Specifying the Processes and States of Negotiation

Shamimabi Paurobally and Jim Cunningham
Imperial College of Science and Technology, London
{sp1, rjc}@doc.ic.ac.uk

Abstract

Negotiation can be considered to be an important aspect of commerce. It is part of those wider dynamic processes whereby commercial goals are achieved by the parties to a contract. Overt negotiation, as deal making, is often suppressed by agreed rules of encounter, but it is rarely absent altogether. In this paper, while recognising the need for more complete logics to represent both states and processes with abstraction, we start to build a more formal link between negotiation and Artificial Intelligence. We illustrate the use of dynamic logic to specify a shopping scenario between a retailer and a customer agent. The negotiation that arises in such a scenario can follow one of the negotiation models described. From a given negotiation model we obtain the corresponding negotiation protocol. These formulations have allowed us to remove inconsistencies and ambiguities in less formal models and to suppress issues such as concurrency. Finally, we discuss how an agent, given that it has mental states and a library of plans, can find a path for negotiation that will not only be successful in achieving its goal but also be optimal.

Keywords: negotiation, electronic commerce, goals, dynamic logic.

1 Introduction

Studies in various sectors indicate technical features needed for successful interactive electronic commerce: security, privacy, payment, auditing and notary services, negotiation, brokering as well as human expertise (Cunningham et al 1998). Current on-line trading systems do not support all of these features, although there tends to be provision for security and payment. Users, providers and software developers also need interoperability protocols so that systems can interact with components in different domains and enable service providers and customers to coordinate and converge towards mutually satisfactory deals. In an open market, customers have to be able to inspect service offers regardless of the specific tool used to publish them. Standardised access to catalogues must be coupled with complete, up-to-date distribution and inspection of information to allow greater choices, wider user access, on-line enrolment and compatibility with other catalogues. Then, with each party having their own model of what makes a good deal and a bad deal, negotiation mechanisms can allow interaction between consumers and providers towards the resolution of an agreement. An agreement between parties must be established and drafted as a contract and subsequently engaged. A contract is a conceptually shared structure that binds parties together in a set of mutual obligations such as the exchange of commercial services. Finally, customer retention, service quality and competitiveness of on-line service delivery are key aspects of relationship management, involving long term interaction between provider and customer.

So for electronic commerce to flourish we need to expedite negotiations and provide standardised protocols and methods for the automation or partial automation of negotiation as a step towards agreement resolution. For example, negotiation can arise in several cases during a shopping process, when negotiating on price and availability, on the payment method or on the after-sales services. A negotiation process is a joint process between a number of agents, where there are various negotiation states that represent the degrees of co-operation, commitment and agreement. In this paper, we present a shopping scenario between a retailer and a customer agent and its representation in a dynamic logic. From the shopping scenario we can see where negotiation processes arise and that these negotiations may comply with a negotiation model. We also provide an abstract theory of the states involved in the shopping scenario and derive the paths leading to a particular shopping state. We then give several negotiation models that can be followed by agents engaged in a negotiation process. The possible paths of a negotiation are derived from the diagrammatic specification of the negotiation models using

dynamic logic. A negotiation can follow a possible set of paths to reach a goal state. Finally we discuss the question of how an agent can find a successful negotiation path given it has mental states and it complies with the negotiation model.

2 Representing States and Processes

It is difficult to find notation where processes and states are given equal status, let alone form the basis for a simple and rational calculus for an executable system. Traditional imperative programming languages, such a C and Pascal leave reasoning about programs to external axiomizations which impose complexity and incompleteness. Object oriented languages provide otherwise missing capacity for data abstraction, but without solving the problems of the need for a rational calculus. Process calculi like CCS (Milner 1989) do not consider state, while specification languages like Z are removed from executable systems. In logic programming, we see demonstration of the practical capacity of predicate logic to define and combine data to give more abstract properties through which one can reason about machine state, but at some cost in confusing state transition processes with inference itself. Dynamic logic (Goldblatt 1987) is rare in providing reasoning about the effect of processes on states of affairs, but in its primitive form it lacks process abstraction, and has no seriously executable form as a programming system.

Thus there is no established notation to even represent both the states and processes of an active agent, let alone a calculus for deciding why a particular negotiation should achieve the mutual goals. Instead we shall improvise, conceptually treating an agent as capable of atomic actions, each constituting a primitive process which may be combined in more complex ones. Dynamic logics of action fit naturally with a multi-modal theory of agent beliefs, goals and intentions. A suitably rich first order logic with action terms and variables appears capable of a practical reasoning system which can relate processes to goal states.

Informal Syntax of the Logic

Our syntax is an adaption of the program logic described in (Goldblatt 1987). We associate an agent with processes by prefixing the process with an agent in the same way an object is suffixed by its methods. So the party executing a process and the process itself are separated with a full stop, e.g., *r:retailer.display* means retailer *r* executes the *display* process. Usually we omit the agent category and denote a joint process between two parties with '$\cup$' as in *(c* $\cup$ *r).shopping*. A process, whether joint or not, may be decomposed into a sequence of sub-processes each coupled with the agent or agents executing that sub-process. An action is an atomic process. The process denoted by *a*;*b* is composed of *a* followed by *b* in sequence, a^* denotes zero or more iterations and a^+, one or more iterations e.g. {*r* $\cup$ *c* }. *shopping = r:retailer.display ; c:customer.browse*$^+$*; c:customer .choose**

A state test operator *?* allows sequential composition to follow only if successful. For example *c.browse*? ; *c.choose* is the process *c.choose* if *c.browse* is true, otherwise it fails. We remark at this stage that a conventional program-like *if … then…, repeat … until …* commands could be more intuitive than the primitive test, union and iteration operators, ?, $\cup$, *, $^+$, borrowed from dynamic logic for task composition, but we reserve judgement on notation because there are tougher criteria in relating successful task composition to the goals and constraints of the agents.

When we specify the informal models of negotiation presented in section 4 we introduce additional connectives to give a logic that applies to a list of propositions. We use the prolog style for denoting lists, so that an empty list is [ ], a singleton list is [*X*] and a list containing at least one proposition is [*X*|*T*]. Thus '*none_of*' is a connective that takes a list of propositions and returns true if all the propositions are false, '*one_of*' takes a list of propositions and returns true if exactly one of the propositions is true. Formally,

none_of (**[ ]**) $\leftrightarrow$ *true*, *none_of* (**[** *H* | *T* **]**) $\leftrightarrow$ $\neg$ *H* $\wedge$ *none_of* (*T*).

one_of (**[ ]**) $\leftrightarrow$ *false* , *one_of* (**[** *H*|*T* **]**) $\leftrightarrow$ (*H* $\wedge$ *none_of* (*T*)) $\vee$ ($\neg$ *H* $\wedge$ *one_of* (*T*)).

For example *none_of* (**[** *A, B, C* **]**) $\leftrightarrow$ $\neg A \wedge \neg B \wedge \neg C$, while

one_of (**[** *A, B, C***]**) $\leftrightarrow$ $(A \wedge \neg B \wedge \neg C) \vee (\neg A \wedge B \wedge \neg C) \vee (\neg A \wedge \neg B \wedge C)$.

3 Tasks of a Shopping Scenario

Consider the interaction between a retailer and a consumer, familiar to all as shopping. One can envisage a simple retail agent as seeking to maximise a long term profitability goal by wise buying, efficient distribution and stock management, and successful sales. In contrast, the goals of a retail customer are more subtle, involving, say maintenance of a personally adequate supply of garments,

foods, or other consumables, through purchase within a fixed budget. The joint shopping process is illustrated in Figure 1 by an incomplete hierarchical JSD diagram (Jackson 1975), where task sequencing is left to right, iteration is loosely indicated by the symbol *, and shading distinguishes the activities of different or joint agents. Note that at the lowest level each task is attributed to a single agent, but it is not yet necessarily atomic.

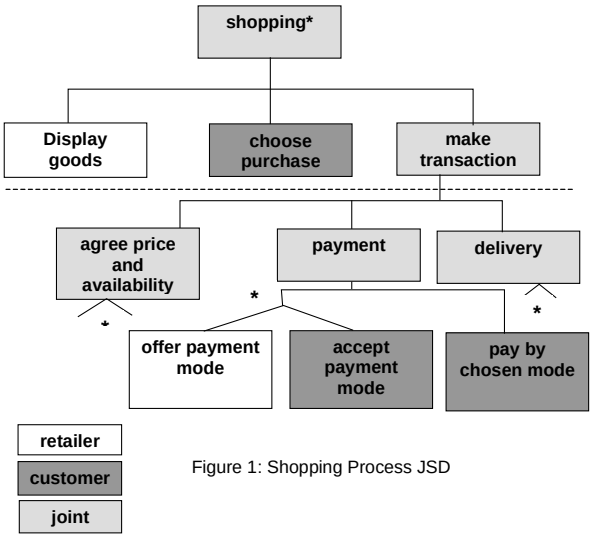

Figure 1: Shopping Process JSD

The shopping process is a joint process which can be decomposed into abstract parts done by either one agent or both agents at a time. First the retail agent displays the goods being offered, perhaps in the form of a catalogue or an event sent on an event channel to broadcast what is proposed. The customer then browses through the offers and chooses what he/she wants to purchase. At this stage there has not been any interaction between the two agents except for the sequencing itself, and synchronising this is a suppressed detail. The process of making a transaction will embody greater interaction between the two agents. In figure 1, the *make transaction* joint process has been broken down into three joint processes. These three processes can themselves be further broken down into sequential actions done jointly by the two agents or by a single agent. The illustrated decomposition is by no means definite and is just one of many possible decompositions.

Using the process operations we can express the hierarchical shopping task as a hierarchical joint process:

(*retailer* ∪ *customer*).*shopping* = (*r*:*retailer.display_goods; c:customer.choose_purchase*;
{*r* ∪ *c*}.*make_transaction*)*

{*r* : *retailer* ∪ *c* : *customer* }. *make_transaction* = {*r* ∪ *c*}.(*agree_price_and_availability* ; *payment* ;
delivery)

{*r*:*retailer* ∪ *c:customer*}.*payment* = (*r.offer_payment_mode* (*m*); *c.accept_payment_mode*(*m*))$^{+}$;
c.pay_by_mode(*m*)

{*r*:*retailer* ∪ *c:customer*}.*agree_price_and_availability* = *c.within_budget*(*r.advertised_price*) ?;
r.in_stock? *null.*

The first composition expresses the shopping process is a joint process between the retailer and customer agents and consists of three sequential processes: the retailer *display goods*, the customer *choose purchase* and the joint *make_transaction* process. The joint process of agreeing on price and availability is composed from two sequential test conditions. If the condition that the retailer's advertised price is within the customer's budget succeeds then the next condition of the service being in stock at the retailer is tested. Although a faithful representation of the joint processes implicit in the JSD diagram, this is itself an idealisation which ignores abnormal termination, such as when the customer makes no choice.

4 Joint States of the Shopping Scenario

If we consider the effects of the actions of the agents on the states of the shopping process we can represent them in a shopping state transition diagram whose essence is a dual of the above Jackson Diagram. We use a notation similar to Harel's statechart notation (Harel and Namad 1996). Abnormal transitions can be portrayed conveniently in this notation, and supplements the state transitions implicit in figure 1. Whereas figure 1 is portraying joint tasks and process hierarchy of the shopping process and of the agents, figure 2 is a high-level state transition model that portrays the hierarchy of joint states for the shopping process linked by the abstract *display goods, choose purchase* and *make_ transaction* tasks. We can derive the essential structure of the state transition diagrams from the coloured JSD and vice versa, provided we ignore abnormal transitions. Through a JSD to represent the process hierarchy and a collection of diagrams for the implicit state transitions, we can suppress the full complexity of potentially concurrent interaction processes.

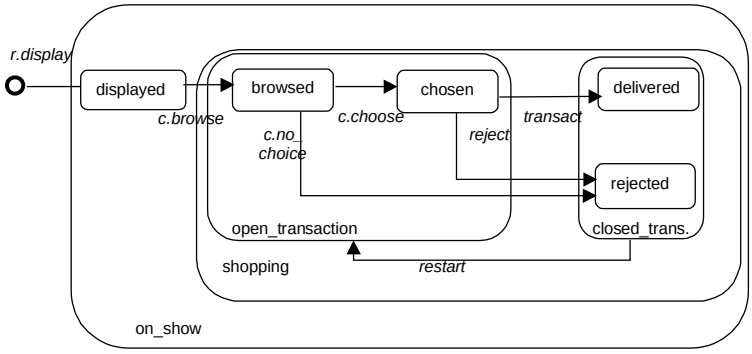

Figure 2: First Level State Transition Diagram of the Shopping Process

In figure 2, the parent state *on_show* is the overall state condition. The substate *shopping* is entered by the customer browsing. The substates of the *shopping* state are *open_transaction and closed_transaction* which themselves include other substates. *Rejected* and *delivered* are terminal substates that may be the subgoals of the agents. Section 6 deals with possible ways for achieving these goals and subgoals. The shopping state is entered either by the retailer making a display, leading to the *displayed* state or the entrance of a customer leading to an *open_transaction* where the customer may not have chosen yet. Various state transitions change the substates of *shopping* and eventually terminate in the *closed_transaction* state. From the *closed_transaction* state, the customer or the retailer can restart a transaction, shown in figure 2 by the restart transition and shown in figure 1 by the iteration of the shopping process.

The high level state transition *transact* (*make_transaction* in figure 1) can be further broken down as required by the JSD to show negotiation over the price and availability, the mode of payment and the delivery. The negotiation over price may follow the state transition model of something like figure 3, leading to a *closed* state. Likewise, after the retailer has offered goods for a certain price and the customer has agreed to this offer, negotiation on the payment mode can follow the same negotiating model but parameterised with a different issue i.e. the mode of payment. Eventually the state of negotiation on payment mode changes from *open* to *agreed*, a substate of the *closed* state.

Each state in the joint shopping model may also imply certain consequential states of affairs on each party's side. For example in the *displayed* state of the shopping process we may infer a parent state of the retailer having certain goods for sale. Similarly the *chosen* state of the shopping process may imply a need for a customer state of *has enough money*. So the state transition diagram in figure 2 represents the joint states of the shopping process but implies states of the involved agents. By speaking of joint states, we are supposing that the corresponding distinct states of the individual agents are implicitly known and controlled by lower level protocols for interaction. We do not discuss these assumptions here. In a distributed system they must be ensured by task and session management.

A Logical theory of the state transition diagram

We can represent the relation between parent states and their substates and the state transitions in dynamic logic. The logical theory shown here is that of the high level state diagram in figure 3 and

describes the relations between the high level states. The state transitions can themselves be decomposed into more detail. So the *transact* transition can follow some negotiation model with its corresponding theory.

The first three equivalencies show the relation between the parent and the substates. For example the first one says that the *shopping* state consists of disjoint *open_transaction* and *closed_transaction* substates, so if the current state is either *open_transaction* or the *closed_transaction* state then the *shopping* state is valid. The relation between *open_transaction* and *closed_transaction* states and their substates are similarly represented:
shopping ↔ *one_of* **[** *open_transaction, closed_transaction* **]**
closed_transaction ↔ *one_of* **[** *delivered , rejected* **]**
open_transaction ↔ *one_of* **[** *browsed, chosen* **]**
on_show ↔ *one_of* **[** *displayed, shopping* **]**

The actions performing state transitions can also be characterised by equivalences:
¬ *on_show* ↔ [*r.display*] *displayed* (1)
displayed ↔ [*c.browse*] *browsed* (2)
browsed ↔ *one_of* **[** [*c.no_choice*] *rejected*, [*c.choose*] *chosen***]** (3)
chosen ↔ *one_of* **[** [{*r* ∪ *c* }. *reject*] *rejected* , [{*r* ∪ *c* }. *transact*] *delivered* **]** (4)
closed_transaction ↔ *restart* [*open_transaction*] (5)

According to action condition 1, provided the retailer was not already involved in an *on_show* state, when the retailer makes a display action, the state changes to *displayed*. The entrance of the customer by browsing, in condition 2, leads to the *browsed* state which is a substate of *open_transaction* and *shopping* states.

Likewise for equivalence 3, if the current state is *browsed*, either the customer does not choose and triggers the state of the shopping to the *rejected* state or the customer invokes a choose transition to the *chosen* state. The terminal *closed_transaction* states, *rejected* and *delivered* can be entered from the *chosen* source state by the customer or the retailer making a *reject* or after a successful joint *transact* state transition.

The Paths of the Shopping Process
We can consider the shopping process as a graph whose nodes are the states and edges are the state transitions. The logical theory above represents the shopping process graph. Condition 1 gives the entry edges and conditions 2, 3, 4 and 5 the internal edges of the graph. The *delivered* node or state is the goal. We need to find paths to get from an entry node to the goal node. Here we can use the '?' notation to test whether we are able to find a path to a certain node or state and check if that state holds.

For example the paths to an *open_transaction* state are the union of the following sets of paths
- the customer entering the *open_transaction* through a *browse* action
- paths to the *closed_transaction* state in sequence with the edge of either the customer or the retailer restarting the transaction.
- paths to the *chosen* state

From our earlier presumption that an open transaction state is either a *browsed* or a *chosen* state we can infer the paths to the *closed_transaction* state and thus the paths to an agent's goal. Let p_0 be paths to *open_transaction* and p_c be paths to *closed_transaction*. Then p_0 and p_c can be derived from the above conditions.
$p_0 = p_c$; *restart* ∪ ((¬ *on_show*)? *r.display ; c.browse*) (6)
$p_c = p_0$; ((*chosen* ? *transact* ∪ *reject*) ∪ (*browsed* ?; *c.choose*; (*transact* ∪ *reject*) ∪ *c.no_choice*)) (7)

Eliminating p_0 by substitution in (7) gives:
p_c = (p_c ; *restart* ∪ ((¬ *shopping*)? *r.display ; c.browse*)) ; ((*chosen* ? *transact* ∪ *reject*) ∪ (*browsed*?; (*c.choose* ; (*transact* ∪ *reject*) ∪ *c.no_choice*)) (8)

Let *entries* = (¬ *on_show*)? *r.display ; c.browse* and *exits* = (*chosen* ? *transact* ∪ *reject*) ∪ (*browsed* ?; *c.choose* ; (*transact* ∪ *reject*) ∪ *c.no_choice*). We thus have:
$p_c = p_c$; *restart* ; *exits* ∪ *entries*; *exits*, whence: (9)
p_c = *entry_points*; *exits* ∪ *(restart* ; *exits*) * (10)

So a path to the *closed_transaction* state consists of entry to the *open_transaction* state followed by *exit* paths from that state with optional reiterations by restarting the transaction followed by *exit* paths. The recursive path equation (9) is solved as (10) in the regular algebra of paths, when viewing the state transition diagram as a graph, by postulating the inference rule:

$$(x = x\ ;\ A \cup B) \Rightarrow (x = B;A^*).$$

See (Backhouse and Carre 1973).

5 Bilateral Negotiation Models

Following our illustration of task analysis for the first level processes and states of the joint shopping process, we find that details of the transaction involve a sequence of negotiations over price, payment and delivery. These too can have similar representation and analysis.

A negotiation protocol is the set of public rules that dictate the conduct of an agent towards other agents when carrying out some negotiation. Agents involved in the same negotiation need to comply to a common negotiation protocol to ensure that they and all other participants following the same rule will coordinate meaningfully. The models presented here show the relationships between the states of a negotiation process and permitted transitions between the states, so implicitly define a negotiation protocol which is made explicit in the logical theory of the model. A group of agents involved in a negotiation process can invoke state transitions to achieve terminal states or goal states. If the goal state is to reach an agreement and is achieved, an engagement process can be launched for commitment and for setting up contracts between the agents. These models can form the basis for further customisation into more specific negotiation models. Negotiation may also be on various issues that could be a concurrent execution of single-issue models or another compound model.

In a negotiation process, there are constraints on joining the negotiation and on the disclosure of information. An agent who is not involved in a particular negotiation cannot become a member or join in without authorisation. An agent may be involved in several concurrent negotiations. In each, an agent has a task to negotiate. It associates the negotiation process with its participants, the subject and model of negotiation. So an agent is able to distinguish between its possibly multiple negotiations to decide the current negotiation state and in choosing the appropriate action. Implicitly each negotiation state is characterised with a tuple (negotiation process, participants, negotiation model, negotiation subject) and each negotiation action is implicitly subscripted with the tuple (negotiation process, perpetrator of that action, negotiation model, negotiation subject).

5.1 Bilateral Negotiation Models

The bilateral negotiation model defines the protocol followed by two parties looking for an agreement concerning a subject of negotiation. The negotiation is over one issue called the negotiation subject. In bilateral negotiation, we can think of a world inhabited by two negotiating agents, a negotiation process in a particular state, a model of negotiation and a subject of negotiation. There are a number of possible worlds that are accessible from the current world and some of these possible worlds have a successful negotiation state. We describe two bilateral negotiation models, a generic one and an expanded one.

Generic Bilateral Negotiation

Some of the abstract states in a bilateral negotiation are:

offered: In this state, the subject of negotiation cannot be changed and can only be agreed to or rejected. The offering party is committed to the offer.

proposed: proposed is a sub-state of offered and may be agreed to or the subject can be further changed by transition to a requested state.

requested: This state allows a respondent to change the subject of a negotiation and the other party to respond with propose or offer or suggest.

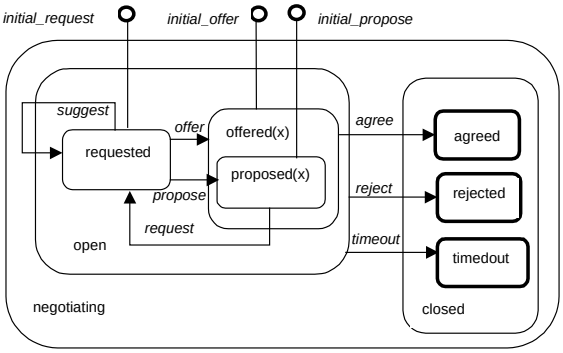

Figure3: Generic Bilateral Negotiation Model

A user initiates the negotiation process in either a proposed, requested or offered state on the negotiation subject. The bilateral negotiation model, shown in figure 3, is derived from (OSM SARL 1998). It incorporates corrections of the ambiguities emerging from our methods.

The logical theory of the model:
negotiating ↔ one_of (**[** *open, closed* **]**)
open ↔ one_of (**[** *requested, offered* **]**)
closed ↔ one_of (**[** *agreed, rejected, timeout* **]**)
proposed(*X*) → *offered*(*X*)

Thus the negotiating state condition holds if and only if it is an open or closed negotiation state. Similarly it is in an open state only if it is a requested or offered state. The closed state consists of the *agreed*, *rejected* and *timedout* substates. If agents X and Y are involved in such a bilateral negotiation, the primitive actions leading to the states are:

¬ *negotiating* ↔ [X.*initial_request*] *requested*
¬ *negotiating* ↔ [X.*initial_offer*] *offered*(*X*)
¬ *negotiating* ↔ [X.*initial_propose*] *proposed*(*X*)
} 3 entry points

offered(*X*) ↔ [Y.*agree*] *agreed* ∧ ¬(*X*= *Y*).
requested ↔ *one_of* (**[** [*Y.offer*] *offered*(*Y*) , [*Y.propose*] *proposed*(*Y*), [Y.*suggest*] *requested* **]**) .
proposed(*X*) ↔ [*Y.request*] *requested* ∧ ¬(*X*=*Y*).
open ↔ *one_of* (**[** [(*X* ∪ *Y*).*reject*] *rejected* , [*timeout*] *timedout* , [*offered*(*X*)? *Y.agree*] *agreed* **]**) ∧ ¬(*X*=*Y*).

On entering the negotiation, an *initial_request* action by agent X leads to the *requested* state. There are three transitions that can be performed by Y from this requested state – an *offer* leading to the *offered* state, a *propose* leading to the *proposed* state and the *suggest* leading to the *requested* state. From state *offered* triggered by X, an *agree* action by Y leads to the *agreed* state. From the open state, a *reject* by both agents or a *timeout* action may occur.

Expanded Bilateral Negotiation Model
In some cases we may need a richer model of negotiation that allows more interactions and allows the parties to be able to do more than restricted primary actions. In an expanded model, we include capabilities for the agents to seek answers to questions and to use argumentation and persuasion to influence agent beliefs and to achieve their goals. Not everything here has a simple logical theory. In the *investigating* state, expert advice may be asked and given by the involved agents or by third parties. This state can co-exist at the same time as the state reached in the enclosing negotiation and finally exits to continue the initial negotiation. One party can partly-reject an issue it dislikes but can still continue negotiating, with both parties keeping in mind not to use the rejected issue again.

With relation to a business to business negotiation scenario, we may need to be able to break a negotiation into sub-negotiations that are solved before going back to the encompassing negotiation. This allows the nesting of negotiations and the ability to change dimension in negotiating, first dealing with subgoals and subsequently negotiating about the main issue. When one of the agents needs to delay the negotiation process, the negotiation state changes to pending. The state chart in figure 4 allows for simultaneous negotiation processes to take place, where an agent A negotiates with agent B

on multiple processes giving rise to several negotiation threads in a negotiable state. Or agent A may also negotiate with other agents in parallel with agent B. Agent A can then use the proceedings of its other negotiations to influence the negotiation with agent B.

This negotiation model is an attempt to show richer interactions between agents as compared to a generic bilateral negotiation. A formal theory for this expanded negotiation model can be expressed as before but there are several open issues that still need to be clarified in this model.

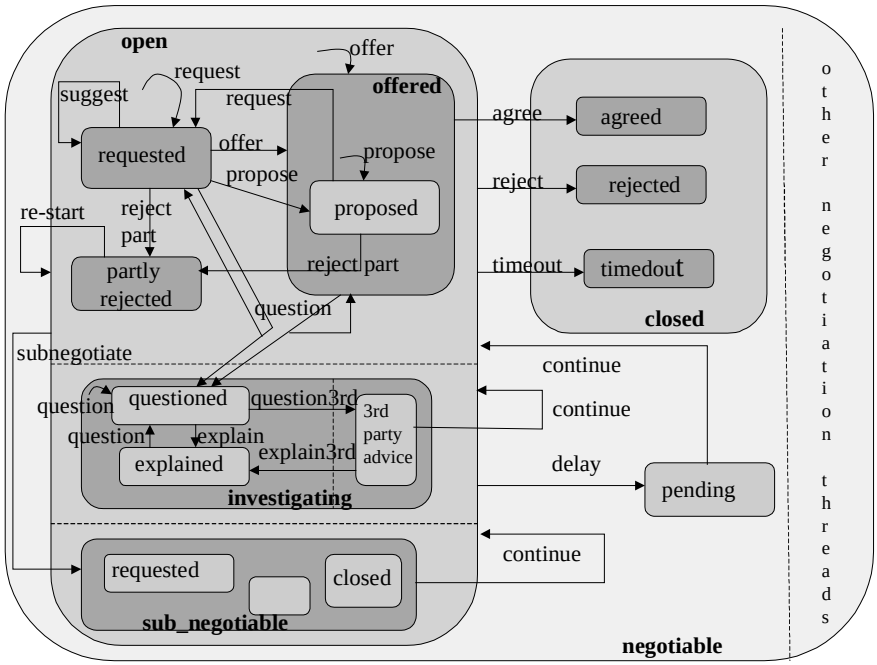

Figure 4: Expanded Bilateral Negotiation Model

5.2 Other Negotiation Models – Auction, Multi-lateral and Promissory

Auction

A popular model of negotiation is auction of which there are several protocols for bidding and disclosure, including English auctions, Dutch auctions, sealed bid auctions and Vickrey. Auctions specify the boundaries of the negotiation process and therefore are a quite simple form of negotiation to implement. There are automated, semi-automated and manual auctions currently online. Auctions have been implemented due to the known constraints of the negotiation process. The bargaining is restricted to price and the seller's strategy to assign awards is made known. Figure 5 shows a basic model for English auctions.

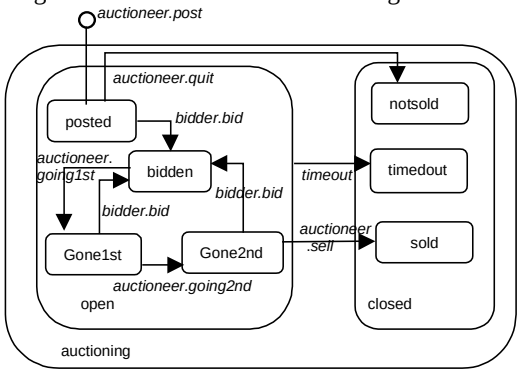

Figure 5: Simple English Auctioning Model

Multilateral Negotiation

A multilateral negotiation model defines the protocol relevant for submitting motions in a quorum, for seconding and amending these motions, and subsequent voting within a community of two or more agents. A participant initiates the process by raising a motion on a subject. The *pending* state follows under which the initiator can withdraw the motion or the motion may time out leading to a *withdrawn* state or another participant can second the motion leading to a *seconded* state. In the *seconded* state, the countdown to a vote timeout is activated. In this same state, any user may invoke the amend transition to change the subject of negotiation or may call a transition to the *voting* state leading to an *agreed* or *rejected* state. Other models for multilateral negotiation provide for one to many parties and many to many parties, as in protocols for channel communication.

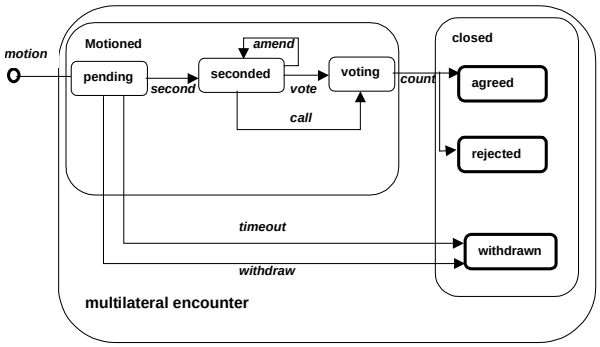

Figure 6: Multilateral Negotiation Model

Promissory Negotiation

Promissory models specify promises to execute a process, the right of one party to call on a promise and the obligation to fulfil a promise, for example by engaging into contract or by fulfilling the contract. The promissory negotiation process is initialised when a provider either makes a promise resulting in the consumer having a *right* to call upon the promise, or when the provider makes a commit which results in a *pending obligation* of the provider. In the *promised* state, the consumer can call upon the promise and request a transition for his right to be fulfilled. This leads to a *pending obligation* of the provider. The right of the consumer can timeout and lead to the terminal *expired* state. From the pending state, either the provider fulfils her/his obligation by calling the fulfil compound transition or the *pending* state times out and leads to the *overdue* sub-state. The fulfil transition launches a bilateral negotiation between the two agents and the result of the bilateral negotiation determines the result of the promissory negotiation. A waive transition may be invoked by either the consumer or the provider.

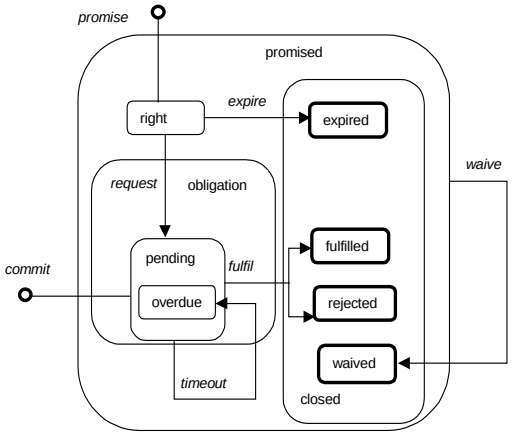

Figure 7: Promissory Model

6 Finding a Path for Negotiation

The expanded bilateral, auctioning, promissory and multilateral negotiation models can be expressed in dynamic logic in the same way as for the generic bilateral negotiation model. Negotiating agents can use the formal

theories as inference engines in order to comply to a negotiation protocol. From such specification, the abstract rule of a particular negotiation protocol emerges. For example, in the bilateral negotiation, where the most abstract state is *negotiating*, the abstract process *negotiate* represents all the paths leading to a closed state of negotiation. This can be represented as: $\neg$ *negotiating* $\leftrightarrow$ [*negotiate*] *closed*

The *negotiate* process is the set of paths that the retailer and the customer can take to reach their goal state. Negotiating agents must use a subpath of the abstract path *negotiate* such that from an open state, entered by a *initial_request, initial_offer* or *initial_propose*, they seek an *agreed* state that satisfies their goal. Thus $\exists$ *p:path* <*open*? *p* > *agreed* asserts that from *open*, there is a path *p* leading to *agreed*, a possible outcome of *negotiate*. The paths may themselves be represented in dynamic logic, deduced from the logical theory of the negotiation paths, or may form part of the library of an agent's plans of actions. A rational agent will seek to derive a path that will lead it to an *agreed* state. An abstract negotiating state like *agreed* has significance for an agent because the *agreed* state represents a subgoal of the agent, moving it closer to its greater goal.

Referring back to the scenario of the shopping process in section 3, the stages of offering and accepting a payment mode can follow a bilateral negotiation model. The retailer and customer will use an instance of the abstract process *negotiate*. An example of a path for {*r:retailer,c:customer*}.*negotiate_payment* is:
r.propose_mode(*m*); (*c.request_mode*(*n*) ;(*r.propose_mode*(*x*) *; c.request_mode*(*y*)) * ; *r.offer_mode*(*z*))$^{\leq 1}$; (*c.accept_mode*(*z*) | *c.reject_mode*(*z*)).

6.1 Gaining the Maximum Utility

Although, each agent has its own goal – to make a profit and to buy a service at a certain price respectively – when making a transaction, two agents will negotiate and converge to some final set of values that satisfies both goals. A rational agent will not only seek to achieve its goal of finding an *agreed* negotiation state. It will also try to achieve such a goal in a manner that will be most cost effective for it. Define the utility of a goal as the difference between the worth of the goal and the cost of attaining the goal, an agent's reservation cost as the maximum cost that it is willing to pay to achieve its goal. We can now think of each agent as having a partially ordered set of goal utility values, where different agents may have differing utility orderings and mutual knowledge of each other's utility values is incomplete. The agent goal may be the maximum utility, without being more than the reservation costs. For example, a customer agent has a preferred goal of acquiring a service at a minimum price and to this end will negotiate with the subgoal of simultaneously closing the negotiation in an *agreed* state at the minimum cost but with also the highest worth to him. An agent can thus have a library of plans containing a set of paths that enable it to interact with and respond to other agents with the aim of satisfying its goal.

The agent has also to comply with the negotiation protocol and the external and internal constraints of the environment. These plans can be constructed by analysing the agent's individual beliefs and the common beliefs of the agents about the negotiation and their goals and constraints. The individual beliefs of an agent may include the costs, worths and utilities of the states of the negotiation as well as the agent's library of plans, time constraints affecting negotiation and beliefs about other agents.

As shown in section 4, an agent can infer the set of the paths leading to a particularly state when it is given the logical theory of the state transitions and conditions. Likewise, given the relations between states and substates and the primitive actions leading to these states from negotiation models, the paths towards an *agreed* state can be derived. We have seen that using the algebra of paths, a set of paths can be composed in terms of other paths and edges with $\cup$, ;, etc. For deterministic behaviour, a utility value can be assigned to each path. The agent can interpret the utility value of paths to influence its response. We may, for example, interpret

- *utility*(*path_a* ; *path_b*) as *utility*(*path_a*) + *utility*(*path_b*).
- *utility*(*path_a* $\cup$ *path_b*) as *minimum or maximum utility*(*path_a, path_b*) depending on the goal of the agent.
- *utility*(ϵ), the null path, as utility of value 0.
- *utility*(ϕ), the empty set of paths, as utility of value $+\infty$ or $-\infty$ depending on the goal.
- *utility*(*a**) as *minimum* or *maximum utility*(ϵ , *utility*(*a*) + *utility*(*a**)), again depending on the goal.

7 Related Work

Parsons, Sierra and Jennings (1998) presents a formal model of argumentation-based reasoning and negotiation for autonomous agents. They show how agents can construct arguments to help guide their proposals, how agents can critique proposals and how agents can exchange arguments to help guide their problem solving behaviour towards acceptable solutions. In an article by Oliveira and Rocha (1999), multiple criteria and distributed constraint formalisms are used to construct a negotiation protocol and to select organisations to become members of a virtual organisation. Their multi-agent architecture has a coordinator agent to help in the virtual organisation formation process. The agents also use learning algorithms to adapt to changes in the environment. Negotiations in the CASBA system (Vetter and Pitsch 1999) are parallel and follow pre-defined rule-based strategies which can be adapted by the user and evolve through learning methods. Agents agree to only one offer and must notify each other if they agree to the last offer made by the other agent. Then both agents submit their agreement information to an administrator agent for checking the deal. Four types of auctions are supported in this system and agreements are implied by posting the service and by bidding. For a more game theoretic analysis see Rosenchein and Zlotkin 1994.

8 Conclusion

Information and communication technologies have given birth to Electronic Commerce. However, to escalate from web surfing to interactive online trading, an open architecture needs to support requirements amongst which automated negotiation is a major one. One can think of scenarios where user agents will have goals to be satisfied and will negotiate on behalf of their owners with other agents on a market-place. To do so, the agents need to have negotiation models which dictate the public protocols for effective cooperation between the agents and enable eventual convergence towards a state that satisfy the agents goals. In these models, an agent joins an instance of negotiation by establishing a member role, interacts in the negotiation by operations to apply and invoke transitions and commands on states. Models of negotiation can be further customised following the patterns presented in this paper. Even if their private negotiation models differ, two agents must agree to a common negotiation model. This common negotiation model must form part of the belief of each agent and is a common belief. Thus the mutual beliefs of the agents include the negotiation protocol to be complied with and the current states of the negotiation and the goals.

From an attempt to represent a scenario of negotiation in a shopping process, we have exposed a need for notation whereby processes and states are both represented. Using dynamic logic, we have illustrated the construction of a logical theory for a shopping process and similar interactions. We have found that this theory construction process can expose errors in less formal definitions. At an abstract level, from an initial state, an agent can follow a set of execution paths to perform the negotiation process and end in a state that satisfies its goal. Stored paths form the plans of the agent. When constrained by the model of negotiation an agent should still in principle find a path leading to desirable states. The individual goals and intentions of the agent will determine the chosen paths. When a provider and a customer negotiate there is already some common knowledge that each agent has about each other. The customer knows the provider's goal is to sell his goods. This common knowledge of the other's intentions also plays some part in the choice of the optimal path. As the negotiation progresses, each agent can build up more beliefs about the other agents. With its beliefs and goals, an agent can interpret its logical analysis of paths to seek the best way to satisfy its goal.

Acknowledgements

This work has been done in the context of a the OSM+ project (ACTS 211). We wish to acknowledge constructive comments from Stephen McConnell on an earlier draft and from reviewers of an AAAI workshop. Shamimabi Paurobally also wishes to acknowledge the support of the trustees of the Beit Fellowships.

9 References

Cunningham, J.; Paurobally, S.; Diacakis, A.; Lorenzen, L.; Gross, G.; and McConnell, M. 1998. Satisfying Requirements for Electronic Commerce. *In Proceedings of Trends in Electronic Commerce 98, Hamburg.*

OSM SARL. 1998. *Negotiation Facility*. http://www.omg.org/pub/ec/99-03-01.

Rosenschein, J., and Zlotkin, G. 1994. Rules of encounter, Designing Conventions for Automated Negotiation among Computers. *MIT Press.*

Goldblatt, R. 1987. Logics of time and computation. *CSLI*

Jackson, M. 1975. Structured Design. *Academic Press.*

Parsons S., Sierra C., Jennings N., Agents that reason and negotiate by arguing, 1998
Backhouse, R., and Carre, B. 1973. Regular Algebra Applied to Path-finding Problems. *In J. Inst. Maths Applics. 1975, Volume 15, pages 161-186.*
Milner, R., 1989. Communication and Concurrency, *Prentice Hall.*
Oliveira, E., Rocha A., 1999. Agents advanced features for negotiation in Electronic Commerce and Virtual Organisations formation process, *this volume*.
Vetter, M., Pitsch, S., 1999, Towards a Flexible Trading Process over the Internet, *this volume*.