Temporal Linear Logic as a Basis for Flexible Agent Interactions

Duc Q. Pham, James Harland
School of CS&IT
RMIT University
GPO Box 2476V
Melbourne, 3001, Australia
{qupham,jah}@cs.rmit.edu.au

ABSTRACT

Interactions between agents in an open system such as the Internet require a significant degree of flexibility. A crucial aspect of the development of such methods is the notion of commitments, which provides a mechanism for coordinating interactive behaviors among agents. In this paper, we investigate an approach to model commitments with tight integration with protocol actions. This means that there is no need to have an explicit mapping from protocols actions to operations on commitments and an external mechanism to process and enforce commitments. We show how agents can reason about commitments and protocol actions to achieve the end results of protocols using a reasoning system based on temporal linear logic, which incorporates both temporal and resource-sensitive reasoning. We also discuss the application of this framework to scenarios such as online commerce.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Intelligent Agents; D.3.2 [**Programming Languages**]: Language Classifications

General Terms

Theory, Design

Keywords

Agent communication languages and protocols, logic and formal models of agency and multi-agent systems

1. INTRODUCTION AND MOTIVATION

Recently, software development has evolved toward the development of intelligent, interconnected systems working in a distributed manner. The agent paradigm has become well suited as a design metaphor to deal with complex systems comprising many components each having their own thread of control and purposes and involved in dynamic and complex interactions.

In multi-agent environments, agents often need to interact with each other to fulfill their goals. Protocols are used to regulate interactions. In traditional approaches to protocol specification, like those using Finite State Machines or Petri Nets, protocols are often predetermined legal sequences of interactive behaviors. In frequently changing environments such as the Internet, such fixed sequences can quickly become outdated and are prone to failure. Therefore, agents are required to adapt their interactive behaviors to succeed and interactions among agents should not be constructed rigidly.

To achieve flexibility, as characterized by Yolum and Singh in [11], interaction protocols should ensure that agents have autonomy over their interactive behaviors, and be free from any unnecessary constraints. Also, agents should be allowed to adjust their interactive actions to take advantages of opportunities or handle exceptions that arise during interaction.

For example, consider the scenario below for online sales. *A merchant Mer has 200 cricket bats available for sale with a unit price of 10 dollars. A customer Cus has $50. Cus has a goal of obtaining from Mer a cricket bat at some time. There are two options for Cus to pay. If Cus uses credit payment, Mer needs a bank Ebank to check Cus's credit. If Cus's credit is approved, Ebank will arrange the credit payment. Otherwise, Cus may then take the option to pay via PayPal. The interaction ends when goods are delivered and payment is arranged.*

A flexible approach to this example should include several features. Firstly, the payment method used by Cus should be at Cus's own choice and have the property that if Cus's credit check results in a disapproval, this exception should also be handled automatically by Cus's switching to PayPal. Secondly, there should be no unnecessary constraint on the order in which actions are performed, such as which of making payments and sending the cricket bat should come first. Thirdly, choosing a sequence of interactive actions should be based on reasoning about the intrinsic meanings of protocol actions, which are based on the notion of *commitment*, i.e. which refers to a strong promise to other agent(s) to undertake some courses of action.

Current approaches [11, 12, 10, 1] to achieve flexibilities using the notion of commitment make use of an abstract layer of commitments. However, in these approaches, a mapping from protocol actions onto operations on commitments

AAMAS'07 May 14–18 2007, Honolulu, Hawai'i, USA.

978-81-904262-7-5 (RPS)

as well as handling and enforcement mechanisms of commitments must be externally provided. Execution of protocol actions also requires concurrent execution of operations on related commitments. As a result, the overhead of processing the commitment layer makes specification and execution of protocols more complicated and error prone. There is also a lack of a logic to naturally express aspects of resources, internal and external choices as well as time of protocols.

Rather than creating another layer of commitment outside protocol actions, we try to achieve a modeling of commitments that is integrated with protocol actions. Both commitments and protocol actions can then be reasoned about in one consistent system. In order to achieve that, we specify protocols in a declarative manner, i.e. what is to be achieved rather then how agents should interact. A key to this is using logic. Temporal logic, in particular, is suitable for describing and reasoning about temporal constraints while linear logic [3] is quite suitable for modeling resources. We suggest using a combination of linear logic and temporal logic to construct a commitment based interaction framework which allows both temporal and resource-related reasoning for interaction protocols. This provides a natural manipulation and reasoning mechanism as well as internal enforcement mechanisms for commitments based on proof search.

This paper is organized as follows. Section 2 discusses the background material of linear logic, temporal linear logic and commitments. Section 3 introduces our modeling framework and specification of protocols. Section 4 discusses how our framework can be used for an example of online sale interactions between a merchant, a bank and a customer. We then discuss the advantages and limitations of using our framework to model interaction protocols and achieve flexibility in Section 5. Section 6 presents our conclusions and items of further work.

2. BACKGROUND

In order to increase the agents' autonomy over their interactive behaviors, protocols should be specified in terms of what is to be achieved rather than how the agents should act. In other words, protocols should be specified in a declarative manner. Using logic is central to this specification process.

2.1 Linear Logic

Logic has been used as formalism to model and reason about agent systems. Linear logic [3] is well-known for modeling resources as well as updating processes. It has been considered in agent systems to support agent negotiation and planning by means of proof search [5, 8].

In real life, resources are consumed and new resources are created. In such logic as classical or temporal logic, however, a direct mapping of resources onto formulas is troublesome. If we model resources like A as "one dollar" and B as "a chocolate bar", then $A \Rightarrow B$ in classical logic is read as "from one dollar we can get a chocolate bar". This causes problems as the implication allows to get a chocolate bar (B is true) while retaining one dollar (A remains true).

In order to resolve such resource - formula mapping issues, Girard proposed the constraints on which formulas will be used exactly once and can no longer be freely added or removed in derivations and hence treating linear logic formulas as resources. In linear logic, a linear implication $A \multimap B$, however, allows A to be removed after deriving B, which means the dollar is gone after using one dollar to buy a chocolate bar.

Classical conjunction (and) and disjunction (or) are recast over different uses of contexts - multiplicative as combining and additive as sharing to come up with four connectives. A $\otimes$ (*multiplicative conjunction*) A, means that one has two As at the same time, which is different from $A \wedge A = A$. Hence, $\otimes$ allows a natural expression of proportion. A $\wp$ (*multiplicative disjunction*) B, means that if not A then B or vice versa but not both A and B.

The ability to specify choices via the additive connectives is a particularly useful feature of linear logic. A & (*additive conjunction*) B, stands for one own choice, either of A or B but not both. A $\oplus$ (*additive disjunction*) B, stands for the possibility of either A or B, but we don't know which. As remarked in [5], & and $\oplus$ allow choices to be made clear between internal choices (one's own), and external choices (others' choice). For instance, to specify that the choice of places A or B for goods' delivery is ours as the supplier, we use $A\&B$, or is the client's, we use $A \oplus B$.

In agent systems, this duality between inner and outer choices is manifested by one agent having the power to choose between alternatives and the other having to react to whatever choice is made.

Moreover, during interaction, the ability to match consumption and supply of resources among agents can simplify the specification of resource allocations. Linear logic is a natural mechanism to provide this ability [5]. In addition, it is emphasized in [8] that linear logic is used to model agent states as sets of consumable resources and particularly, linear implication is used to model transitions among states and capabilities of agents.

2.2 Temporal Linear Logic

While linear logic provides advantages to modeling and reasoning about resources, it does not deal naturally with time constraints. Temporal logic, on the other hand, is a formal system which addresses the description and reasoning about the changes of truth values of logic expressions over time [2]. Temporal logic can be used for specification and verification of concurrent and reactive programs [2].

Temporal Linear Logic (**TLL**) [6] is the result of introducing temporal logic into linear logic and hence is resource-conscious as well as deals with time. The temporal operators used are $\bigcirc$ (next), $\Box$ (anytime), and $\Diamond$ (sometime) [6]. Formulas with no temporal operators can be considered as being available only at present. Adding $\bigcirc$ to a formula A, i.e. $\bigcirc$A, means that A can be used only at the next time and exactly once. Similarly, $\Box$A means that A can be used at any time and exactly once. $\Diamond A$ means that A can be used once at some time.

Though both $\Box$ and $\Diamond$ refer to a point in time, the choice of which time is different. Regarding $\Box$, the choice is an internal choice, as appropriate to one's own capability. With $\Diamond$, the choice is externally decided by others.

2.3 Commitment

The concept of social commitment has been recognized as fundamental to agent interaction. Indeed, social commitment provides intrinsic meanings of protocol actions and states [11]. In particular, persistence in commitments introduces into agents' consideration a certain level of predictability of other agents' actions, which is important when agents deal with issues of inter-dependencies, global constraints or

resources sharing [7].

Commitment based approaches associate protocols actions with operations on commitments and protocol states with the set of effective commitments [11]. Completing the protocol is done via means-end reasoning on commitment operations to bring the current state to final states where all commitments are resolved. From then, the corresponding legal sequences of interactive actions are determined. Hence, the approaches systematically enhance a variety of legal computations [11].

Commitments can be reduced to a more fundamental form known as pre-commitments. A *pre-commitment* here refers to a potential commitment that specifies what the owner agent is willing to commit [4], like performing some actions or achieving a particular state. Agents can negotiate about pre-commitments by sending proposals of them to others. The others can respond by agreeing or disagreeing with the proposal or proposing another pre-commitment. Once a pre-commitment is agreed, it then becomes a commitment and the process moves from negotiation phase to commitment phase, in which the agents act to fulfill their commitments.

3. MODELING AGENT INTERACTIONS

Protocols are normally viewed external to agents and are essentially a set of commitments externally imposed on participating agents. We take an internal view of protocols, i.e. from the view of participating agents by putting the specification of commitments locally at the respective agents according to their roles.

Such an approach enables agents to manage their own protocol commitments. Indeed, agents no longer accept and follow a given set of commitments but can reason about which commitments of theirs to offer and which commitments of others to take, while considering the current needs and the environment. Protocols arise as commitments are then linked together via agents' reasoning based on proof search during the interaction. Also, ongoing changes in the environment are taken as input into the generation of protocols by agent reasoning. This is the reverse of other approaches which try to make the specification flexible to accommodate changes in the environment. Hence, it is a step closer to enabling emergent protocols, which makes protocols more dynamic and flexible to the context.

In a nutshell, services are what agents are capable of providing to other agents. Commitments can then be seen to arise from combinations of services, i.e. an agent's capabilities. Hence, our approach shifts specifying a set of protocol commitments to specifying sets of pre-commitments as capabilities for each agent. Commitments are then can be reasoned about and manipulated by the same logic mechanism as is used for the agents' actions, resources and goals, which greatly simplifies the system.

Our framework uses TLL as a means of specifying interaction protocols. We encode various concepts such as resource, capability and commitment in TLL. The symmetry between a formula and its negation in TLL is explored as a way to model resources and commitments. We then discuss the central role of pre-commitments, and how they are specified at each participating agent. It then remains for agents to reason about pre-commitments to form protocol commitments, which are subsequently discharged.

3.1 Modeling resources and capabilities

A unit of consumable resources is modeled as a proposition in linear logic. Numeric figures can be used to abbreviate a multiplicative conjunction of the same instances. For example, 2 $dollar = dollar \otimes dollar$. Moreover, such $\bigcirc^3 A$ is a shorthand for $\bigcirc \bigcirc \bigcirc A$.

In order to address the dynamic manipulation of resources, we also include information about the *location* and *ownership* in the encoding of resources to address the relocation and changes in possession of resources during agent interaction. That resource A is located at agent α and owned by agent β is expressed via a shorthand notation as $A@\alpha_\beta$, which is treated as a logic proposition in our framework. This notation can be later extended to a more complex logic construct to reason about changes in location and ownership.

In our running example, a cricket bat *cricket_b* being located at and owned by agent Mer is denoted as $cricket\_b@M_M$. After a successful sale to the agent customer Cus, the cricket bat will be relocated to and owned by agent Cus. The formula $cricket\_b@C_C$ will replace the formula $cricket\_b@M_M$ to reflect the changes.

Our treatment of *unlimited resources* is to model it as a number σ of copies of the resource's formula such that the number σ is chosen to be extremely large, relative to the context. For instance, to indicate that the merchant Mer can issue an unlimited number of sale quotes at any time, we use $\sigma \square\, sale\_quote@M_M$.

Declaration of *actions* is also modeled in a similar manner as of resources.

The *capabilities* of agents refer to producing, consuming, relocating and changing ownership of resources. Capabilities are represented by describing the state before and after performing them. The general representation form is $\Gamma \multimap \Delta$, in which Γ describes the conditions before and Δ describes the conditions after. The linear implication $\multimap$ in linear logic indeed ensures that the conditions before will be transformed into the conditions after. Moreover, some capabilities can be applied at any number of times in the interaction context and their formulas are also preceded by the number σ.

To take an example, we consider the capability of agent Mer of selling a cricket bat for 10 dollars. The conditions before are 10 dollars and a payment method from agent Cus: $10\$@C_C \otimes pay\_m@C_C$. Given these, by applying the capability, Mer will gain 10 dollars ($10\$@M_M$) and commit to providing a cricket bat ($cricket\_b@M_M^\perp$) so that Cus will get a cricket bat ($cricket\_b@C_C$). Together, the capability is encoded as $10\$@C_C \otimes pay\_m@C_C \multimap 10\$@M_M \otimes cricket\_b@M_M^\perp \otimes cricket\_b@C_C$.

3.2 Modeling commitments

We discuss the modeling of various types of commitments, their fulfillments and enforcement mechanisms.

Due to duality in linear logic, positive formulas can be regarded as formulas in supply and negative formulas can be regarded as formulas in demand. Hence, we take an approach of modeling non-conditional or *base commitments* as negative formulas. In particular, by turning a formula into its negative form, a base commitment to derive the resources or carry out the actions associated with the formula is created. In the above example, a commitment of agent Mer to provide a cricket bat ($cricket\_b@M_M$) is $cricket\_b@M_M^\perp$.

A base commitment is *fulfilled* (discharged) whenever the committing agent successfully brings about the respective

resources or carries out the actions as required by the commitment. In TLL modeling, this means that the corresponding positive formula is derived. Resolution of commitments can then be naturally carried out by inference in TLL. For example, $cricket\_b@M_M$ will fulfil the commitment $cricket\_b@M_M^\perp$ and both formulas are automatically removed as $cricket\_b@M_M \otimes cricket\_b@M_M^\perp \vdash \perp$.

Under a further assumption that agents are expected to resolve all formulas in demand (removing negative formulas), this creates a driving pressure on agents to resolve base commitments. This pressure then becomes a natural and internal enforcement mechanism for base commitments.

A commitment with conditions (or *conditional commitment*) can be modeled by connecting the conditions to base commitments via a linear implication. A general form is $\Gamma \multimap \Delta$ where Γ is the condition part and Δ includes base commitments. If the condition Γ is derived, by consuming Γ, the linear implication will ensure that Δ results, which means the base commitments in Δ become effective. If the conditions can not be achieved, the linear implication can not be applied and hence commitment part in the conditional commitment is still inactive.

In our approach, conditional commitments are specified in their potential form as pre-commitments of participating agents. Pre-commitments are negotiated among agents via proposals and upon being accepted, will form conditional commitments among the engaged agents. Conditional commitments are interpreted as that the condition Γ is required of the proposed agent and the commitment part Δ is the responsibility of the owner (proposing) agent. Indeed, such interpretation and the encoding of $\multimap$ realize the notion of a conditional commitment that owner agent is willing to commit to deriving Δ given the proposed agent satisfies the conditions Γ.

Conditional commitments, pre-commitments and capabilities all have similar encodings. However, their differences lie in the phases of commitment that they are in. Capabilities are used internally by the owner agent and do not involve any commitment. Pre-commitments can be regarded as capabilities intended for forming conditional commitments. Upon being accepted, pre-commitments will turn into conditional commitments and bring the two engaged agents into a commitment phase. As an example, consider that Mer has a capability of selling cricket bats: $(10\$@C_C \otimes pay\_m@C_C) \multimap (10\$@M_M \otimes cricket\_b@M_M^\perp \otimes cricket\_b@C_C)$. When Mer proposes its capability to Cus, the capability acts as a pre-commitment. When the proposal gets accepted, that pre-commitment will turn into a conditional commitment in which Mer commits to fulfilling the base commitment $cricket\_b@M_M^\perp$ (which leads to having $cricket\_b@C_C$) upon the condition that Cus derives $10\$@C_C \otimes pay\_m@C_C$ (which leads to having $10\$@M_M$).

Breakable commitments which are in place to provide agents with the desired flexibility to remove itself from its commitments (cancel commitments) are also modeled naturally in our framework. A base commitment $Com^\perp$ is turned into a breakable base commitment $(cond \oplus Com)^\perp$. The extra token *cond* reflects the agent's internal deliberation about when the commitment to derive Com is broken. Once *cond* is produced, due to the logic deduction $cond \otimes (cond \oplus Com)^\perp \vdash \perp$, the commitment $(cond \oplus Com)^\perp$ is removed, and hence breaking the commitment of deriving Com. Moreover, a breakable conditional commitment is modeled as $A \multimap (1\&B)$, instead of $A \multimap B$. When the condition A is provided, the linear implication brings about $(1\&B)$ and it is now up to the owner agent's internal choice whether 1 or B is resulted. If the agent chooses 1, which practically means nothing is derived, then the conditional commitment is deliberately broken.

3.3 Protocol Construction

Given the modeling of various interaction concepts like resource, action, capability, and commitment, we will discuss how protocols can be specified.

In our framework, each agent is encoded with the resources, actions, capabilities, pre-commitments, any pending commitments that it has. Pre-commitments, which stem from services the agents are capable of providing, are designated to be fair exchanges. In a pre-commitment, all the requirements of the other party are put in the condition part and all the effects to be provided by the owner agent are put on the commitment part to make up a trade-off. Such a design allows agents to freely propose pre-commitments to any interested parties.

An example of pre-commitments is that of an agent Merchant regarding a sale of a cricket bat: $[10\$@C_C \otimes pay\_m@C_C \multimap 10\Box\$@M_M \otimes \Box cricket\_b@C_C \otimes cricket\_b@M_M^\perp]$. The condition is the requirement that the customer agent provides 10 dollars, which is assumed to be the price of a cricket bat, via a payment method. The exchange is the cricket bat for the customer ($\Box cricket\_b@C_C$) and hence is fair to the merchant.

Protocols are specified in terms of sets of pre-commitments at participating agents. Given some initial interaction commitments, a protocol emerges as agents are reasoning about which pre-commitments to offer and accept in order to fulfill these commitments.

Given a such a protocol specification, we then discuss how interaction might take place. An interaction can start with a request or a proposal. When an agent can not achieve some commitments by itself, it can make a request of them or propose a relevant pre-commitment to an appropriate agent to fulfill them. The choice of which pre-commitments depends on if such pre-commitments can produce the formulas to fulfill the agent's pending commitments.

When an agent receives a request, it searches for pre-commitments that can together produce the required formulas of the requests. Those pre-commitments found will be used as proposals to the requesting agents. Otherwise, a failure notice will be returned.

When a proposal is received, the recipient agent also performs a search with the inclusion of the proposal for a proof of those formulas that can resolve its commitments. If the search result is positive, the proposal is accepted and becomes a commitment. The recipient then attempts to fulfill conditions of the commitments. Otherwise, the proposal is refused and no commitment is formed.

Throughout the interaction, proof search has played a vital role in protocol construction. Proof search reveals that some commitments can not be resolved locally or some pre-commitments can be used to resolve pending commitments, which prompts the agent to make a request or proposal respectively. Proof search also determines which pre-commitments are relevant to fulfillment of a request, which helps agents to decide which pre-commitments to propose to answer the request. Moreover, whether a received proposal

is relevant to any pending commitments or not is also determined by a search for proof of these commitments with an inclusion of the proposal. Conditions of proposals can be resolved by proof search as it links them with the agents' current resources and capabilities as well as any relevant pre-commitments. Therefore, it can be seen that proof search performed by participating agents can link up their respective pre-commitments and turn them into commitments as appropriate, which give rise to protocol formation. We will demonstrate this via our running example in section 4.

3.4 Interactive Messages

Agents interact by sending messages. We address agent interaction in a simple model which contains messages of type requests, proposals, acceptance, refusal and failure notice.

We denote "Source to Destination:" prior to each message to indicate the source and destination of the message. For example "Cust to Mer:" denotes that the message is sent from agent Cust to agent Mer.

Request messages start with the key word REQUEST: "REQUEST + $formula$". Formulas in request messages are of commitments.

Proposal messages are preceded with "PROPOSE". Formulas are of capabilities. For example, α to β: "PROPOSE $\Gamma \multimap \Delta$" is a proposal from agent α to agent β.

There are messages that agents use to response to a proposal. Agents can indicate an acceptance: "ACCEPT", or a refusal: "REFUSE". To notice a failure in fulfilling a request or proposal, agents reply with that request or proposal message appended with "FAIL".

3.5 Generating Interactions

As we have seen, temporal linear logic provides an elegant means for encoding the various concepts of agent interaction in a commitment based specification framework. Appropriate interaction is generated as agents negotiate their specified pre-commitments to fulfill their goals. The association among pre-commitments at participating agents and the monitoring of commitments to ensure that all are discharged are performed by proof search. In the next section, we will demonstrate how specification and generation of interactions in our framework might work.

4. EXAMPLE

We return to the online sales scenario introduced in Section 1.

4.1 Specifying Protocol

We design a set of pre-commitments and capabilities to implement the above scenario. For simplicity, we refer to them as rules.

Rules at agent Mer

Mer has available at any time 200 cricket bats for sale and can issue sale quotes at any time:
$200 \Box cricket\_b@M_M \otimes \sigma \Box sale\_quote@M_M$.

Rule 1: Mer commits to offering a cricket bat ($cricket\_b@M_M^{\perp}$) to Cus ($\Box cricket\_b@C_C$) if Cus pays 10 dollars ($10\$@C_C$) either via Paypal or credit card. The choice is at Cus.
$\sigma \Box [10\$@C_C \otimes (Paypal\_paid@M_M \oplus credit\_paid@M_M)$
$\multimap 10 \Box \$@M_M \otimes \Box cricket\_b@C_C \otimes cricket\_b@M_M^{\perp}]$

Rule 2: If EBank carries out the credit payment to Mer then the requirement of credit payment at Mer is fulfilled:
$\sigma \Box [credit\_paym@M_B \multimap \Box credit\_paid@M_M]$

Rule 3: If Ebank informs Mer of its disapproval of Cus's credit then Mer will also let Cus know.
$\sigma \Box [credit\_not\_appr@M_B \multimap credit\_not\_appr@C_B]$

Rules at agent Ebank

Rule 4: Upon receiving a sale quote from Mer, at the next time point, Ebank commits to either informing Mer that Cus's credit is not approved ($\Box credit\_not\_appr@M_B$) or arranging a credit payment to Mer ($\bigcirc credit\_paym@M_B$). The decision is dependent on the credibility of Cus and hence is external ($\oplus$) to Ebank and Mer:
$\sigma \Box [sale\_quote@M_M \multimap \bigcirc(\Box credit\_not\_appr@M_B) \oplus$
$\Box credit\_paym@M_B]$

Rules at agent Cus

Cus has an amount of 50 dollars available at any time, can be used for credit payment or cash payment: $\Box\$50@C$.
Cus has a commitment of obtaining a cricket bat at some time: $[\Diamond cricket\_b@C_C]^{\perp}$.

Rule 5: Cus will pay Mer via Paypal if there is an indication from EBank that Cus's credit is not approved:
$\sigma \Box [credit\_not\_appr@C_B \multimap \Box Paypal\_paid@M_M]$

4.2 Description of the interaction

Cus requests from Mer a cricket bat at some time. Mer replies with a proposal in which each cricket bat costs 10 dollars. Cus needs to prepare 10 dollars and payment can be made by credit or via Paypal.

Assuming that Cus only pays via Paypal if credit payment fails, Cus will let Mer charges by credit. Mer will then ask EBank to arrange a credit payment. EBank proposes that Mer gives a quote of sale and depending on Cus's credibility, at the next time point, either credit payment will be arranged or a disapproval of Cus's credit will be informed. Mer accepts and fulfills the conditions. If the first case happens, credit payment is done. If the second case happens, credit payment is failed, Cus may back track to take the option paying via Paypal.

Once payment is arranged, Mer will apply its original proposal to satisfy the Cus's request of a cricket bat and hence removing one cricket bat and adding 10 dollars into its set of resources.

4.3 Interaction

1. Cus can not fulfill its commitment of $[\Diamond(cricket\_b@C_C)]^{\perp}$ and hence, makes a request to Merchant:
C to M: **REQUEST** $[\Diamond cricket\_b@C_C]^{\perp}$

2. To meet the request, Mer searches for applicable rules. One applications of rule 1 can derive $\Box cricket\_b@C_C$ and $\Box cricket\_b@C_C \vdash \Diamond cricket\_b@C_C$. Mer will propose rule 1 at a time instance n1 to Cus as a pre-commitment.
M to C: **PROPOSE**
$\bigcirc^{n1}[10\$@C_C \otimes (Paypal\_paid@M_M \oplus credit\_paid@M_M)$
$\multimap 10 \Box \$@M_M \otimes \Box cricket\_b@C_C \otimes cricket\_b@M_M^{\perp}]$

With similar analysis, Cus determines that given the con-

ditions can be satisfied, the proposals can help to derive its request. Hence,
C to M: **ACCEPT**

Cus analyzes the conditions of the accepted proposal by proof search.
$\vdash \bigcirc^{n1} 10\$@C_C$;
$\vdash \bigcirc^{n1} Paypal\_paid@M_M$ *or* $\vdash \bigcirc^{n1} credit\_paid@M_M$; –(*)–
$\vdash \bigcirc^{n1} 10\$@C_C \otimes (\bigcirc^{n1} Paypal\_paid@M_M \oplus \bigcirc^{n1} credit\_paid@M_M)$
$\vdash \bigcirc^{n1}(10\$@C_C \otimes (Paypal\_paid@M_M \oplus credit\_paid@M_M))$

From (*), one way to satisfy the conditions is for Cus to derive, at the next n1 time points, 10 dollars ($\bigcirc^{n1} 10\$@C_C$); and to choose paying via Paypal ($\bigcirc^{n1} Paypal\_paid@M_M$) OR by credit payment ($\bigcirc^{n1} credit\_paid@M_M$).

3. Deriving $\bigcirc^{n1} 10\$@C_C$: as Cus has 50 dollars, it can make use of 10 dollars: $10\Box\$@C_C \vdash \Box 10\$@C_C \vdash \bigcirc^{n1} 10\$@C_C$.

There are two options for payment method, the choice is at agent Cus. We assume that Cus prefers credit payment.
4. Deriving $\bigcirc^{n1} credit\_paid@M_M$: Cus can not derive this formula by itself, hence, it will make a request to Mer:
C to M: **REQUEST** $[\bigcirc^{n1} credit\_paid@M_M]^{\perp}$.

5. Rule 2 at Mer is applicable but Mer can not derive its condition ($\bigcirc^{n1} credit\_paym@M_B$). Hence, Mer will further make a request to EBank.
M to E: **REQUEST** $[\bigcirc^{n1} credit\_paym@M_B]^{\perp}$
Ebank searches and finds rule 4 applicable. Because $credit\_paym@M_B$ will be available one time point after the rule's application time, Ebank proposes to Mer an instance of rule 4 at the next **n1-1** time points.

6. B to M: **PROPOSE** $\bigcirc^{n1-1}[quote@M_M \multimap \bigcirc(\Box\, credit\_not\_appr@M_B \oplus \Box credit\_paym@M_B)]$

With similar analysis, Mer accepts the proposal.
M to B: **ACCEPT**
The rule condition is fulfilled by Mer as $\Box quote@M_M \vdash \bigcirc^{n1-1} quote@M_M$. Hence, Ebank then applies the proposal to derive:
$\bigcirc^{n1-1} \bigcirc (\Box credit\_not\_appr@M_B \oplus \Box credit\_paym@M_B)$.
$\oplus$ indicates the choice is external to both agents. There are two cases, Cus's credit is approved or disapproved.

For simplicity, we show only the case where Cus's credit is approved. At the next (n1-1) time point,
$\bigcirc^{n1-1} \bigcirc (\Box credit\_not\_appr@M_B \oplus \Box credit\_paym@M_B)$ becomes $\bigcirc^{n1-1} \bigcirc \Box credit\_paym@M_B \vdash \bigcirc^{n1} credit\_paym@M_B$.
As a result, at the next n1 time points, **Ebank will arrange the credit payment.**

7. **Mer fulfills Cus's initial request.**
When any of $\bigcirc^{n1} Paypal\_paid@M_M$ (if Cus pays via Paypal) or $\bigcirc^{n1} credit\_paid@M_M$ (if Cus pays by credit card) is derived, $\bigcirc^{n1}(credit\_paym@M_M \oplus Paypal\_paid@M_M)$ is also derived, hence **the payment method is arranged**.

Together with the other condition $10\$@C_C$ being satisfied, this allows the initial proposal to be applied by Mer to derive $\bigcirc^{n1} cricket\_b@C_C$ and a commitment of $\bigcirc^{n1} cricket\_b@M_M^{\perp}$ for Mer, which is also resolved by the resource $\Box cricket\_b@M_M$ available at Mer.
Any values of n1 such that $n1 - 1 \geq 0 \Leftrightarrow n1 \geq 1$ will allow Mer to fulfill Cus's initial request of $[\Diamond cricket\_b@C_C]^{\perp}$.
The interaction ends as all commitments are resolved.

4.4 Flexibility

The desired flexibility has been achieved in the example. It is Cus's own decision to proceed with the preferred payment method. Also, non-determinism that whether Cus's credit is disapproved or credit payment is made to Mer is faithfully represented. If an exception happens that Cus's credit is not approved, $credit\_not\_appr@C_B$ is produced and Cus can backtrack to paying via Paypal. Rule 5 will then be utilized to allow Cus to handle the exception by paying via Paypal.

Moreover, in order to specify that making payments and sending cricket bats can be in any order, we can add $\Diamond$ in front of payment method in rule 1 as follows:
$\sigma \Box [10\$@C_C \otimes \Diamond(Paypal\_paid@M_M \oplus credit\_paid@M_M) \multimap 10 \Box \$@M_M \otimes \Box cricket\_b@C_C \otimes cricket\_b@M_M^{\perp}]$.

This addition in the condition of the rule means that the time of payment can be any time up to Cus's choice, as long as Cus pays and hence, the time order between making payments and sending goods becomes flexible.

5. ENCODING ISSUES

5.1 Advantages of TLL framework

Our TLL framework has demonstrated natural and expressive specification of agent interaction protocols. Linear implication ($\multimap$) expresses causal relationship, which makes it natural to model a removal or a consumption, especially of resources, together with its consequences. Hence, in our framework, resource transformation is modeled as a linear implication of the consumed resources to the produced resources. Resource relocation is modeled as a linear implication from a resource at one agent to that resource at the other agent. Linear implication also ensures that fulfillment of the conditions of a conditional commitment will cause the commitments to happen. Moreover, state updates of agents are resulted from a linear implication from the old state to the current state.

Temporal operators ($\bigcirc$, $\Box$ and $\Diamond$) and their combinations help to specify the time of actions, of resource availability and express the time order of events. Particularly, precise time points are described by the use of $\bigcirc$ operator or multiple copies of it. Based on this ability to specify correct time points for actions or events, time order or sequencing of them can also be captured. Also, a sense of duration is simulated by spreading copies of the resources or actions' formulas across multiple adjacent time points. Moreover, uncertainty in time can represented and reasoned about by the use of $\Box$ and $\Diamond$ and their combinations with $\bigcirc$. $\Box$ can be used to express outer non-determinism while $\Diamond$ expresses inner non-determinism. These time properties of resources, actions and events are correctly respected through out the agent reasoning process based on sequent calculus rules.

Furthermore, the centrality of the notion of commitment in agent interaction has been recognized in many frameworks [11, 12, 1, 10, 4]. However, to the best of our knowledge, modeling commitments directly at the propositional level of such a resource conscious and time aware logic as TLL is

firstly investigated in our framework. Our framework models base commitments as negative formulas and conditional commitments via the use of linear implication and/or negative formulas. The modeling of commitments has a number of advantages:

- Commitments are represented directly at the propositional logic level or via a logic connective rather than a non-logical construct like [11], which makes treatment of commitments more natural and simple and allows to make use of readily available proof search systems like using sequent calculus for handling commitments. Existing logic connectives like $\otimes, \&, \oplus, \multimap$ are also readily available for describing the relationships among commitments.
- Fulfillment of commitments then becomes deriving the corresponding positive formulas or condition formulas, which then simply reduces to a proof search task. Also, given the required formulas, fulfillment of commitments can be implemented easily and automatically as deduction ($com \otimes com^{\perp} \vdash \perp$).
- The enforcement of commitments is also internal and simply implemented via the assumption that agents are driven to remove all negative formulas for base commitments and via the use of linear implication for conditional commitments.

Regarding making protocol specification more flexible, our approach has marked a number of significant points.

Firstly, flexibility of protocol specifications in our framework comes from the expressive power of the connectives of TLL. $\&$ and $\oplus$ refer to internal and external choices of agents on resources and actions while $\Box$ and $\Diamond$ refer to internal choices and external choices in time domain. Given that flexibility includes the ability to make a sensible choice, having the choices expressed explicitly in the specification of interaction protocols provides agents with an opportunity to reason about the right choices during interaction and hence explore the flexibility in them.

Secondly, instead of being sequences of interactive actions, protocols are structured on commitments, which are more abstract than protocol actions. Execution of protocols is then based on fulfilling commitments. Hence, unnecessary constraints on which particular interactive actions to execute by which agents and on the order among them are now removed, which is a step forward to flexibility as compared to traditional approaches. On the other hand, in the presence of changes introduced externally, agents have the freedom to explore new sets of interactive actions or skip some interactive actions ahead as long as they still fulfill the protocol's commitments. This brings more flexibility to the overall level of agents' interactive behaviors, and thus the protocol.

Thirdly, the protocol is specified in a declarative manner essentially as sets of pre-commitments at each participating agents. To achieve goals, agents use reasoning based on TLL sequent calculus to construct proofs of goals from pre-commitments and state formulas. This essentially gives agents an autonomy in utilization of pre-commitments and hence agents can adapt the ways they use these to flexibly deal with changing environments.

In particular, as proof construction by agents selects a sequence of pre-commitments for interaction, being able to select from all the possible combinations of pre-commitments in proof search gives more chances and flexibility than selecting from only a few fixed and predefined sequences. It is then also more likely to allow agents to handle exceptions or explore opportunities that arise. Moreover, as the actual order of pre-commitments is determined by the proof construction process rather than predefined, agents can flexibly change the order to suit new situations.

Fourthly, changes in the environment can be regarded as removing or adding formulas onto the state formulas. Because the proof construction by agents takes into account the current state formulas when it picks up pre-commitments, changes in the state formulas will be reflected in the choice of which relevant pre-commitments to proceed. Hence, the agents have the flexibility in deciding what to do to deal with changes.

Lastly, specifying protocols in our framework has a modular approach which adds ease and flexibility to the designing process of protocols. Protocols are specified by placing a set of pre-commitments at each participating agent according to their roles. Each pre-commitment can indeed be specified as **a process in its own** with condition formulas as its input and commitment part's formulas as its output. Execution of each conditional commitment is a relatively independent thread and they are linked together by the proof search to fulfill agents' commitments. As a results, with such a design of pre-commitments, one pre-commitment can be added or removed without interfering the others and hence, achieving a modular design of the protocols.

5.2 Limitations of TLL Framework on Modeling

As all the temporal operators in TLL refer to concrete time points, we can not express durations in time faithfully. One major disadvantage of simulating a duration of an event by spreading copies of that event over adjacent time points continuously (like $\bigcirc A \otimes \bigcirc^2 A \otimes \ldots \bigcirc^{10} A$) is that it requires the time range to be provided explicitly. Hence, such notion like until can not be naturally expressed in TLL.

Commitments of agents can be in conflict, especially when resolving all of them requires more resources or actions than what agents have. Our work has not covered handling commitments that are in conflict.

Another troublesome aspect of this approach is that the rules for interaction require some detailed knowledge of the formulas of temporal linear logic. Clearly it would be beneficial to have a visually-based tool similar to UML diagrams which would allow non-experts to specify the appropriate rules without having to learn the details of the formulas themselves.

6. CONCLUSIONS AND FURTHER WORK

This paper uses TLL for specifying interaction protocols. In particular, TLL is used to model the concept of resource, capability, pre-commitment and commitment with tight integration as well as their manipulations with respect to time. Agents then make use of proof search techniques to perform the desired interactions.

In particular, the approach allows protocol specifications to capture the meaning of interactive actions via commitments, to capture the internal choices and external choices of agents about resources, commitments and about time as well as updating processes. The proof construction mechanism

provides agents with the ability to dynamically select appropriate pre-commitments, and hence, help agents to gain the flexibility in choosing the interactive actions that are most suitable and the flexibility in the order of them, taking into consideration on-going changes in the environment.

Many other approaches to modeling protocols also use the commitment concept to bring more meaning into agents' interactive actions. Approaches based on commitment machines [11, 12, 10, 1] endure a number of issues. These approaches use logic systems that are limited in their expressiveness to model resources. Also, as an extra abstract layer of commitments is created, more tasks are created accordingly. In particular, there must be a human-designed mapping between protocol actions and operations on commitments as well as between control variables (fluent) and phases of commitment achievement. Moreover, external mechanisms must be in place to comprehend and handle operations and resolution of commitments as well as enforcement of the notion of commitment on its abstract data type representations. This requires another execution in the commitment layer in conjunction with the actual execution of the protocol. Not only these extra tasks create an overhead but also makes the specification and execution of protocols more error prone.

Similar works in [8] and [9] explore the advantages of linear logic and TLL respectively by using partial deduction techniques to help agents to figure out the missing capabilities or resources and based on that, to negotiate with other agents about cooperation strategies. Our approach differs in bringing the concept of commitment into the modeling of interaction, and providing a more natural and detailed map for specifying interaction, especially about choices, time and updating using the full propositional TLL. Moreover, we emphasize on the use of pre-commitments as interaction rules with a full set of TLL inference rules to provide the advantages of proof construction in achieving flexible interaction.

Our further work will include using TLL to verify various properties of interaction protocols such as liveness and safety. Also, we will investigate developing an execution mechanism for such TLL specifications in our framework.

Acknowledgments

We are very thankful to Michael Winikoff for many stimulating and helpful discussions of this material. We also would like to acknowledge the support of the Australian Research Council under grant DP0663147.

7. REFERENCES

[1] A. K. Chopra and M. P. Singh. Contextualizing commitment protocol. In *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 1345–1352, New York, NY, USA, 2006. ACM Press.

[2] E. A. Emerson. Temporal and modal logic. *Handbook of Theoretical Computer Science*, B, Chapter 16:995–1072, 1990.

[3] J.-Y. Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.

[4] A. Haddadi. *Communication and Cooperation in Agent Systems: a pragmatic theory*. Springer-Verlag, Berlin Heidelberg, 1995.

[5] J. Harland and M. Winikoff. Agent negotiation as proof search in linear logic. In *AAMAS '02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, pages 938–939, New York, NY, USA, 2002. ACM Press.

[6] T. Hirai. *Temporal Linear Logic and Its Applications*. PhD thesis, Graduate School of Science and Technology, Kobe University, 2000.

[7] N. R. Jennings. Commitments and conventions: The foundation of coordination in multi-agent systems. *The Knowledge Engineering Review*, 8(3):223–250, 1993.

[8] P. Küngas. Linear logic, partial deduction and cooperative problem solving. In J. A. Leite, A. Omicini, L. Sterling, and P. Torroni, editors, *Declarative Agent Languages and Technologies, First International Workshop, DALT 2003. Melbourne, Victoria, July 15th, 2003. Workshop Notes*, pages 97–112, 2003.

[9] P. Küngas. Temporal linear logic for symbolic agent negotiation. *Lecture Notes in Artificial Intelligence*, 3157:23–32, 2004.

[10] M. Venkatraman and M. P. Singh. Verifying compliance with commitment protocols. *Autonomous Agents and Multi-Agent Systems*, 2(3):217–236, 1999.

[11] P. Yolum and M. P. Singh. Commitment machines. In *Proceedings of the 8th International Workshop on Agent Theories, Architectures, and Languages (ATAL-01)*, pages 235–247. Springer-Verlag, 2002.

[12] P. Yolum and M. P. Singh. Flexible protocol specification and execution: applying event calculus planning using commitments. In *AAMAS '02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, pages 527–534, New York, NY, USA, 2002. ACM Press.

APPENDIX

A. TEMPORAL SEQUENT RULES FOR TLL

$$\frac{A, \Gamma \vdash \Delta}{\Box A, \Gamma \vdash \Delta}\ \Box L \qquad \frac{!\Gamma, \Box\Delta \vdash A, \Diamond\Lambda, ?\Sigma}{!\Gamma, \Box\Delta \vdash \Box A, \Diamond\Lambda, ?\Sigma}\ \Box R$$

$$\frac{!\Gamma, \Box\Delta, A \vdash \Diamond\Lambda, ?\Sigma}{!\Gamma, \Box\Delta, \Diamond A \vdash \Diamond\Lambda, ?\Sigma}\ \Diamond L \qquad \frac{\Gamma \vdash A.\Delta}{\Gamma \vdash \Diamond A, \Delta}\ \Diamond R$$

$$\frac{!\Gamma, \Box\Delta, \Xi \vdash A, \Phi, \Diamond\Lambda, ?\Pi}{!\Gamma, \Box\Delta, \bigcirc\Xi \vdash \bigcirc A, \overline{\bigcirc}\ \Phi, \Diamond\Lambda, ?\Pi}\ \bigcirc$$

$$\frac{!\Gamma, \Box\Delta, \Xi, A \vdash \Phi, \Diamond\Lambda, ?\Pi}{!\Gamma, \Box\Delta, \bigcirc\Xi, \overline{\bigcirc}\ A \vdash \overline{\bigcirc}\ \Phi, \Diamond\Lambda, ?\Pi}\ \overline{\bigcirc}$$

$$\frac{!\Gamma, \Box\Delta, \Xi \vdash \Phi, \Diamond\Lambda, ?\Pi}{!\Gamma, \Box\Delta, \bigcirc\Xi \vdash \overline{\bigcirc}\ \Phi, \Diamond\Lambda, ?\Pi}\ \bigcirc \rightarrow \overline{\bigcirc}$$