Contract Representation for Run-time Monitoring and Enforcement

E-mails:

Carlos Molina-Jimenez <Carlos.Molina@ncl.ac.uk>
Santosh Shrivastava <Santosh.Shrivastava@ncl.ac.uk>
Ellis Solaiman <Ellis.Solaiman@ncl.ac.uk>
John Warne <J.P.Warne@ncl.ac.uk>

School of Computing Science, University of Newcastle upon Tyne,
Newcastle upon Tyne, NE1 7RU, England

Abstract

Converting a conventional contract into an electronic equivalent that can be executed and enforced by computers is a challenging task. The difficulties are caused by the ambiguities that the original human-oriented text is likely to contain. The conversion process involves the conversion of the original text into mathematical notation. In this paper we discuss how standard conventional contracts can be described by means of Finite State Machines (FSMs). This mathematical description helps eliminate ambiguities from the original text before the contract is coded into a computer program. The paper describes how to map the rights and obligations extracted from the clauses of the contract into the states, transition and output functions, and input and output symbols of a FSM. The FSM guarantees that the clauses stipulated in the contract are observed when the program that implements the contract is executed. Also, the paper suggests a middleware service required for the enactment of the contract represented as a FSM.

Keywords: Contract, electronic contract, finite state machine, contract representation, contract monitoring, contract enforcement, contract implementation.

Contract Representation for Run-time Monitoring and Enforcement

Carlos Molina-Jimenez, Santosh Shrivastava, Ellis Solaiman and John Warne
School of Computing Science, University of Newcastle upon Tyne,
Newcastle upon Tyne, NE1 7RU, England

Abstract

Converting a conventional contract into an electronic equivalent that can be executed and enforced by computers is a challenging task. The difficulties are caused by the ambiguities that the original human-oriented text is likely to contain. The conversion process involves the conversion of the original text into mathematical notation. In this paper we discuss how standard conventional contracts can be described by means of Finite State Machines (FSMs). This mathematical description helps eliminate ambiguities from the original text before the contract is coded into a computer program. The paper describes how to map the rights and obligations extracted from the clauses of the contract into the states, transition and output functions, and input and output symbols of a FSM. The FSM guarantees that the clauses stipulated in the contract are observed when the program that implements the contract is executed. Also, the paper suggests a middleware service required for the enactment of the contract represented as a FSM.

1 Introduction

The concept and the use of contracts are not new to today's society. Legal contracts can be traced back to ancient times [8]. There are records that indicate that legal contracts were used by the Mesopotamians circa 2300-428 BC for selling and purchasing slaves, land and crops. Also, there is evidence that shows that the Mesopotamians knew how to write and sign legal contracts for hiring services such as houses and workers; and for establishing partnerships between two or more land-owners. Since contracts have been used for a long time, we humans know how to write (in English for example), interpret and execute a contract. We can be sure that managers of the contractual parties will know how to read the English text and make sense of it. Once the contracting parties have agreed on the clauses to include in a contract, the composition of the English text of the contract does not represent a great challenge. Consequently, contracts were not a subject of research interest until the early 90s, when it first appeared that the Internet might offer a viable alternative for conducting business. As in the conventional world, contracts are crucial for conducting commercial transactions in the electronic world. Without a contract, the legal position of two or more enterprises trading over the Internet is uncertain. Even a transaction as simple as buying an electronic book requires a contract. While contracts in the conventional world are well understood, contracts in the electronic world are not yet well understood.

To prevent confusion with the terminology we use in this paper, it is probably advisable to begin by agreeing on the following terms: in this paper we use the term

contract to refer to a conventional contract written in a natural language. On the other hand, we use the term *e-contract* to refer to computer-oriented contracts, that is, to contracts that are interpreted and executed by computer. Although everybody agrees that without e-contracts e-business will never reach its full potential, e-contract related technology is not yet mature. For example, we do not know how a standard contract written by humans and for humans, can be converted into an e-contract. That is, into a computer program that, when executed, enforces what is stipulated in the English text contract. We have been examining contracts and attempting to convert them into their equivalent e-contracts. Not surprisingly, we have found that the conversion of contracts into e-contracts is not a trivial process.

We have found that Finite State Machines (FSMs) are quite adequate for modelling e-contracts. One aim of this paper is to discuss how conventional business contracts written in English can be described, by means of FSMs. The advantage of having a contract expressed by FSMs or other mathematical model is that the model can be checked for correctness properties first. Once the FSM model is free of ambiguities that are likely to exist in the original contract, we can proceed with writing the additional code needed to implement the contract. When executed, this program (or set of programs) will enact the contract, monitor its behaviour and enforce its clauses. The second aim of this paper is to describe the run-time infrastructure for monitoring and enforcement of e-contracts.

The scenario of the problem we are investigating can be described as follows: given a standard contract written in English, which for example can be downloaded from the Internet or from another source, how can the text be converted it into a FSM. This means that we are not investigating how to negotiate contracts or e-contracts over the Internet; we assume that the contract already exists. Our job is to express it in FSMs, possibly, after editing the original text to correct ambiguities.

2 Contracts and e-contracts

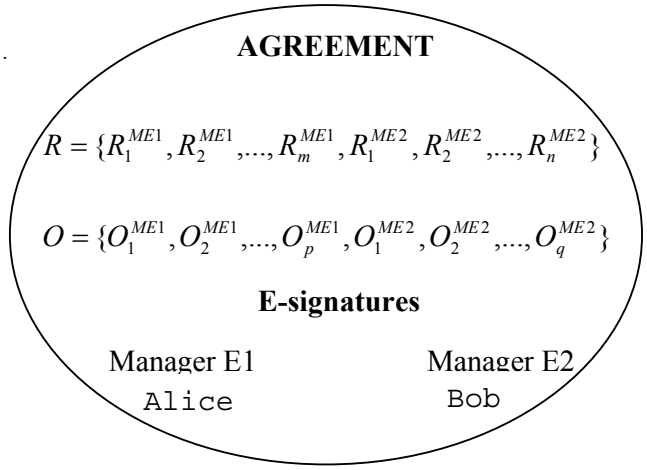
A *contract* can be defined as a paper document that stipulates that the signatories (two or more) agree to observe the clauses stipulated in the document. Each entry in the document is called a *term* or a *clause*. Moreover, it is common practice to specify what *role* (manager, accountant, supervisor, etc.) each of the signatories (Alice, Bob, Doug, etc.) play within their respective enterprises.

An *e-contract* is an electronic version of a conventional contract. It is an electronic document that stipulates that the signing entities (two or more) agree to observe clauses stipulated in the document.

As can be seen from the definitions of conventional and electronic contracts, the main idea behind conventional and electronic contracts is the same: the two of them regulate the interaction between two or more trading parties. However, in spite of the close similarities, there is a crucial difference between the two kinds of contract. A conventional contract is human oriented, whereas an e-contract is computer-oriented. Consequently, the former tolerates ambiguities, the latter does not.

2.1 Rights and obligations

The clauses of a contract stipulate how the signing parties are expected to behave. In other words, they list the rights and obligations of each signing party. It should be possible to precisely extract a set of rights and a set of obligations from the clauses of a given contract. If this is true, an e-contract with two contracting parties can be represented as shown in Fig.1.



E1,E2—Enterprises, R_i—Right, O_i—Obligation
ME1—Manager of E1, ME2—Manager of E2

Fig. 1. Main elements of an e-contract.

Note that for the sake of simplicity, in this paper we discuss examples of contracts with only two contracting parties. However, all our concepts, models, and examples can be generalised to $n \geq 2$ parties as long as n is finite.

Let us assume that in Fig.1 E1 is managed by Alice and that E1 is interested in purchasing some items from E2. Similarly, Bob is the manager of E2 which is an enterprise interested in supplying items to E1. The contract shown in this figure has been signed by Alice as the manager of enterprise E1 after agreeing with Bob that she will observe m rights, $R = \{R_1^{ME1}, R_2^{ME1}, ..., R_m^{ME1}\}$ and p obligations, $O = \{O_1^{ME1}, O_2^{ME1}, ..., O_p^{ME1}\}$. On the other hand, Bob signed that he, as the manager of enterprise E2, would observe n rights, $R = \{R_1^{ME2}, R_2^{ME2}, ..., R_n^{ME2}\}$ and q obligations, $O = \{O_1^{ME2}, O_2^{ME2}, ..., O_q^{ME2}\}$.

In this discussion, a *right* can be defined as an authorization to do something. Because it is only an authorization, a right **may or may not** be exercised. In the context of the execution of an e-contract, a right is an authorization to perform an operation that will

affect the behaviour of the execution of the e-contract. For example, the signed e-contract of Fig.1 can stipulate that Alice, as a manager of E1, has the right to send an offer to sell to Bob, the manager of E2. Because this is a right, it is up to Alice to send or not to send the offer to Bob; Bob will not be disappointed if he does not receive the offer.

Similarly, an *obligation* can be defined as a duty that **must** be performed. In the context of the execution of an e-contract, an obligation is a duty to perform an operation that will affect the behaviour of the execution of the e-contract. A failure to perform such a duty means a breach of the e-contract. For example, the text of the e-contract shown in Fig.1 might stipulate that upon receiving an offer to sell from Alice, Bob has the obligation to reply to her with an *OfferAccepted* or *OfferRejected* message.

Note that the execution of a right or an obligation such as *SendOfferAccepted* or *SendOfferRejected* will, at a lower level of abstraction, demand access to one or more objects such as files, databases and printers. A question that arises here is whether Alice and Bob have the right to access the objects affected by their operations. This is an issue of authentication and can be left out of the discussion while we talk at the level of rights and obligations. At this level we can assume that the persons that execute the e-contract are granted permission to access the objects they need. Our view is that at object level, rights to access resources can be implemented on top of a Role-Based Access Control like the standard RBAC[7].

Note that in our e-contract each right and obligation is given a name; naming right and obligations is crucial in our understanding of e-contracts. Being able to name each right and obligation individually means that we can identify, monitor and enforce each of them at run-time, that is, when the contract is enacted.

3 E-contracts, communication protocols and FSMs

From books on the theory of Computer Science (see for example [12]) we know that, formally, a finite state machine M is defined as the quintuple $[S, I, Z, \delta, \lambda]$, where $S = \{s_1, s_2, ..., s_m\}$, $I = \{i_1, i_2, ..., i_n\}$ and $Z = \{z_1, z_2, ..., z_p\}$ are finite nonempty sets of states, input symbols and output symbols, respectively. $\delta : S \times I \rightarrow S$ is the transition function and $\lambda : S \times I \rightarrow Z$ is the output function.

Informally, M describes an abstract system that stays in a given state until it receives an external stimulus. When such stimulus is received, the system reacts by doing something (for example, sending an output signal) and then moves to a different state. Note that *do something* might mean do nothing in some circumstances and that the new state is not necessarily different from the previous. The behaviour of this abstract system is deterministic. The quintuple $[S, I, Z, \delta, \lambda]$ unambiguously defines what to do and where to go next.

Because of their high level of abstraction, FSMs are used to describe and model a great variety of systems. In particular, the computer science community has gained a great deal of experience in the use of FSMs for describing communication protocols, and built several tools for validating such protocols. For example, Spin [9] is a well known protocol validator among the academic community.

We have introduced communication protocols into the discussion about e-contracts for a valid reason: we strongly argue that from the point of view of the interaction and synchronisation between the parties involved, e-contracts are equivalent to communication protocols. We claim that e-contracts, as communication protocols are, can be precisely abstracted by FSMs. The advantage of looking at contracts as FSMs is that we can put into practice all the existing machinery that was originally developed for studying communication protocols. For instance, we can resort to Spin to validate an e-contract before converting it into the actual computer program that will enact it. The goal of a validation process is to analyze what is known as the correctness properties of the system. In other words, the essence of the validation is to discover, at an early stage, whether the execution of an e-contract takes the contracting parties into unacceptable situations. Among other things, validating the FSM model of an e-contract should reveal the existence of states (conditions in the e-contract) that are not reachable, that is, states for which there is no path from the initial state. If one of these unreachable states represents the receipt of the goods the situation would be unacceptable and the e-contract would need to be re-written. In the same order, the validator should show that at some point, the two contracting parties reach a final state (contract deal for example) instead of being left in a transient state for ever. To mention another example, the validation should reveal whether the e-contract allows purchasers to receive goods before paying for them.

A question that naturally arises at this point is how the rights and obligations of an e-contract can be expressed in a FSM.

4 Description of e-contracts

In this section we will discuss how the components of an e-contract can be mapped into the components of a FSM.

4.1 Mapping conditions, rights and obligations into a FSM

At the level of rights and obligations an e-contract is often more easily understood as a set of FSMs, one for each contracting party. So, from our example in Fig.1, we have one FSM for the purchaser and one FSM for the supplier, FSM_P, FSM_S respectively.

The physical location of each FSM is irrelevant to the functionality of the contract and is decided at the time of implementation. For the moment let us assume that FSM_P is located within Alice's enterprise and FSM_S is located within Bob's enterprise. To enact the e-contract these two FSMs must share a common communication channel to interact with each other, that is, the output of FSM_P is somehow connected to the input of FSM_s and vice versa. We will now discuss how the rights and obligations stipulated in an e-contract can be mapped into the FSMs.

It is easy to reason about the operations of an e-contract, with the following general syntax in mind:

if $event_1$ *&* $condition_q = true$
perform $operation_1$ *and switch to* $state_1$

else if $event_2$ *&* $condition_q$ *= true*
perform $operation_2$ *and switch to* $state_2$
… … …
else if $event_m$ *&* $condition_q$*= true*
perform $operation_m$ *and switch to* $state_m$

This syntax expresses the idea that, at some point an e-contract can be at any of *n* possible conditions ($condition_1$, $condition_2$,…,$condition_n$). If the e-contract is in a given $condition_q$ (for example, *WatingForOffer*), there is a finite and well defined set of events ($event_1$, $event_2$, …,$event_m$) that can affect the future behaviour of the contract. The occurrence of $event_i$ determines what objects (variables, files, database, etc.) within the system change their values, that is, the event determines to which new condition the systems switches. Similarly, there is a finite and well defined set of operations ($operation_1$, $operation_2$,…, $operation_m$) that can be executed when the system is in $condition_q$. The $event_i$ determines the operation to be executed.

Bearing in mind the discussion presented in Section 3, we argue that in terms of FSMs, the set of conditions of the general syntax presented above, can be mapped into the set S of states of a FSM. Similarly, the set of events can be mapped into the set I of input symbols of the FSM. In the same order, the set of operations can be mapped into the set Z of output symbols of the FSM. Finally, we can map the set of switches to the next condition into the transitional function δ. It is important to bear in mind that the operation *donothing* is a valid operation. In this discussion we represent it with the symbol ε.

Thus, in terms of FSMs, we can express the above syntax as shown in Fig.2, where *e* and *o* stand for event and operation, respectively.

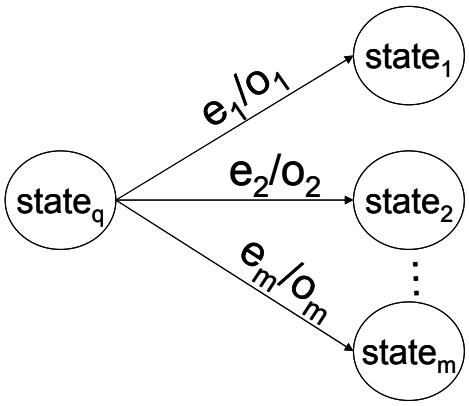


Fig. 2. Mapping of events, conditions and operations of an e-contract into a FSM state.

4.2 Description of a simple e-contract using FSMs

To show what rights and obligations look like, we will discuss a very simple example of a e-contract for offering and purchasing goods remotely, for example, over the Internet. As an attentive reader will notice, the e-contract has serious ambiguities; this

will be discussed in Section 5.2. The e-contract which is signed by a purchaser and a supplier contain, among other data, the following clauses:

1. Offering
1.1 The supplier may use his discretion to send offers to the purchaser.
1.2 The purchaser is entitled to accept or reject the offer, but he shall notify his decision to the supplier.
2. Commencement and completion
2.1 The contract shall start immediately upon signature.
2.2 The purchaser and the supplier shall terminate the e-contract immediately after reaching a deal for buying an item.

From this English text contract clause we can extract the sets of rights and obligations for the purchaser and the supplier and express them in terms of operations for FSMs.

Purchaser's rights:

R_1^P : SendAccepted -- right to accept offers.

R_2^P : SendRejected -- right to reject offers.

Purchaser's obligations:

O_1^P : StartEcontract -- obligation to start the e-contract.

O_2^P : SendAccepted or SendRejected -- obligation to reply to offers.

O_3^P : EndEcontract -- obligation to terminate the e-contract.

Supplier's rights:

R_1^s : SendOffer -- right to send offers.

Supplier's obligations:

O_1^S : StartEcontract -- obligation to start the e-contract.

O_2^S : EndEcontract -- obligation to terminate the e-contract.

To be consistent with the notation in Fig.1 we now specify the sets of rights and obligations: $R = \{R_1^P, R_2^P, R_1^S\}$ and $O = \{O_1^P, O_2^P, O_3^P, O_1^S, O_2^S\}$. We show how the sets R and O are mapped into FSMs in Fig. 3.

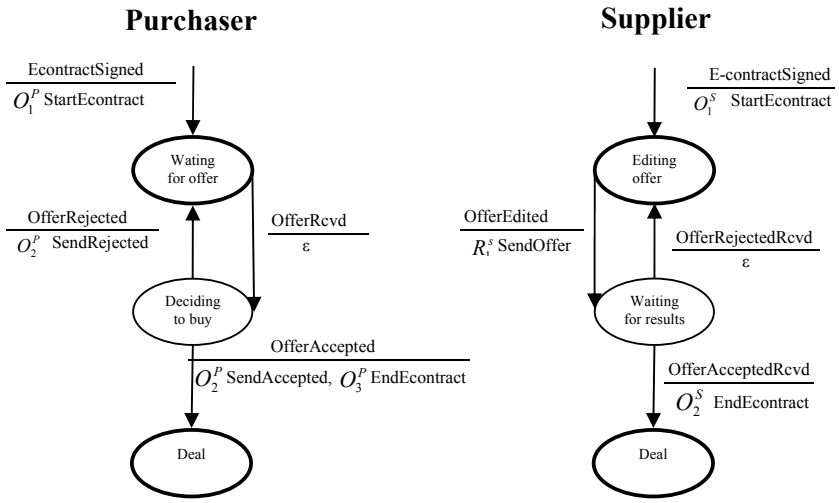


Fig. 3. FSM Representation of an ambiguous e-contract for the purchase of goods.

As can be appreciated from Fig.3, we have used, two FSMs to precisely describe the English text contract of our example. However, as they are, the two FSMs of Fig. 3 only describe the behaviour of the two contracting parties, they do not monitor or enforce it. As you can see, the elements of the sets of rights and obligations are also shown in the figure.

5 Monitoring and enforcement of e-contracts

During the execution of an e-contract rights and obligations are triggered by local and remote events. In this section we will show how two FSMs trigger rights and obligations on each other.

5.1 Invocation of rights and obligations

To reason about how the contractual rights and obligations can be monitored and enforced by a FSM, it is useful to look at the rights and obligations a contracting party has at a given state of the execution of the e-contract. In terms of FSMs, this is equivalent to looking at the set of operations that can be executed when the FSM of the contractual party is at $state_q$. It is useful to classify this set into two subsets: the subset of operations the owner of the FSM has the right to perform and the subset of operations that person has the obligation to perform, $\{o_1, o_2, ..., o_m\}$ and $\{o_{m+1}, o_{m+2}, ..., o_p\}$, respectively.

To illustrate how the rights and obligations are triggered we will examine Fig.4. This figure shows a snapshot of the two FSMs that model an e-contract for the purchase and supply of e-goods.

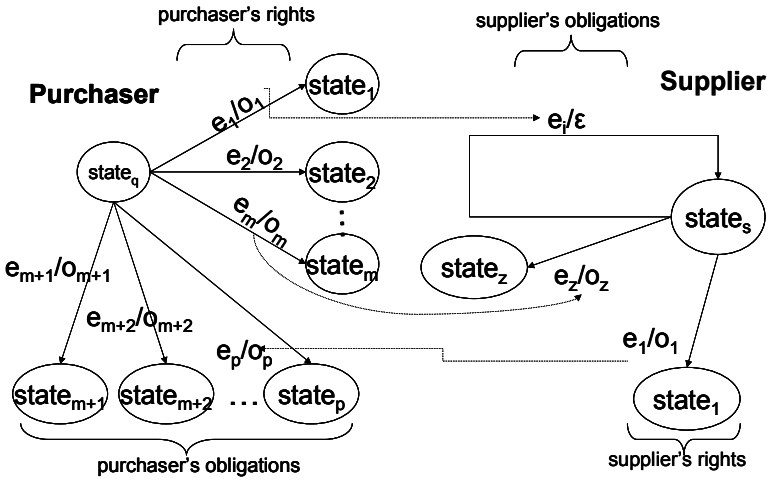


Fig. 4. Interaction of two FSMs by means of rights and obligations.

Let us say, the execution of the e-contract at the purchaser's side is at state $state_q$.
As it can be seen from the figure, the rights and obligations the purchaser has when his FSM is at $state_q$ can be mapped into the sets $\{o_1, o_2, ..., o_m\}$ and $\{o_{m+1}, o_{m+2}, ..., o_p\}$, respectively.
Executing an operation from the subset $\{o_1, o_2, ..., o_m\}$ means exercising a right given by the e-contract. Since each operation o_i is paired to an event e_i, the operation o_i can be executed only after the occurrence of e_i. How does event e_i occur?

Event e_i can be triggered internally within the purchaser's enterprise or externally, say for example, within the supplier's enterprise. Since the execution of operation o_i is optional, the event of e_i might be deliberately triggered by the purchaser (for example, I wish to send a purchase order). Also, it can be the result of an unavoidable situation within the purchaser's enterprise (for example, the mainframe computer is non-functional) or it can be triggered by a message received from the supplier (for example, *would you like to buy this item?*).

Executing an operation o_{m+i} from subset $\{o_{m+1}, o_{m+2}, ..., o_p\}$ means complying with the contractual obligations the purchaser has when his FSM is at $state_q$. As with the rightful operations, the obligatory operations are paired to events which are triggered internally, or externally.

It is important to understand that exercising a right at one side of the contract might or might not have an effect at the other side. This depends on what the text of the e-contract stipulates. The execution of operation o_i at the purchaser's side might trigger a right, an obligation, or nothing at the supplier's side. By nothing we mean that the supplier's is not notified about the execution of the operation o_i at the purchaser's side.

Similarly, the execution of an obligatory operation o_{m+i} from the subset $\{o_{m+1}, o_{m+2}, ..., o_p\}$ might trigger a right, an obligation or nothing at the supplier's side.

The dashed line pointing from the pair e_1 / o_1 at the supplier's side to the pair e_p / o_p at the purchaser's side implies that at state$_s$ the supplier has the right to execute the operation o_1. The English text of the contract stipulates that the purchaser (being at state$_{q)}$ has the obligation to execute operation o_p as a response.

Similarly, the dashed line pointing from e_m / o_m to e_z / o_z shows that at state$_q$ the purchaser has the right to execute the operation o_m. As a response to this operation, the supplier has the obligation to execute the operation o_z. The dashed line from e_1 / o_1 to e_1 / ε shows that the purchaser's has the right to execute the operation o_1. However, the execution of such operation demands nothing at the supplier's side; the supplier is breaching the e-contract.

To show how these ideas can be used in practice, we will apply them now to the example of e-contract discussed in Section 4.2.

5.2 Description, monitoring and enforcement of an e-contract

In practice, it is likely that contracts will be written by lawyers and then passed on to technical people to convert the original English text into a computer program that will monitor and enforce what the contract stipulates.

From our own experience we have learnt that the first difficulty the technical person faces in this e-situation is the ambiguities that the English text contract is likely to have. The standard contract discussed in this section is not an exception. Although it contains only eight lines and looks correct at first glance, it has a serious ambiguity.
The contract text does not specify the time for sending the offer. Neither does it specify the time for sending the notification about rejecting or accepting the offer.

These two omissions render the English text contract difficult to convert into a useful e-contract. It is true that the e-contract can still be implemented and enacted but the purchaser's FSM will hang silently until the supplier decides to send an offer. If for some reason the supplier forgets to send his offer, the two FSMs will hang silently forever or until the purchaser or the supplier use another channel (a telephone, for example) to investigate the problem. Telephone calls are intensively used for clarifying situations in conventional business, however, in e-contracts they are not acceptable because they are exactly what e-contracts are meant to prevent.

To be consistent with our arguments we show the English text of the example discussed in Section 4.2 after editing it to correct the ambiguities it had.

1. Offer

1.1 The supplier may use his discretion to send offers to the purchaser.

1.2 If no offer is sent within seven days after the signature of the e- contract, the e-contract shall be terminated.

1.3 The purchaser is entitled to accept or reject the offer, but he shall notify his decision to the supplier within five days after the receipt of the offer.

2. Breaches, reminders and disputes

2.1 If the purchaser fails to reply to the purchaser's offer within two days, the supplier shall terminate the contract. This will be regarded as an abnormal termination of the e- contract.

2.2 An abnormal termination of the contract entitles the purchaser and the supplier to call each other to resolve disputes.

3. Commencement and completion

3.1 The contract shall start immediately after signing it.

3.2 The purchaser and the supplier shall terminate the e-contract immediately after reaching a deal for buying a good.

We will not extract the set of rights and obligations from the text of the contract because, apart from minor changes, they look the same as the ones listed in Section 4.2. We will show only the FSMs that describe, monitor and enforce the e-contract.

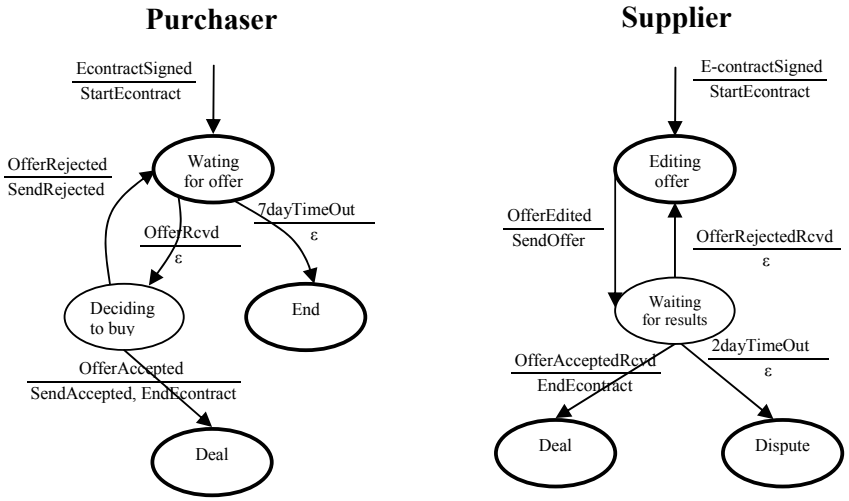


Fig. 5. FSM Representation of an unambiguous e-contract for the purchase of goods.

We have to admit that though the FSM model we present for this rather simple e-contract looks correct at first glance, we do not guarantee it is completely free from inconsistencies. We argue that it is too adventurous to claim that the electronic representation of a contract is free of inconsistencies before the model that describes it is validated using formal tools such as the Spin validator.

What can be done if the validator discovers that the model of an e-contract contains some inconsistencies? We have found out that this situation is rather common with

existing standard contracts. We strongly argue that the existence of inconsistencies in standard English text contract is going to be the normal rather than exceptional.

Because of this, we believe that, except for the fairly well standardized contracts (see Section 5.3) the conversion of a contract into an e-contract is an interactive process. The interaction involves the lawyer and the technical person in charge for the implementation of the e-contract:

1. The lawyer edits the English text contract.
2. The technical person converts the contract into a formal model and validates it.
3. If inconsistencies are discovered in the contract, the technical person goes back to the lawyer (point 1) to request him to correct the English text, taking care that the main purpose of the contract is not changed.
4. If the validator indicates that the model is free from inconsistencies and the lawyer is satisfied with the last version of the English text contract, the technical person proceeds to convert the model into the actual program that will enact the e-contract.
5. Once the English text contract and the e-contract are ready, the contracting parties can sign and enact it.

In Appendix A, we present an example of a contract that we have converted from English text into FSMs. It is a contract for the purchase and supply of e-goods and is inspired in an example for the purchase and supply of goods that appeared in [6]. We changed the original text to make it more illustrative in terms of essential e-contract interactions. Most importantly, we changed the original text to eliminate several ambiguities that prevented the contract from being described with FSMs.

5.3 Ready to fill in, sign and enact e-contracts

We are aware that the manual conversion process discussed in section 5.2 from the English text contract to the electronic contract is not the best solution.
The ideal scenario would be one where an English text contract can be converted by a lawyer into an e-contract that monitors and enforces an agreement, without the intervention of a "technical person". Yet this is currently unrealistic, if this were possible then we would not need programming for any applications. Automation with current technology however, can be achieved for standard contracts.

In the business world, there is a family of applications where the contracting parties resort to fairly standardized contract templates which are offered ready to be filled in and signed. Examples of these templates are tenant agreements. These contract templates can be bought at the stationery. They are offered on the take-it-or-leave-it basis since the clauses of the contract are not negotiable. The contracting parties can negotiate the data to be written in the blanks, but nothing else.

We believe that for these family of applications it is possible to offer (in return for a fee) ready to fill in and sign e-contracts. We can think of a Web place where standard

English text templates are stored together with their inconsistencies-free e-contracts. The contracting parties can then remotely fill in the template that suit their requirements, sign it, pay for the service and enact the e-contract.

The steps that would be required to enable this can be summarized as follows:

Defining Standard contracts
1. Lawyers compile and edit a number of standard contracts used frequently as "standard" between certain entities.
2. The technical person converts each of the standard contracts into FSMs and validates them.
3. If inconsistencies are discovered in a Standard Contract, the technical person reviews the contract with the relevant lawyer, and the English text is corrected.
4. If the "Validator" indicates that the FSM representing the contract is free from inconsistencies, and the lawyer is satisfied with the standard contract, the standard contract is added to the standard contract data base with its FSM.

Agreement phase
1. The trading partners use a "contract editor" to access the data base and choose a standard contract relevant to them.
2. They fill in the relevant data (deadlines, prices, etc.)
3. Once satisfied they both electronically sign the contact.

Enactment phase
1. Both parties take the Signed contract FSM and "plug it in" the contract enforcement system.
2. The contract enforcer uses the FSM code to monitor and enforce the contract agreements between the parties.

How an e-contract can be implemented is discussed in Section 6.

5.4 Who monitors and enforces?

One particularity of the FSMs shown in Fig.4 is that each FSM monitors and enforces the rights and obligations of its owner. Thus the supplier's FSM allows the supplier to execute only the operations he has the right to execute and nothing else. Likewise, the FSM enforces the supplier to execute the operations he has the obligation to execute. The purchaser's FSM works in a similar way. In other words, the rights and obligations stipulated in the English text contract are monitored and enforced at both sides. The advantage of implementing an e-contract by FSMs that enforce both right and obligations at each side is that the receiver of a request to perform a local operation knows that the requester has the right to request such operation, thus, the receiver may perform the operation without checking the rights of the requester. This approach for modeling e-contracts is just one possibility.

Another alternative for modeling an e-contract is to release the check of the rights of the owner of the FSM at his side. In this paradigm, only the FSM of the receiver checks that the requester is requesting operations he has the right to. For instance, we can think of the purchaser of the Fig.4 being allowed to send any request to the

supplier. Thus each operation invoked by the purchaser would reach the supplier's FSM. The advantage of this alternative is that each FSM has to check only obligations, consequently the FSMs become simple.

6 Infrastructure support

Regardless of the details of the implementation, any software designed to implement an e-contract will contain two main components: a *contract enforcer* and a *middleware service* for information sharing.

The contract enforcer is a piece of software that guarantees that the rights and obligations stipulated in the contract are monitored and enforced. An example of obligation that can be enforced by this piece of software is *send offer within three days after signing the e-contract*.

The middleware service is a layer of software that regulates the interaction between two (or more) contracting parties who, despite mutual suspicion, wish to interact and share information with each other. Thus the middleware provides generic services that can be used to support arbitrarily complex interactions between contracting parties. Among the services this middleware should provide are authentication and collection of non-repudiable evidence of the actions performed by the contracting parties. An example of evidence that can be collected is a non-repudiable record that an offer was sent on a certain date.

In the architecture that we propose in this paper for implementing e-contracts, the contract enforcer is implemented as a set of FSMs. As we discussed it in Sections 4.1 and 5.4, we suggest the use of one FSM for each contracting party.

For non-repudiable information sharing we suggest the use of a B2BObject middleware [2] developed at Newcastle University. It is distributed object-based middleware, implemented in Java that supports information sharing between parties that do not necessarily trust each other. Once deployed, each party holds a local copy of shared information (encapsulated in objects). Access to and update of this information is subject to non-repudiable validation by each party.

Fig. 6 shows how an e-contract that involves a purchaser and a supplier is implemented. As can be seen the rights and obligations of each contracting party are described, monitored and enforced by the two FSMs. Consequently, these two FSMs determine the behaviour of the e-contract.

The dashed line that goes from the supplier to the purchaser shows what happens when the supplier sends an offer. When the offer is ready, the supplier invokes a send operation, the supplier's FSM switches to its *Waiting for response* state and makes a *SendOffer* call to the local copy of a shared B2BObject (that implements the operation). The local B2BObj collects, and signs, evidence of the operation and requests coordination of the proposed update to its state with the purchaser's B2BObj. The purchaser's B2BObj verifies the evidence provided and makes an up-call to the purchaser's FSM to validate the B2BObj operation. Upon receiving the up-call, the purchaser's FSM switches to the *Deciding to buy state*. The dashed line from the

purchaser's FSM to the supplier's FSM shows how the purchaser's response is transmitted to the supplier. The B2BObjects middleware ensures that all operations performed by the purchaser and the supplier are recorded and are non-repudiable. One of the major advantages of B2BObjects is that it ensures this without the need of involving trusted third parties.

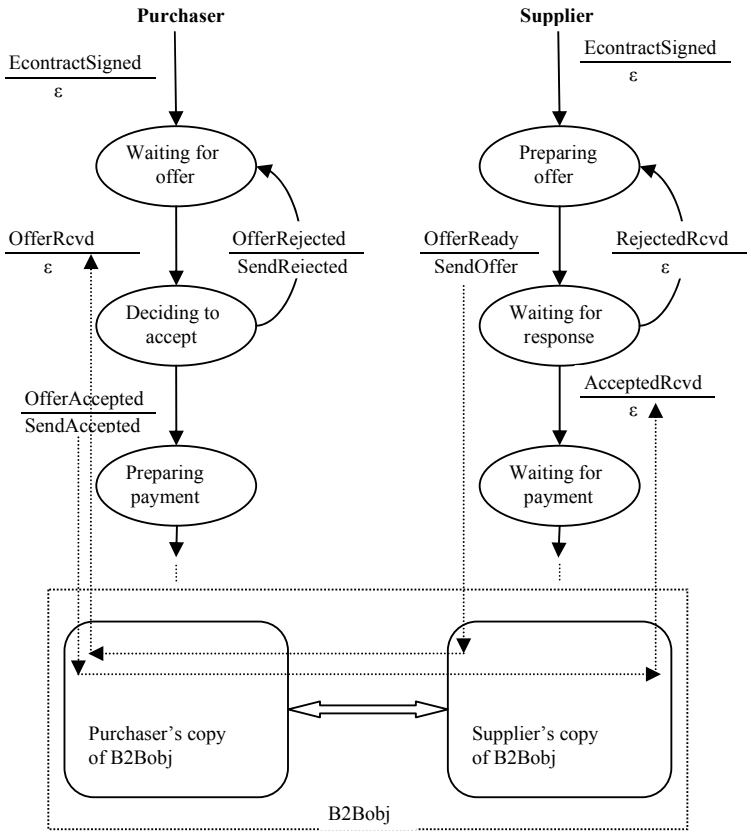


Fig. 6. Architecture for implementing e-contracts.

7 Related work

Monitoring and controlling electronic transactions is addressed by Minsky et al in a number of papers on Law Governed Interaction (LGI) [11]. LGI is an infrastructure that allows members of a group to interact using agents, where agents are entities that interact with each other. A *policy* in LGI is defined as four-tuple: (M,g,CS,L) where: M is the set of messages regulated by this policy; g is an open and heterogeneous set of agents that exchange messages belonging to M; CS is a mutable set of control sates, one (CSx) for every member (x) of group g; L is an enforced set of rules that regulate the exchange of messages between members of g.

Law enforcement as described in [14] is achieved as follows: the law L is enforced by a set of trusted entities called controllers that mediate the exchange of messages (M) between members of group g. For every active member x in g, there is a controller Cx placed between x and the communication medium. Every controller carries the law L.

The controller Cx assigned to x computes the ruling of L for every event at x, and the ruling is carried out locally. Controllers act similarly to our Contract Enforcer (the FSM), which enforces the agreed contract, and regulates interactions between the parties.

Another work of relevance to contract monitoring and enforcement is the Ponder Policy Specification Language [3]. Ponder is a language for specifying management and security policies for distributed systems. Ponder can express authorization and obligation policies as well as some basic descriptive policies through an associated name service. But it is only a specification language and does not provide means for monitoring or enforcing policies.

The idea of monitoring and enforcement of policies specifically for e-contracts is introduced in the works of Milosovic et al [6] [10] [15]. In Milocevic's work, a contract is built of four elements: agreement, consideration, capacity, and legal purpose. These elements take the form of clauses that cover standard contract data. In [15], "A possible sequence of contract operations" is proposed. These include the "Establishment Phase" where the parties negotiate the terms of the contract and sign it, and the "Performance Phase", where the contract is monitored and enforced. The proposal assumes in an implicit way the participation of third parties. In [6], the authors pay particular attention to the benefits of using standard contracts, a concept that is also important in our work. The B2B model they introduce has the following key elements: Contract repository, Contract notary, Contract Monitor, and Contract Enforcer. The Contract Monitor has the central component CMM (Contract Monitor Manager), which receives policy statements of the form:

<policy> :: = <variable_declaration>
when *<condition>*
<action>
must [not] occur where *<condition>*
otherwise *<trigger_action>;*
<action> :: = ***action(****<action_name>, <actor>,*
*<audience>,<time>,<body>****)***
<trigger_action> ::= ***trigger_action(****<action_name>,*
*<audience>,<body>****)***

Upon receiving a policy statement, it is analysed by the CMM, and the CE is signalled if a violation is detected.

The main difference between the above formulation of policy and our design is that using B2BObjects, our implementation does not require a trusted third party for monitoring and enforcing. Monitoring and enforcing is done by the parties to the contract themselves. Therefore the "overall" e-contract in our case needs to be split between the parties depending on what each of them monitors and enforces.

Research conducted at the Cambridge Computer Laboratory [1], discusses contract representation and enforcement using occurrences. An occurrence is basically an event. Contracts are stored in an occurrence store. Workflow occurrences are also stored there, and a collection of stored queries is maintained. Each query describes the occurrences promised and prohibited under the provisions of the contracts and policies of an organisation. [13] also proposes a mechanism that discovers inconsistencies between business contracts and organisational policies.

Work that considers state representation of contracts is introduced in [4] and [5]. In [5] an informal schematic notation for e-contracts is introduced. It can be used to summarize the structure of agreements as collections of interrelated obligations. However formal semantics had not been developed for the notation. In [4], the authors present a simple architecture for an e-market where a controller agent is used to undertake the resolution of possible disputes between parties to an agreement. The controller may hold a representation of a contract in a model language, which implicitly defines state spaces. Also the representation is accessible to each party so that each party knows what it is supposed to do, and what to expect from its counter party.
The controller in this architecture acts as a judge, using information from the contract, and other sources such as advisors for the resolution of disputes. This is a different line of research to ours, as we concentrate on using Finite State Machines to explicitly represent contracts, and enforce them. Any disputes that arise in our case are not currently part of our research.

8 Conclusions and recommendations

Converting an existing standard contract written in English or other human languages into an e-contract is a challenging yet achievable task. The result of the task should be a computer program that, when executed, performs, monitors and enforces the business operations stipulated in the original human oriented document. The ambiguities that are normally present in human oriented contracts make the conversion a difficult process aimed at correcting such ambiguities without changing the main goal of the original text contract. To find ambiguities in the text contract it is strongly advisable to convert it into formal notation.

In this paper, we have shown how finite state machines can be used as a formal notation when it comes to converting text contracts into e-contracts. Finite state machines are scientifically well established, and come with a large body of theory that can be applied to e-contracts. We have found that a finite state machine is a simple yet expressive model for describing, validating and implementing e-contracts.

We have shown how the rights and obligations stipulated in a text contract can be precisely described with a set of finite state machines. From our own experience, we have learnt that a finite state machine model is a useful tool for finding ambiguities in text contracts. Most importantly, we have found that a set of finite state machines can be used to actually implement an e-contract. The FSM would monitor and enforce the rights and obligations of each contracting party, when the e-contract is enacted.

In our ongoing work we are in the process of implementing e-contracts on top of the B2BObjects middleware service. B2BObjects is used to regulate the interaction between the contracting parties and to collect evidence of each of their actions. Using B2BObjects we can show that e-contracts can be monitored and enforced without requiring the involvement of independent trusted third parties.

We are aware that some contract clauses are easier to express in terms of prohibitions rather than rights and obligations. However, we recommend that contracts should be described only in terms of right and obligations and resist the temptation of including

the concept of prohibitions simply because it is easier to describe what must be done rather than what cannot be done. We believe that the concepts of rights and obligations are expressive enough to describe whatever the contracting parties wish to stipulate in their contracts.

9 References

[1] Alan S. Abrahams and Jean M. Bacon, "Representing and Enforcing E-Commerce Contracts Using Occurrences" The Fourth International Conference on Electronic Commerce Research (ICECR-4) Dallas, TX, USA, November 8 - 11, 2001.

[2] N. Cook, S.K. Shrivastava, and S.M. Wheater, "Distributed Object Middleware to Support Dependable Information Sharing between Organisations", Proc. IEEE Int. Conf. on Dependable Syst. and Networks (DSN-2002), Bethesda USA, June 2002.

[3] N. Damianou, N. Dulay, E. Lupu, and M. Sloman, " The Ponder Policy Specification Language", in Proc. Int. Workshop on Policies for Distributed Systems and Networks (POLICY), Bristol, UK, Springer-Verlag LNCS 1995, Jan. 2001.

[4] Aspassia Daskalopulu, Theo Dimitrakos , and Tom Maibaum, "E-Contract Fulfilment and Agents' Attitudes". Proceedings of ERCIM WG E-Commerce Workshop on The Role of Trust in e-Business, Zurich, October, 2001.

[5] Aspassia Daskalopulu. "Modelling Legal Contracts as Processes". *Legal Information Systems Applications*, 11th International Conference and Workshop on Database and Expert Systems Applications, IEEE C. S. Press, pp. 1074–1079, 2000.

[6] Andrew Goodchild, Charles Herring, and Zoran Milosevic, "Business Contracts for B2B", Proceedings of the CAISE*00 Workshop on Infrastructure for Dynamic Business-to-Business Service Outsourcing, Stockholm, June 5-6,2000.

[7] David F. Ferraiolo, Ravi Sandhu, Serban Gavrila, D. Richard Kuhn and Ramaswamy Chandramouli, "Proposed NIST Standard for Role-Based Access Control", ACM Transactions on Information and Systems Security, Vol. 4, No. 3, August 2001.

[8] Paul Halsall, "Ancient History Sourcebook: A Collection of Contracts from Mesopotamia, c. 2300--428 BCE",http://www.fordham.edu/halsall/ancient/mesopotamia-contracts.html,March, 1999.

[9] Gerard J. Holzmann, "Spin Home Page: ON-THE-FLY, {LTL} MODEL CHECKING with SPIN", http://spinroot.com/spin/whatispin.html, October, 2002.

[10] Olivera Marjanovic, and Zoran Milosevic, "Towards Formal Modelling of e-Contracts" Proceedings of Fifth IEEE International Enterprise Distributed Object Computing Conference, Seattle, Washington, September 04-07, 2001.

[11] Naftaly H. Minsky, Victoria Ungureanu, "Law-Governed Interaction: A Coordination and Control Mechanism for Heterogeneous Distributed Systems". ACM Press, New York, NY, USA, TOSEM 9(3): 273-305, 2000.

[12] Franco P. Preparata, and Raymond T. Yeh, "Introduction to Discrete Structures for Computer Science and Engineering", Addison--Wesley Publishing Company, 1974.

[13] Brian Shand, and Jean Bacon, "Policies in Accountable Contracts", 3rd International Conference on Policies for Distributed Systems (POLICY 2002), June, 2002-11-15

[14] Victoria Ungureanu, and Naftaly H. Minsky. “Establishing Business Rules for Inter_Enterprise Electronic Commerce”. International Symposium on Distributed Computing (DISC) Workshop on Distributed Algorithms (WDAG), Toledo, Spain, October 4-6, 2000.

[15] Zoran Milosevic, and Andy Bond, “Electronic Commerce on the Internet: What is Still Missing ?”, Proc. 5th Conf. of Internet Society, pp.245-254, Honolulu June 1995.

Appendix A: Sample of a standard commercial contract

E-CONTRACT FOR THE PURCHASE AND SUPPLY OF E-GOODS

This Deed of Agreement is entered into as of the Effective Date identified below.

BETWEEN [Name] of [Address] (To be known as the (Supplier) in this Agreement) AND: [Name] of [Address] (To be known as the (Purchaser) in this agreement)

WHEREAS (Supplier) desires to enter into an agreement to supply (Purchaser) with [Item] (To be known as (E-goods) in this Agreement).
NOW IT IS HEREBY AGREED that (Supplier) and (Purchaser) shall enter into an agreement subject to the following terms and conditions:

1. Definitions and Interpretations
1.1 Price, Dollars or $ is a reference to the currency of the [Country].
1.2 All information (purchase order, payment, notifications, etc.)is to be sent electronically.
1.3 This agreement is governed by [Country] law and the parties hereby agree to submit to the jurisdiction of the Courts of the [Country] with respect to this agreement.

2. Commencement and Completion
2.1 The commencement date is scheduled as [date].
2.2 The completion date is scheduled as [date].
2.3 The schedule may be modified by agreement as defined in Section 9.

3. Purchase Orders
3.1 The (Purchaser) **shall** follow the (Supplier) price lists.
3.2 The (Purchaser) **shall** present (Supplier) with a purchase order for the provision of (E-goods) within 7 days of the commencement date.
3.3 The (Supplier) **shall** notify the (Purchaser) of acceptance or rejection of the purchase order within 7 days after the receipt of the purchase order.
3.4 If the purchase order is rejected, the (Purchaser) **shall** remedy the situation within 14 days after the receipt of the notification.

4. Delivery
The delivery of the (E-good) is the responsibility of the (Purchaser). The (Supplier) **shall** keep the E-good available for downloading at the specified e-address for at least 14 days after receipt of payment. The (Purchaser) **shall** download the E-good within this period of time.

5. Payment
5.1 The payment **shall** be sent in full to the (Supplier) within 7 days after receiving a notification of acceptance or the purchase order.
5.2 The (Supplier) **shall** notify the (Purchaser) of acceptance or rejection of the payment within 7 days after the receipt of the payment.

6. E-goods rejection

6.1 If the (E-goods) do not comply with the order or the (Supplier) does not comply with any of the conditions, then the (Purchaser) is, at its sole discretion, **entitled to** reject the (E-goods).
6.2 The (Purchaser) **shall** notify the (Supplier), of acceptance of the (E-goods) or return the (E-goods) to the (Supplier), within 7 days after receiving them.

7. Replacement and refund
7.1 The (Supplier) **may** use its discretion to replace the (E-goods) according to the invoice or refund any monies paid.
7.2 The (Supplier) **shall** notify the (Purchaser) of refusal to replace or refund, within 14 days after the receipt of the rejected (E-goods) or replace or refund any monies paid, within 14 days after the receipt of the rejected (E-goods).

8. Termination
8.1 If (Purchaser) or (Supplier) fail to carry out any of its obligations and duties under this agreement the offender is to be considered in **breach of the e-contract.** In this case, the offended party may issue a notice specifying the breach and request that it be remedied within 14 days after receipt of such notice.
8.2 If a breach is not remedied, the offended party may send, within 7 days after the failure to receive a remedy to the offence, a notification of e-contract termination containing:
(a) no-complains.
(b) an invitation to resolve disputes outside this e-contract.

9. Disputes
9.1 (Supplier) and (Purchaser) **shall** attempt to settle all disputes, claims or controversies arising under or in connection with the agreement through consultation and negotiations in good faith and a spirit of mutual cooperation.
9.2(Supplier) and (Purchaser) **shall** provide electronic evidences about breaches of the e-contract.
9.3 This method of determination of any dispute is without prejudice to the right of any party to have the matter judicially determined by a [Country] Court of competent jurisdiction.

10. Amendment
10.1 This agreement **may** only be amended in writing signed by or on behalf of both parties.

E-SIGNATURES
In witness whereof (Supplier) and (Purchaser) have caused this agreement to be entered into by their duly authorized representatives as of the effective date written below.

Effective date of this agreement: [day] day of [month] [year]

[E-signature]	[E-signature]
[Person]	[Person]
[Role]	[Role]
E-address for Notices:	
[E-address]	[E-address]

Split of rights and obligations
Before describing the e-contract in FSM notation, it is advisable to extract, from the English text, the purchaser's and supplier's Rights (R) and Obligations (O).

Supplier's obligations

O01: Notify Purchaser of acceptance or rejection of Purchase order within 7 days after receipt of purchase order.
O02: Place e-goods at e-address for 14 days after receipt of payment.
O03: Notify Purchaser of acceptance or rejection of remedy request if the purchaser is not satisfied with the e-goods.
O04: Provide remedies for breaches of the e-contract.
O05: Provide electronic evidences of breach of the e-contract.

Supplier's rights

R01: Reject a Purchase order if it is not consistent with the Requirements Of the contract.
R02: use his discretion to refund monies paid for rejected E-goods.
R03: Use his discretion to replace rejected E-goods.
R04: Notify the Purchaser about breaches of the e-contract.
R06: Terminate the e-contract if the Purchaser does not provide remedy within 14 days of receiving a notice that he/she is in breach of contract.
R06: Amend contract but only in agreement with the Purchaser.

Purchaser's obligations

O01: Follow the supplier's price lists.
O02: Present a purchase order within 7 days of the commencement date.
O03: Send purchase order electronically.
O04: Correct a Purchase order within 14 days after receipt of a notification of rejection of the Purchase order.
O05: Send full payment within 7 days after receiving notification of acceptance of purchase order.
O06: Download the E-good\s within 14 days after the receipt of the payment.
O07: Send acceptance or rejection of e-goods within 7 days of receiving them.
O08: Provide remedies for breaches of e-contract.
O09: Provide electronic evidences of breach of e-contract.

Purchaser's rights

R01: Download the E-goods at anytime within 14 days after payment.
R02: Reject E-goods that fail to match the description.
R03: Receive rejected E-goods replacement.
R04: Receive refund for rejected E-goods.
R05: Notify the supplier about breaches of the e-contract.
R06: Terminate the e-contract if the Supplier does not provide remedy within 14 days of receiving a notice that he/she is in breach of contract.
R07: Amend contract but only in agreement with the supplier.

In Fig.1 and Fig. 2 PO stands for Purchase Order.

Supplier

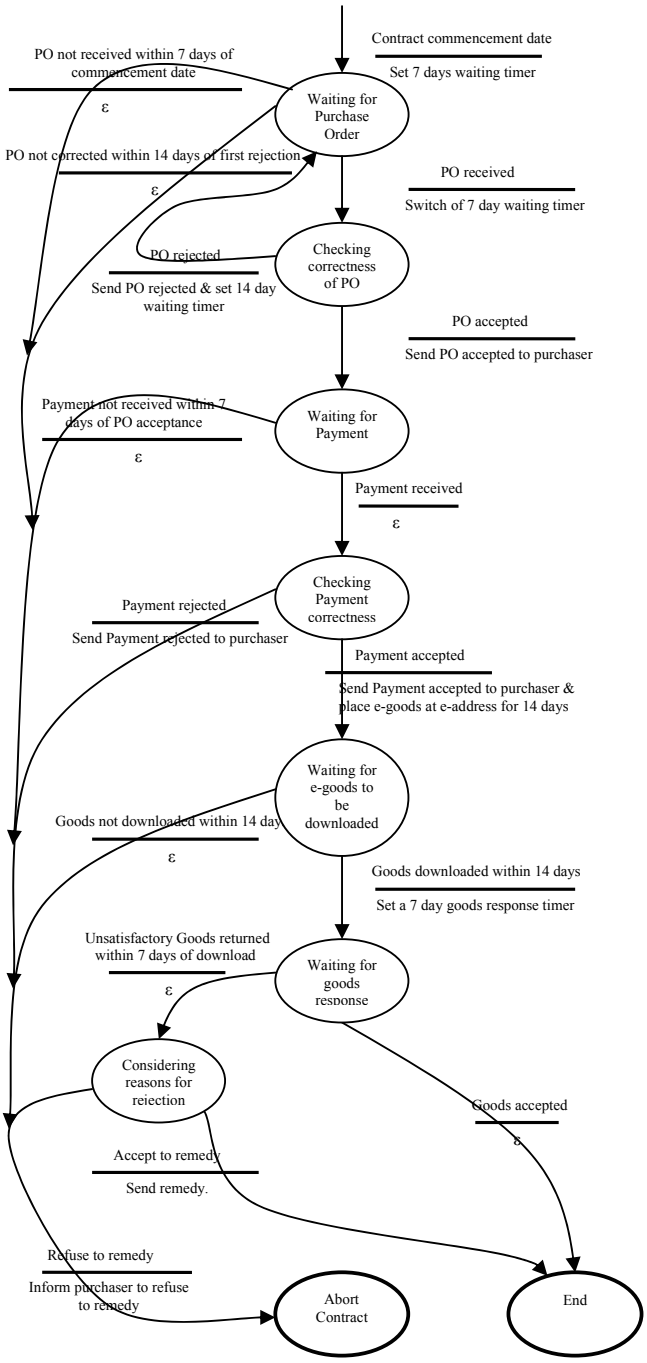


Fig. 1. Supplier's FSM.

Purchaser

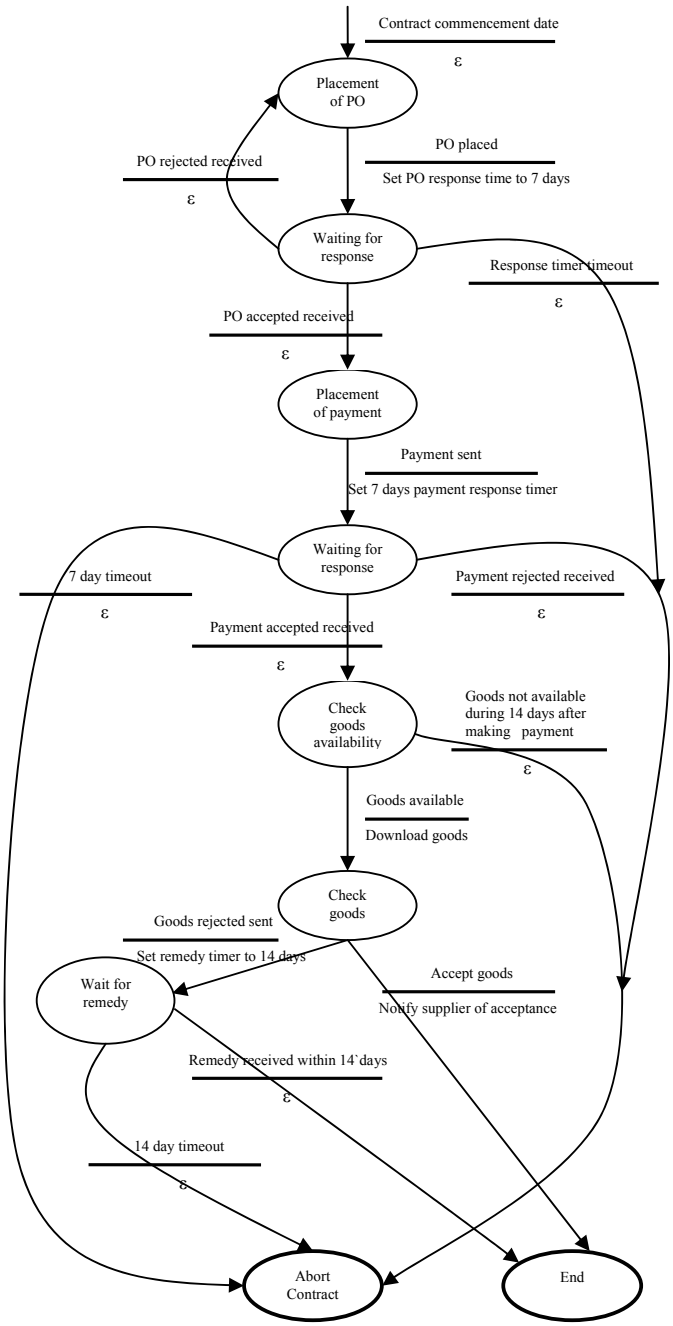


Fig. 2. Purchaser's FSM.