will follow the commitment life cycle (illustrated in Figure 1).

3.3 Messages in Nala

Agents can change the state of a negotiation and commitment by interacting with each other through messages. They are the principle mechanism used by agents to communicate their intentions during a negotiation. Definition 2 defines messages within Nala. Messages within Nala are mapped to commitment operators to help provide a higher level meaning to the interactions.

Definition 2. *A message is a tuple in the form of type(ID, sender, debtor, creditor, antecedent, consequent), where Type* $\in$ {*offer, accept, reject, counter, detached, satisfied, release, expire, cancel*}*; ID* $\in \mathbb{N}$ *is an identifier which uniquely identifies each negotiation path; Sender identifies the agent who is sending the message; Debtor and creditor are agent roles; antecedent and consequent are propositions that the roles are negotiating over.*

Offer: An offer message uses the create commitment operator to instantiate a conditional commitment directed from its debtor to its creditor. An offer message will move a negotiation from an inactive state to an active state.
Accept: An accept message translates to the create commitment operator to instantiate a second reciprocal commitment and transitions the negotiation into the successful state.
Reject: A reject message utilises the release commitment operator, which causes the created commitment to expire. A reject message moves the negotiation into the failed state.

Counter: A counter message uses the create commitment operator to create a new, but related commitment from the related offer message. A counter message does not change a negotiations state, the negotiation remains in the active state as agents are still evaluating offers.

If a negotiation enters the successful state, then two reciprocal commitments are instantiated. The reciprocal commitments start in a conditional state. After negotiating to the successful state, the agents utilise a further set of messages to manipulate the life cycle of these reciprocal commitments.

Detached: A detached message uses the declare commitment operator to inform the other agent that they fulfilled a proposition that was agreed during the negotiation. When the message is sent, one of the reciprocal commitments enters the detached state (its antecedent was fulfilled) while the other commitment is automatically satisfied (its consequent was fulfilled).

Satisfied: A satisfied message translates to the declare commitment operator which is used to inform the other agent that they fulfilled a proposition that was agreed during the negotiation. The agent is declaring that the consequent proposition of the remaining commitment has been fulfilled, transitioning this commitment into the satisfied state.

Release: A release message utilises a release commitment operator to expire one of the reciprocal commitments.

Expire: An expire message uses a release commitment operator. It is used by the second agent to expire the remaining reciprocal commitment after a release message.

Cancel: A cancel message utilises a cancel commitment operator to violate an active commitment. This message establishes an agents accountability requirement towards another agent.

3.4 Operationalising Agent Interactions

The meaning behind operations has now been expressed through commitments and how the negotiation and commitments are manipulated has been specified. The ordering of messages has not yet been clearly defined. In this paper we use BSPL to help specify the operational aspects of the agent interactions. This interaction protocol language uses the flow of information to help express the causality of messages.

```
1 protocol Negotiation {
2   Roles: Debtor, Creditor
3   Public Parameters out ID, out result
4   Private Parameters offered, requested, rejected
5
6   Debtor ↦ Creditor: Offer [out ID, out offered, out
        requested]
7   Accept(Creditor, Debtor, in ID, in offered, in requested, out
        result)
```

```
8  Creditor ↦ Debtor: Reject [in ID, in offered, in requested,
       out result]
9  Creditor ↦ Debtor: Reject [in ID, in offered, in requested,
       out rejected]
10 Counter(Creditor, Debtor, in ID, in requested, in rejected,
       out result)
11}
```

Listing 1.1. Negotiation protocol

Listing 1.1 shows the Negotiation protocol. This protocol is the highest level protocol within this specification, it is a composed protocol which consists of references to both messages and sub-protocols. An enactment of negotiation requires two agents to fulfil the roles of creditor and debtor.

Parameters: There are several parameters outlined on lines 3 and 4. The public parameters are ID (key) and result. Both of these parameters are adorned with out, this information must be generated by enacting this protocol. ID has the qualifier key, which states that it must be unique. ID is what differentiates one negotiations from another. The private parameters offered and requested are parameters that bind what is being negotiated about. The rejected parameter is used if and only if a counter offer is being created. Private parameters are not required to be bound.
Offer Message: Line 6 illustrates an offer message. The message is directed from the debtor to the creditor and has three parameters, ID, offered and requested. All of these parameters are adorned with out, which forces the debtor to generate information to bind to these parameters when they send the message. By reviewing the rest of the protocol, it is clear that offer is the only message that can bind these parameters, this means any enactment of negotiation is required to start with an offer message.
After an offer message has been created, the only public parameter remaining to be bound is result. By reviewing the protocol it can be seen that this parameter is generated by the Offer protocol on line 7, the reject message on line 8 and the Counter protocol on line 10. This is significant because parameters are immutable and can only be bound once, only one of these messages can be sent to complete an enactment of the negotiation protocol.
Accept Protocol: Line 7 illustrates the Accept protocol. Accept is not a message but another sub-protocol of negotiation. The order of the roles in Accepts interface is important. The Creditor and Debtor are reversed, which allows them to switch roles in an enactment of Accept. The parameters ID, offered and requested are all adorned with in which means to enact this protocol, these parameters must be known. Result is adorned with out, this parameter must be generated from the Accept protocol.
Reject Messages: Lines 8 and 9 demonstrate two alternative reject messages. Both messages are directed from the creditor to the debtor and contain the parameters ID, offered and requested are all adorned with in which means the creditor must know these parameters to be able to send this message. The final

parameter is different. The reject message on line 8 has the parameter result which is adorned with out. By sending this message, the agent would be binding result and therefore completing the current enactment of the negotiation protocol. Line 9's final parameter is the private parameter rejected, it is adorned with out, so the creditor of the message needs to generate this information, by sending this message, it does not finish the enactment of the negotiation.
Counter Protocol: Line 10 demonstrates the Counter sub-protocol. Like with Accept, the roles are reversed which allows them to switch roles. The parameters ID, requested and rejected are all adorned with in. The rejected parameter is important as this is generated from the reject message on line 9. This means that a counter can only be created after that reject message has been sent. The final parameter result is adorned with out so this parameter must be generated by the protocol.

By looking at the way information is bound to the protocols bindings three possible complete enactments of this protocol can be observed. After an offer message has been sent, an offer can either be accepted, rejected or rejected with a counter offer being sent.

```
1 protocol Counter {
2   Roles: Debtor, Creditor
3   Public Parameters out ID key, in requested, out result
4   Private Parameters alternative
5
6   Debtor ↦ Creditor: Offer [in ID, out alternative, in
        requested]
7   Accept(Creditor, Debtor, in ID, in alternative, in
        requested, out result)
8   Creditor ↦ Debtor: Reject [in IDy, in alternative, in
        requested, out result]
9 }
```

Listing 1.2. Counter Protocol

Listing 1.2 demonstrates counter. An enactment requires two agents to fulfil the roles of debtor and creditor. Agents fill a role in a protocol from where they are placed within a protocol's interface when it is called. In Listing 1.1 the counter protocol's interface can be seen on line 10 with creditor placed first and debtor second. This allows the agent playing creditor in the negotiation protocol to switch into the debtor role within this protocol. The public parameters of key and requested are adorned with in, meaning they are already bound when a counter protocol is called. Result is adorned with out so must be generated during an enactment for the protocol to be complete.

Offer Message: The offer message on line 5 is different to the previous offer message in Listing 1.1. In this protocol, both ID and the requested parameter are adorned as in as they are already bound from the previous offer, the debtor is

seeking to create a favourable counter offer. The alternative parameter is adorned as out and is required to be generated by the debtor of the enactment.

Once alternative resource has been bound by the offer message being sent, the only parameter remaining to be bound is result. This can be bound either by the remaining accept or reject message.

Accept Protocol: Line 6 demonstrates the accept protocol in counter. Accepts interface is different in Counter than in Negotiation. The second parameter is no longer offered but the private parameter alternative. For an agent to enact the accept protocol, the parameters of ID, alternative and requested must be known to it while the agent must binds the result parameter.

Reject Message: Line 7 shows shows the reject message which has identical parameter bindings and adornments as the accept message on line 6. What changes is what information is bound to the result binding.

```
1 protocol Accept {
2   Roles: Creditor, Debtor
3   Public Parameters in ID key, in offered, in requested, out result
4   Private Parameters commitment, released, detached
5
6   Creditor ↦ Debtor: AcceptMsg[in ID, in offered, in
        requested, out commitment]
7   Creditor ↦ Debtor: release[in ID, in commitment, out
        released]
8   Debtor ↦ Creditor: expire[in ID, in released, out result]
9   Creditor ↦ Debtor: detached[in ID, in commitment, out
        detached]
10  Debtor ↦ Creditor: satisified[in ID, in detached, out
        result]
11  Debtor ↦ Creditor: cancel[in ID, in detached, out result]
12 }
```

Listing 1.3. Accept Protocol

Listing 1.3 describes the accept protocol. An enactment requires two agents to fulfil the roles of creditor and debtor. The public parameters of ID, offered and requested are adorned with in which means this information must be known for the an accept protocol to be enacted. There are two ways for the accept protocol to be called, from Listing 1.1 on line 7 and Listing 1.2 on line 6. The only public parameter that has not been bound is result, which gives the final outcome of the interaction. When result is bound, the enactment of Accept is

complete which will propagate up to the Negotiation protocol in Listing 1.1 and end the interaction between the two agents.

Accept Message: Line 6 shows an accept message. An accept message is directed from the creditor to the debtor, the message takes in the public parameters ID, offered and requested and binds the the private parameter commitment which enables the agent in the creditor role to follow this message with a detached message or release message.
Release Message: The release message can be seen on line 7 and is a message from the creditor to the debtor. The message takes in the public parameters ID, offered and requested. The agent in the role of creditor must bind information to the released private parameter. This message models an agent reneging on the agreed commitment and releases the debtor from their commitment. A release message does not finish an enactment of Accept as it does not bind the result parameter.
Expire Message: After a expire message has been sent, the private released parameter has been bound, allowing a debtor agent to send an expire message. This message can only be sent by a debtor and binds the result parameter. After a release message, there is still one active commitment. This message releases that remaining commitment, at this point both of the previously reciprocal commitments expire and no longer hold any force over the agents.
Detached Message: The detached message can be seen on line 8, which is a message directed from the creditor to the debtor. When a detached message is sent, the creditor binds information to the private parameter detached. This binding allows the consequent and cancel messages to be sent.
Satisfied Message: The satisfied message can be seen on line 9. This message can only occur after the detached message on line 8 has been sent. When this message is sent, the agent playing the debtor role binds the public parameter result.
Cancel Message: The cancel message is displayed on line 10. Like the satisified message, this can only be sent after the detached message and it also binds the private result parameter.

BSPL separates the operational structure of interactions, it does this by abstracting away the information being passed into abstract parameters. The meaning of a message is often implied by what information is bound to the parameters during the enactment of a protocol [38]. BSPL enables multiple enactments to be displayed within the same protocol, rather than having multiple UML or other traditional software engineering specifications.

Nala uses BSPL to provide message ordering and overall operational structure of the protocol. Layered on top of BSPL are the commitment operators which provide high level meaning, capturing social meaning and accountability requirements.

4 Case Study

We now demonstrate potential interactions between the agents in the context of the Coloured Trails (CT) [21] environment. In CT, agents negotiate through one-shot negotiations in a similar fashion to what was explained in Example 1 from Section 3.1, however they are now negotiating over coloured tiles. An agent can send an offer to another agent with a request to swap tiles, this offer can either be accepted, rejected or countered with a related offer.

4.1 Coloured Trails

Coloured Trails [21] is the environment used for this case study to help demonstrate how agents would negotiate with Nala. The aim of the case study is to try and practically demonstrate a variety of potential interactions between agents during a negotiation, and how they would be portrayed through Nala.

Coloured trails is played by two or more agents, on an board of coloured square tiles with dimensions N×M. For each game, a tile is selected to be the goal for the agents to reach and each agent will then be placed on a non-goal tile with a set of tiles of the same colour palate as the square tiles. A player may move to an adjacent square provided they have a chip of the same colour to use. There is two phases to the game, the negotiation phase is when players can exchange tiles to help them reach the goal. Once the negotiation is over, the movement round begins and agents attempt to get as close to the goal as possible.

Coloured trails was thought to be an appropriate test-bed to demonstrate Nala because the test bed was initially created to study agent behavior during negotiations [21] and it has been used previously to demonstrate how norms operate during negotiations [29].

4.2 Nala in Coloured Trails

Figure 3 shows a potential 4x4 instantiation of the CT testbed. There are two agents, Alex (A) and Becka (B). These agents have picked potential routes to the goal which have been outlined via the solid black lines. Alex has an inventory of five tiles, one of each colour, to get to the goal it would need an additional yellow and has a blue tile to spare. Becka also has five tiles, to get to the goal, she needs a blue tile and has a yellow and teal tile available to trade. We now run through several scenarios to help demonstrate Nala.

Offer Messages At first the negotiation life cycle is in an inactive state as described in Figure 2. Alex needs a tile and decides to send an offer message to Becka, requesting a blue tile in exchange for a yellow tile. When Alex sends an

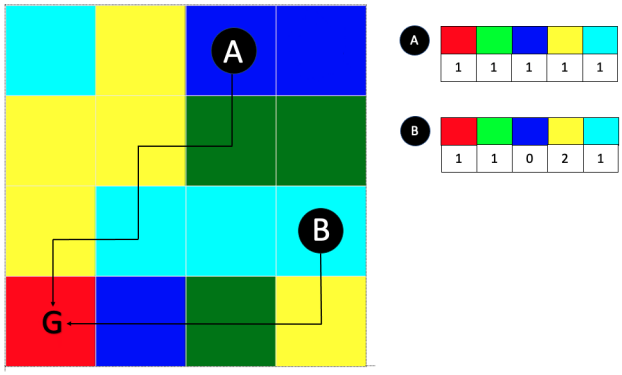

Fig. 3. A Coloured Trails Board with goal (G) and two players (A and B).

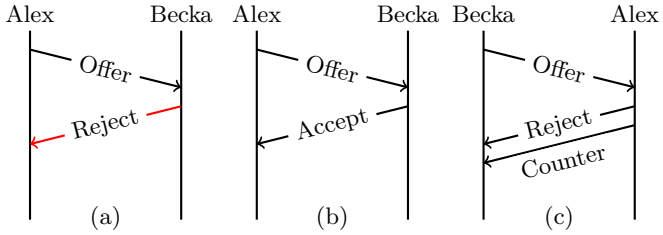

Fig. 4. Three starting enactments of a negotiation: (a) rejected offer (b) a successful negotiation (c) a counter offer.

offer message it has two effects; the negotiation transitions into an active state as Becka is now evaluating the offer Alex sent. When Alex sends the offer message, it also creates a conditional commitment C(Alex, Becka, Yellow, Blue). From this point in the interaction Becka has three potential actions which Becka can perform, as illustrated in Figure 4.

Figure 4a demonstrates an rejected offer. Alex sends an offer message and Becka decides the trade isn't helpful to her. Becka sends a reject message back to Alex, which releases Alex from the commitment C(Alex, Becka, Yellow, Blue). This negotiation enters a failed state and neither agent is committed or accountable to each other.

Figure 4b demonstrates an accepted offer. Alex creates the conditional commitment C(Alex, Becka, Yellow, Blue) towards Becka by sending an offer message. Becka sends an accept message back to Alex, creating a second conditional commitment C(Becka, Alex, Blue, Yellow) that is directed from Becka to Alex. These two commitments are reciprocal. The negotiation enters the successful state; Alex and Becka are now committed to each other to perform the trade.

Figure 4c demonstrates a counter offer. Suppose Becka decides to send an offer instead of Alex. She sends an offer to Alex, asking for a blue tile in exchange

for a cyan tile. This creates the conditional commitment C(Becka, Alex, Blue, Cyan). Alex does not need extra cyan tiles to get to the goal, so Alex sends a reject message. This releases Becka from the previous commitment that was created. Alex knows that Becka wants a blue tile, and as Alex has a blue tile available he decides to send a counter offer. Alex asks Becka for a yellow tile in exchange of a blue tile. This counter offer rejects and overrides the previous offer, instantiating the commitment C(Alex, Becka, Yellow, Blue). Becka can either reject this offer, releasing Alex from his commitment or Becka can accept the offer as illustrated in Figure 4b, which forms a reciprocal commitment.

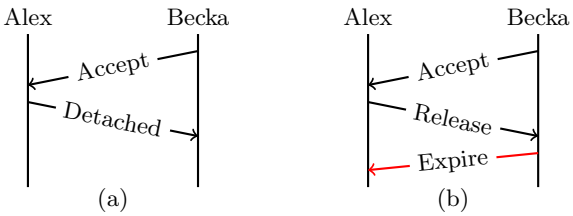

Fig. 5. Enactments after an accept message (a) detached message (b) release message

When an offer or counter offer has been accepted, a pair of reciprocal commitment exists between agents. In Figure 5a, Alex sends a detached message after receiving the accept message. Informing Becka that Alex has transferred the blue tile. This affects the reciprocal commitments differently. As the consequent of C(Alex, Becka, Yellow, Blue) is now true, it is considered satisfied and no longer holds any force over Alex. C(Becka, Alex, Blue, Yellow) moves into a detached state as its antecedent is now true, Becka is now liable to Alex to carry out the trade of a yellow tile.

Figure 5b demonstrates a possible terminal state for the negotiation. After Becka sends the accept message, Alex sends a release message. After which Becka sends an expire message. The combination of a release and expire messages releases both agents from their reciprocal commitments that are currently in force, this causes both commitments to expire. Despite the negotiation being in a successful state as they agreed upon a trade, the agents did not trade any tiles.

Figure 6 follows from Figure 5a. At this point there is only one commitment in force, C(Becka, Alex, Blue, Yellow). This commitment is currently in a detached state. In Figure 6a, Becka sends a satisified message to Alex, informing Alex that a yellow tile has been transferred. Becka satisfies C(Becka, Alex, Blue, Yellow), releasing Becka from the commitment. Both reciprocal commitments that were created during this negotiation have been satisfied as everyone fulfilled their commitments, no one is accountable to perform any more actions.

Figure 6b demonstrates a cancel message. By sending this Becka informs Alex that she is no longer willing or able to send a yellow tile back. As the consequent proposition of C(Becka, Alex, Blue, Yellow) is now false, the commitment

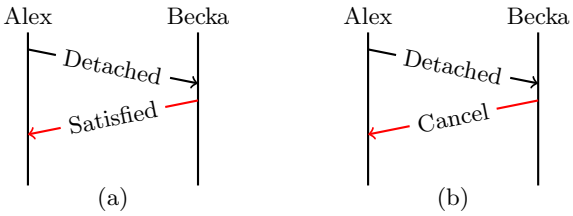

Fig. 6. Enactments after an detached message (a) satisified message (b) cancel message

transitions into the violated state of the commitment life cycle. When looking at the reciprocal commitments C(Alex, Becka, Yellow, Blue) is in the satisfied state and C(Becka, Alex, Blue, Yellow) is in the violated state. By observing the direction of the commitments, we can see that Becka is liable for the violation. By doing this Becka becomes accountable to Alex.

5 Conclusion

Existing negotiation protocols use traditional interaction protocols, which do not capture meaning of interactions or establish accountability. To address the gap, we formalise Nala, a commitment-based negotiation protocol. Nala uses commitments to provide meaning to the agent interactions through the directionality of the commitments created between agents. Accountability requirements are also established through violated commitments. Via a case study, we demonstrate how Nala could be operationalised. The case study illustrates the various paths through a negotiation, explaining the potential actions and the meaning of these actions.

Nala could be further developed in several areas. Negotiations often have an associated time limit or expiration. Previous works [32] have added a temporal component to the commitments antecedent or consequent. These time constraints could alter the behavior of agents, for example causing them to accept offers that they originally did not look favourably on.

Although Nala helps establish accountability requirements, it does not provide a mechanism to represent accountability. If a negotiation goes into a failed or expired state is it possible to assignment accountability in these situations? For example, if an agent wants to accept the current terms of a negotiation, so sends an accept message to the other agent. What happens if this agent is not able to send this message, or it is undelivered? It might cause the negotiation to expire when it should have been accepted. Who should be accountable in this situation? Recent research has shown how this could be implemented [6] which would be an interesting addition allowing Nala to become more robust and transparent.

References

1. Aydoğan, R., Festen, D., Hindriks, K.V., Jonker, C.M.: Alternating offers protocols for multilateral negotiation. In: Fujita, K., Bai, Q., Ito, T., Zhang, M., Ren, F., Aydoğan, R., Hadfi, R. (eds.) Modern Approaches to Agent-based Complex Automated Negotiation, pp. 153–167. Springer International Publishing, Cham (2017). https://doi.org/10.1007/978-3-319-51563-2_10
2. Aydoğan, R., Kafalı, Ö., Arslan, F., Jonker, C.M., Singh, M.P.: NOVA: Value-based negotiation of norms. ACM Transactions on Intelligent Systems and Technology (TIST) **12**(4), 45:1–45:24 (Aug 2021)
3. Baldoni, M., Baroglio, C., Capuzzimati, F., Micalizio, R.: Commitment-based agent interaction in JaCaMo+. Fundamenta Informaticae **159**, 1–33 (03 2018). https://doi.org/10.3233/FI-2018-1656
4. Baldoni, M., Baroglio, C., Marengo, E., Patti, V., Capuzzimati, F.: Engineering commitment-based business protocols with the 2CL methodology. Autonomous Agents and Multi-Agent Systems **28**(4), 519–557 (Jul 2014)
5. Baldoni, M., Baroglio, C., May, K.M., Micalizio, R., Tedeschi, S.: ADOPT JaCaMo: Accountability-Driven Organization Programming Technique for JaCaMo. In: An, B., Bazzan, A., Leite, J., Villata, S., van der Torre, L. (eds.) PRIMA 2017: Principles and Practice of Multi-Agent Systems. pp. 295–312. Springer International Publishing, Cham (2017)
6. Baldoni, M., Baroglio, C., Micalizio, R., Tedeschi, S.: Robustness based on accountability in multiagent organizations. In: Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems. p. 142–150. AAMAS '21, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC (2021)
7. Beer, M., D'inverno, M., Luck, M., Jennings, N., Preist, C., Schroeder, M.: Negotiation in multi-agent systems. Knowledge Engineering Review **14**(3), 285–289 (Sep 1999). https://doi.org/10.1017/S0269888999003021
8. Čapkovič, F., Jotsov, V.: A System Approach to Agent Negotiation and Learning, pp. 133–160. Springer Berlin Heidelberg, Berlin, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13428-9_6, https://doi.org/10.1007/978-3-642-13428-9_6
9. Chopra, A.K., Christie V., S.H., Singh, M.P.: Splee: A declarative information-based language for multiagent interaction protocols. In: Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems. pp. 1054–1063. International Foundation for Autonomous Agents and Multiagent Systems, São Paulo (2017)
10. Chopra, A.K., Singh, M.P.: Agent communication. In: Weiss, G. (ed.) Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence, pp. 99–210. MIT Press, Cambridge (1999). https://doi.org/10.1017/CBO9780511615337.008
11. Chopra, A.K., Singh, M.P.: Cupid: Commitments in relational algebra. In: Proceedings of the 29th AAAI Conference on Artificial Intelligence. pp. 2052–2059. AAAI Press, Austin, Texas (2015)
12. Chopra, A.K., Singh, M.P.: Generalized commitment alignment. In: Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems. pp. 453–461. International Foundation for Autonomous Agents and Multiagent Systems, Istanbul (2015)
13. Chopra, A.K., Singh, M.P.: Custard: Computing norm states over information stores. In: Proceedings of the 15th International Conference on Autonomous Agents; Multiagent Systems. pp. 1096–1105. International Foundation for Autonomous Agents and Multiagent Systems, Singapore (2016)

14. Chopra, A.K., Singh, M.P.: From social machines to social protocols: Software engineering foundations for sociotechnical systems. In: Proceedings of the 25th International Conference on World Wide Web. pp. 903–914. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE (2016). https://doi.org/10.1145/2872427.2883018
15. Chopra, A.K., Singh, M.P.: Accountability as a foundation for requirements in sociotechnical systems. IEEE Internet Computing **25**(6), 33–41 (2021). https://doi.org/10.1109/MIC.2021.3106835
16. Christie, S.H., Chopra, A.K., Singh, M.P.: Refinement for multiagent protocols. In: Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems. pp. 258–266. International Foundation for Autonomous Agents and Multiagent Systems, Auckland (2020)
17. Colombetti, M.: A commitment-based approach to agent speech acts and conversations. In: Workshop on Agent Languages and Communication Policies. pp. 21–29. Association for Computing Machinery, Barcelona (2000)
18. Criado, N., Argente, E., Noriega, P., Botti, V.: Manea: A distributed architecture for enforcing norms in open mas. Eng. Appl. Artif. Intell. **26**(1), 76–95 (Jan 2013). https://doi.org/10.1016/j.engappai.2012.08.007
19. Flores, R.A., Pasquier, P., Chaib-Draa, B.: Conversational semantics sustained by commitments. Autonomous Agents and Multi-Agent Systems **14**(2), 165–186 (Apr 2007). https://doi.org/10.1007/s10458-006-0011-1
20. Gonçalves, E.J.T., Cortés, M.I., Campos, G.A.L., Lopes, Y.S., Freire, E.S., da Silva, V.T., de Oliveira, K.S.F., de Oliveira, M.A.: Mas-ml 2.0: Supporting the modelling of multi-agent systems with different agent architectures. Journal of Systems and Software **108**, 77–109 (2015). https://doi.org/https://doi.org/10.1016/j.jss.2015.06.008
21. Grosz, B.J., Kraus, S., Talman, S., Stossel, B., Havlin, M.: The influence of social dependencies on decision-making: Initial investigations with a new game. In: International Joint Conference on Autonomous Agents and Multiagent Systems. pp. 782–789. IEEE, New York (2004)
22. Günay, A., Chopra, A.K.: Stellar: A programming model for developing protocol-compliant agents. In: Weyns, D., Mascardi, V., Ricci, A. (eds.) Engineering Multi-Agent Systems. pp. 117–136. Springer International Publishing, Cham (2019)
23. Hemaissia, M., El Fallah Seghrouchni, A., Labreuche, C., Mattioli, J.: A multilateral multi-issue negotiation protocol. In: Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems. Association for Computing Machinery, New York (2007). https://doi.org/10.1145/1329125.1329314
24. High-Level Expert Group on AI: Ethics guidelines for trustworthy ai. Report, European Commission, Brussels (Apr 2019)
25. Hudaib, A., Qasem, M.H., Obeid, N.: Fipa-based semi-centralized protocol for negotiation. In: Silhavy, R., Silhavy, P., Prokopova, Z. (eds.) Cybernetics Approaches in Intelligent Systems. pp. 135–149. Springer International Publishing, Cham (2018)
26. for Intelligent Physical Agents, F.: Fipa contract net interaction protocol specification (2002), [Accessed on 02-09-2021]
27. Ji, S., Tian, Q., Liang, Y.: A Petri-Net-Based Modeling Framework for Automated Negotiation Protocols in Electronic Commerce, pp. 324–336. Springer-Verlag, Berlin, Heidelberg (2009)
28. Kafalı, Ö., Ajmeri, N., Singh, M.P.: Specification of sociotechnical systems via patterns of regulation and control. ACM Transactions on Software Engineering

and Methodology (TOSEM) **29**(1), 7:1–7:50 (Feb 2020). https://doi.org/10.1145/3365664
29. Kalia, A.K., Ajmeri, N., Chan, K.S., Cho, J.H., Adalı, S., Singh, M.P.: The interplay of emotions and norms in multiagent systems. In: Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI). pp. 17–23. IJCAI, Macao (2019)
30. Lopes, F., Coelho, H.: Negotiation and Argumentation in Multi-Agent Systems: Fundamentals, Theories, Systems and Applications. Bentham Science Publishers, Sharjah, U.A.E (12 2014). https://doi.org/10.2174/9781608058242114010l
31. Mallya, A.U., Yolum, P., Singh, M.P.: Resolving commitments among autonomous agents. In: Dignum, F. (ed.) Advances in Agent Communication. pp. 166–182. Springer Berlin Heidelberg, Berlin, Heidelberg (2004)
32. Marengo, E., Baldoni, M., Baroglio, C., Chopra, A.K., Patti, V., Singh, M.P.: Commitments with regulations: Reasoning about safety and control in regula. In: The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 2. pp. 467–474. International Foundation for Autonomous Agents and Multiagent Systems, Taipei (2011)
33. Morris-Martin, A., De Vos, M., Padget, J.: Norm emergence in multiagent systems: A viewpoint paper. In: Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems. pp. 2152–2154. International Foundation for Autonomous Agents and Multiagent Systems, Auckland (2020)
34. Palanca, J., Terrasa, A., Carrascosa, C., Julián, V.: Simfleet: A new transport fleet simulator based on mas. In: De La Prieta, F., González-Briones, A., Pawleski, P., Calvaresi, D., Del Val, E., Lopes, F., Julian, V., Osaba, E., Sánchez-Iborra, R. (eds.) Highlights of Practical Applications of Survivable Agents and Multi-Agent Systems. The PAAMS Collection. pp. 257–264. Springer International Publishing, Cham (2019)
35. Pitt, J., Kamara, L., Artikis, A.: Interaction patterns and observable commitments in a multi-agent trading scenario. In: Proceedings of the Fifth International Conference on Autonomous Agents. pp. 481–488. Association for Computing Machinery, New York (2001). https://doi.org/10.1145/375735.376422
36. Ren, Z., Anumba, C.J., Ugwu, O.O.: The development of a multi-agent system for construction claims negotiation. Advanced Engineering Software **34**(11–12), 683–696 (Nov 2003). https://doi.org/10.1016/S0965-9978(03)00107-8
37. Singh, M.P.: A social semantics for agent communication languages. In: Issues in Agent Communication. pp. 31–45. Springer-Verlag, Berlin, Heidelberg (2000)
38. Singh, M.P.: Information-driven interaction-oriented programming: Bspl, the blindingly simple protocol language. In: Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 2. pp. 491–498. International Foundation for Autonomous Agents and Multiagent Systems, Taipei (2011)
39. Singh, M.P.: Norms as a basis for governing sociotechnical systems. ACM Transactions on Intelligent Systems and Technology (TIST) **5**(1), 21:1–21:23 (Dec 2013)
40. Smith, R.G.: The contract net protocol: High-level communication and control in a distributed problem solver. IEEE Trans. Comput. **29**(12), 1104–1113 (Dec 1980). https://doi.org/10.1109/TC.1980.1675516
41. Tesfay, T., El Haouzi, H., Demesure, G., Pannequin, R., Thomas, A.: Multi-agent systems negotiation to deal with dynamic scheduling in disturbed industrial context. Journal of Intelligent Manufacturing **31** (08 2020). https://doi.org/10.1007/s10845-019-01515-7

42. Wang, G., Wong, T.N., Wang, X.H.: A negotiation protocol to support agent argumentation and ontology interoperability in mas-based virtual enterprises. In: 2010 7th International Conference on Information Technology: New Generations. pp. 448–453. IEEE, USA (2010). https://doi.org/10.1109/ITNG.2010.39
43. Weiss, G.: Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. MIT Press, Cambridge (1999)
44. Yolum, P., Singh, M.P.: Flexible protocol specification and execution: Applying event calculus planning using commitments. In: Proceedings of the 1st International Joint Conference on Autonomous Agents and Multiagent Systems: Part 2. pp. 527–534. Association for Computing Machinery, New York (2002). https://doi.org/10.1145/544862.544867