# Towards Formal Modeling of e-Contracts

Olivera Marjanovic and Zoran Milosevic*

School of Information Systems, Technology and Management,
University of New South Wales
Sydney, Australia

*Distributed Systems Technology Centre,
The University of Queensland, QLD 4072 Australia

*E-mails:* o.marjanovic@unsw.edu.au, zoran@dstc.edu.au

## Abstract

*The emerging B2B technologies allow for more automated management of e-contracts including contract drafting, negotiation and monitoring. As technology infrastructure becomes available for electronic exchange of contracts and contract-related messages, the IT community is becoming more interested in modeling of contracts as governance structures for many inter-organisational interactions.*

*This paper presents our initial ideas for formal modeling of e-contracts. This includes specification of deontic constraints and verification of deontic consistency associated with roles in a contract, precise modeling of temporal constraints/estimates and verification of temporal consistency of an e-contract, and finally scheduling of the required actions. The paper also introduces visualisation concepts such as role windows and time maps and describes how they could be used as decision support tools during contract negotiation.*

## 1. Introduction

Businesses globally are undergoing a revolution being driven by a confluence of many different factors such as global competition, increased customer demands and emerging technologies. E-commerce has attained sufficient critical mass to result in the emergence of new business opportunities. Thus, it is little wonder that businesses have adopted e-commerce as a way to reach more customers while enjoying reduced costs.

The last few years have seen a rapid growth in business-to-business (B2B) e-commerce models. Many companies, eager to capitalise on this new market, have joined the world of e-commerce only to have their on-line stores fail because their current business practices could not keep pace with the demands of this new environment. For example, simply offering catalogs on-line and allowing credit card payments are not challenging concepts and do not require any great shift from the long established methods of commerce such as telephone sales. Many industry analysts and corporate leaders believe that simple transaction-based business models will have to be augmented with higher value-added services, if e-marketplaces are to remain competitive.

In order to ensure legality and protect interests of all parties involved in e-commerce, electronic business interactions should be regulated by contracts, as is the case with traditional business interactions. The emerging B2B technologies make it possible to support management of contracts including support for electronic representation, composition, verification of their validity and consistency as well as contract negotiation and monitoring [5].

Currently there are many companies that already offer or are in the process of developing technical platforms and solutions (e.g. BizTalk, e-Speak, J2EE etc.) that enable high-level service composition and execution. As technology infrastructure becomes available for exchanging contract related messages, the IT community is becoming more interested in modeling of contracts as governance structures for many inter-organisational interactions.

The main objective of this paper is to describe our approach towards formal modeling of e-contracts. This includes formal modeling of deontic constraints and verification of deontic consistency associated with roles in a contract, formal modeling of temporal constraints and estimates, verification of temporal consistency of an e-contract and finally scheduling of the required actions. The paper also introduces visualisation concepts such as role windows and time maps. These simple concepts can be used for verification and scheduling but also as decision support tools during contract negotiation.

The paper is organised as follows. Section 2 introduces e-contract building blocks. It gives a short overview of the Reference Model of Open Distributed Processing (RM-ODP) and introduces formal modeling of temporal and deontic constraints. Section 3 describes formal modeling of e-contracts. It also introduces visualisation concepts such as role windows and time maps and explains how they could be used as decision support tools during contract negotiation. Finally, Section 4 describes related work in the area of e-contracting.

# 2. E-contract building blocks

## 2.1.The reference model of open distributed processing (RM-ODP)

The Reference Model of Open Distributed Processing RM-ODP [2] is increasingly being used for modeling of complex, open distributed systems. The ODP enterprise viewpoint defines the purpose, scope and policies for an ODP system. More precisely, the enterprise language introduces concepts and terminology necessary to produce an enterprise specification. With some extensions and modifications, it has been used as a practical framework for modeling of virtual enterprises, in particular e-contracts in B2B services (see for example[1]). In this section, we provide a brief overview of the basic concepts applicable to e-contracting.

A concept of *community* is the main structural element and reflects some grouping of people and resources in the real world. A grouping can be considered a community if it is formed to collectively achieve some objectives. This collective behaviour is expressed in terms of roles where each role identifies some subset of the overall community behaviour that can be meaningfully performed by a single object within the community. The concept of a role is sufficiently general to specify the behaviour of entities which can be either (parts of) IT systems or people.

A *contract* is a generic RM-ODP concept that specifies an agreement governing part of the collective behaviour of a set of objects. It specifies how community objectives can be met. More precisely, it defines obligations, permissions and prohibitions for the roles involved. An *obligation* is a prescription that a particular behaviour is required. An obligation is fulfilled by the occurrence of the prescribed behaviour. A *permission* is a prescription that a particular behaviour is allowed to occur. A permission is equivalent to there being no obligation for the behaviour not to occur. A *prohibition* is a prescription that a particular behaviour must not occur. A prohibition is equivalent to there being an obligation for the behaviour not to occur. These definitions are in a style of formal logic called *deontic logic*. A formal model of obligation, permission and prohibition, based on deontic logic, will be introduced later in the paper.

.

## 2.2. Modeling of time

The ODP-RM Enterprise viewpoint is yet to address the temporal nature of obligations, permissions and prohibitions [3]. However, proper modeling of temporal constraints is critical in e-contracting especially for its preparation and verification.

### 2.2.1. Basic temporal concepts

In this section we introduce primitive temporal concepts needed for expressing temporal constraints and relationships in e-contracting. These primitive concepts can be combined to construct more complex temporal expressions.

- ***Absolute time***

An absolute time value (also called a time point) is commonly specified in terms of UTC (Universal Coordinated Time) that includes specification of different time zones. This time format is commonly used in distributed systems that span several time zones.

When working with absolute time the following relations of temporal precedence are used: “<”, “≤”, “=”,“>”, “≥”, with meaning “before”, “before or at the same time” “at the same time”, “after” or “after or at the same time”. A pair of absolute time values (t1, t2) such that t1 precedes t2 (t1 ≤ t2) is called a time interval.

- ***Relative time***

A concept of relative time is used to model time duration that is independent from any time point e.g. 2 days, 5 hours. To compare two relative time values we use the following relative time operators: “<”, “≤”, “=“ “>”, “≥ “ that are interpreted as “less than”, “less than or equal”, “equal”, “more than”, “more than or equal”.

Note that since relative time does not have any temporal reference, in practice it is often combined with absolute time e.g. 2 days after *Date1* where Date1 can be determined dynamically (an application must be reviewed 2 days after its submission date). This is an example of a more complex temporal expression.

- ***Repetitive (periodic) time***

The concepts of absolute time (time points) and relative time are used together to define a concept of repetitive time. A repetitive time is a set of ordered time points such that the distance between two consecutive time points is constant and correspond to some relative time value d. Thus, a repetitive time values can be represented as:

$$r = (tb, te, d)$$

where *tb* and *tb* correspond to the beginning and end of a time interval that represents the domain of the repetitive time while *d* is a relative time that indicates the distance between time points.

In practice, the concept of repetitive time is used to describe events that occur regularly, starting from a certain point in time and are repeated every d time until the final time point is reached.

### 2.2.2. Temporal constraints

Temporal constraints are different rules that regulate the order, timing and duration of individual actions. It is possible to distinguish between hard and soft temporal constraints. *Hard* temporal constraints usually result in some consequences if the corresponding action is not performed as required (e.g. late grant applications are not accepted). This is of particular importance for actions where any deviation from the prescribed behaviour can be illegal, dangerous or very costly. *Soft temporal* constraints imply that the original temporal constraints could be relaxed under certain circumstances, however each relaxation is likely to lead to some kind of penalty e.g. financial penalty if a project is not completed on time.

- ***Notation***

Before we proceed with formal definitions of temporal constraints, we introduce the notation that will be used throughout the paper to define temporal and deontic constraints.

- *action-id* is a unique action identifier
- *temporal-operator* ∈ {"<", "≤", "=" ">", " ≥ "} is used for comparison of either two relative time values or two absolute time values
- *d-limit* is a relative time value that corresponds to a prescribed time limit
- *type* ∈ {h,s} determines the type of temporal constraint i.e. *h* corresponds to *hard* and *s* to *soft* temporal constraint.
- *temporal-reference* ∈ {'b','e'} is used to denote a beginning 'b' or an end 'e' of an action.
- *deadline* is an absolute time value e.g. *Date1*, *Date2* etc.
- *distance* is a relative time value that corresponds to the distance between two time points.
- *time-period* is a relative time value that determines the period of repetition of an action
- *b-time-point* and *e-time-point* are two absolute time points that determine a domain of the repetitive time
- *otime* denotes an absolute time value when an action is estimated to occur

The above notation should be used to interpret the following definitions of temporal constraints.

- ***Formal definition of temporal constraints***

*Duration constraints* limit duration of individual actions (e.g. verification of an application for life insurance must not take more than 5 working days). Formally, this constraint is represented as:

*Duration (action-id, temporal-operator, d-limit, type)*

For example:

*Duration (ai, ≤, d, h)*

prescribes that action *ai must* be completed in no more than *d* time (as it is a hard temporal constraint). Similarly,

*Duration (ai, ≥ , d, s)*

prescribes that action *ai should* take no less than *d* time to complete (as it is a soft temporal constraint).

Note that a duration temporal constraint does not prescribe when an action should/must start and/or finish, only how long it should/must take.

Hard and soft duration constraints can be visualised as depicted by Figure 1.

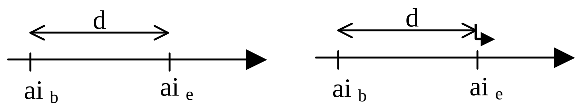


**Figure1. Hard and soft duration constraints for action ai.**

An *absolute deadline constraint* limits, in terms of absolute time, when an action *must/should* finish (e.g. the deadline for grant applications is 2.April, 2001, 5pm sharp). Formally, it is defined as:

*A_Deadline (action-id, temporal-reference, temporal-operator, deadline, type)*

For example:

*A_Deadline(ai, e, ≤, Date1, h)*

prescribes that action *ai must* be completed no later than *Date1*.

Similarly,

*A_Deadline(ai, b, ≤, Date1, s)*

prescribes that action *ai should* start no later than *Date1*.

Hard and soft absolute deadline constraints can be visualised as depicted by Figure 2.

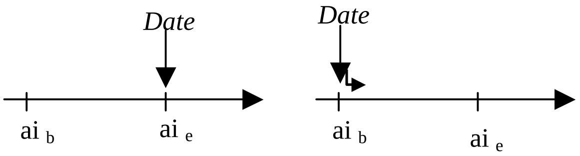


**Figure. 2. Hard and soft absolute deadline constraints**

A *relative deadline constraint* limits when an action must/should begin/end relative to the beginning/end of another action. The distance between two reference points is expressed in terms of relative time. Formally:

*R_Deadline(action1-id, temporal-reference, temporal-operator, action2-id, temporal reference, distance, type)*

For example,

*R_Deadline (aj, b, ≤, ai, e, d, h)*

prescribes that action *aj* must start no later than *d* time after action *ai* is completed.

An example of hard and soft relative deadline constraints is depicted by Figure 3.

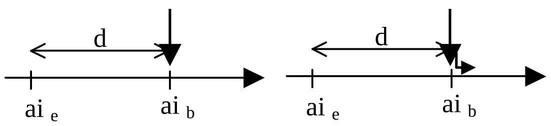


**Figure 3. An example of hard and soft relative deadline constraints**

Note that relative deadline constraints can be also used to prescribe the order of individual actions. For example,

*R_Deadline (aj, b, = , ai, b, -, s)*

prescribes that actions *ai* and *aj* should start at the same time.

*Periodic deadlines* are temporal constraints used to prescribe the occurrence of an action in terms of repetitive time. Formally,

*P_Deadline (action-id, temporal reference, time-period, b-time-point, e-time-point, type)*

For example:

*P_Deadline (ai, e, d, Date1, Date2, h)*

prescribes that action *ai* should be completed every *d* time starting from *Date1* until *Date2* is reached.

This temporal constraint can be visualised as depicted in Figure 4.

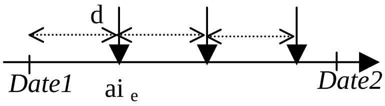


**Figure. 4: An example of a repetitive deadline constraint**

- ***Temporal consistency***

A set of temporal constraints is *mutually consistent*, if and only if it is possible to find any assignment of temporal attributes (beginning, end and duration) for all actions such that all temporal constraints can be satisfied.

For example suppose that the following two constraints are given: An action of testing one's automotive horn must be performed (completed) once per month. However, the same action mustn't occur at the nighttime (e.g. between 7p.m.and 7a.m.). Thus it is possible to find an assignment of temporal attributes for this action that satisfy both temporal constraints (i.e. the action must be performed once per month between 7 a.m. and 7p.m.)

- ***Temporal estimates***

Temporal estimates are not temporal constraints. They are based on the accumulated experience and describe estimated duration and order of individual actions. They are important for scheduling of individual actions and resource planning.

Thus, *estimated duration* of an action is formally modeled as:

*EDuration (action-id, temporal-operator, d-limit)*

For example:

*EDuration (ai, =, d)*

is interpreted that action ai could take d time to complete.

*Estimated occurrence* is used to express the fact that an action could occur after/before some absolute time or periodically every d time.

*EOccurence (action-id, temporal-reference, temporal-operator, otime)*

For example:

*EOccurence (ai, b, <, Date1)*

is interpreted as: action *ai* could start before *Date1*. Again this doesn't mean that ai will start at this time or that it will start at all.

*Estimated order* is used to express how an action could start/end relative to the beginning/end of another action.

*EOrder(action1-id, temporal-reference, temporal-operator, action2-id, temporal-reference)*

For example:

*EOrder(ai, b, <, aj, b)*

is interpreted that action ai could start before action aj starts.

## 2.3. Deontic constraints

In role-based models (such as for example e-contracting), roles and their responsibilities have to be specified explicitly to prevent any possible misunderstanding or ambiguity. In terms of temporal attributes, a contract specification includes two temporal attributes: an absolute time indicating when the contract was signed and a time interval that specify the period of contract's validity. Formally, a contract can be specified as follows (note that for simplicity all other attributes are omitted):

*C (contract-id, …, date-signed, c-begin, c-end)*

where *c-begin* and *c-end* are two absolute time points that determine the period of contract validity. We note that there are other temporal attributes related to the contract, such as those related to the actions of parties to the contract. These are expressed as part of policies applicable to individual parties as discussed in constraints applicable to individual roles as below.

Note that for some types of contracts, the right side of the interval can be initially open (until some other conditions are fulfilled) or specified but later changed (for example a home loan contract can be initially valid for 25 years, but the end date can be changed if additional repayments are made).

Now suppose that contract *ci* is signed on *Date1* and has a period of validity is (*cb, ce*).

*C (ci, …, Date1, cb, ce)*

As already stated, a contract is formally defined as a set of deontic constraints i.e. obligations, permissions and prohibitions of various roles. Our representation of deontic constraints is based on deontic logic that is extended to include the concept of time.

- ***Obligations***

An obligation can be formally represented as:

*O(role, action-id, temporal-reference, temporal-operator, deadline, tdistance, ob, oe)*

where *role* is obliged to perform *action-id* either by the *Deadline* or every *tdistance* starting from *ob* until *oe* is reached. Note that *(ob , oe)* is the period of validity of this deontic constraint.

This deontic constraint is properly defined if the following conditions are satisfied:

a) Time interval (*ob, oe*) has to be contained within (*cb, ce*) i.e.

$$cb \le ob \le oe \le ce$$

b) Absolute time value *deadline* has to be within the period of validity of this deontic constraint i.e.

$$ob \le deadline \le oe$$

c) In the case of repetitive time, *Role* must be able to perform *Action* at least once i.e.

$$ob + tdistance \le oe$$

The following are some examples of obligations:

*O(R1, ai, e, ≤, Date1, -, t1, t2)*

it prescribes that role R1 is obliged to finish action ai no later than Date1. This obligation is valid from time t1 to t2. Observe that *tdistance* attribute is not applicable to this type of deontic constraint.

This deontic constraint will generate two temporal constraints as follows:

If *Date1 = t2* then the deadline could not be extended and both generated temporal constraints will be hard:

*A-Deadline (a1, e, ≤, Date1, h)*
*A-Deadline (a1, b, >, t1, h)*

However, if *Date1*< *t2* then the first temporal constraint will become soft as deadline *Date1* can be extended until *t2*.

*A-Deadline (a1, e, ≤, Date1, s)*

Similarly,

*O(R1, a3, e, =, -, d, t1, t2)*

prescribes that role *R1* is obliged to complete action *a3* every *d* time, starting from time *t1* until time *t2* is reached. As a result the following temporal constraint will be generated:

*P_Deadline (a3, e, d, t1, t2, h)*

- ***Permissions***

A permission can be formally represented as:

*P(role, action-id, temporal-reference, temporal-operator, deadline, tdistance, pb, pe)*

indicates that *role* is permitted to perform *action-id* either by the *deadline* or every *tdistance* starting from *pb* until *pe* is reached.

A permission is well defined if the following conditions are satisfied: a permission has to be valid during the period of contract's validity; absolute time value *deadline* has to be within the period of validity of this permission; and in a case of repetitive time, a *role* should be able to perform *action-id* at least once.

The following are some examples of permissions:

*P(R1, ai, b, >, Date1, -, t1, t2)*

it states that role *R1* is permitted to start action *ai* after *Date1* and it is valid from time *t1* to *t2*.

Permissions do not result in temporal constraints as they do not prescribe that action *ai* must occur. Rather, two temporal estimates will be generated as follows:

*EOccurence (ai, b, > , Date1)*

*EOccurence (ai, e, ≤ , t2)*

meaning that action *ai* could be expected to start after *Date1* and finish by *t2*.

The following is an example of periodic permission:

*P(R2, ai, b, =, -, d, pb, pe)*

that can be interpreted as role *R2* is permitted to perform action *ai* every *d* time starting from *pb* until *pe* is reached. This will generate a number of temporal estimates:

EOccurence (ai, b, =, pb+d)

EOccurence (ai, b, =, pb+2d)

The number of temporal estimations is equal to the maximum number n such that:

$$pb + nd \leq pe$$

- **Prohibitions**

As already stated prohibitions are used to express that an action is forbidden to happen. Formally,

*F(role, action-id, temporal-reference, temporal-operator, atime, fb, fe)*

states that *role* is forbidden to perform *action-id* during a certain period of time - that is determined by absolute time value *atime* and the period of validity of this deontic constraint: is from *fb* to *fe*. Note that prohibitions are defined for a period of time rather repetitively.

This deontic constraint is properly defined if the following conditions are satisfied: its period of validity has to be within the period of contract's validity and an absolute time value *atime* should be within the period of validity of this temporal constraint.

Note that if an action is prohibited for one role that does not imply that all other roles are prohibited to do the same action. For example an administrative officer is prohibited to sign an authorization for overseas travel while CEO is permitted to do it.

## 2.4. Temporal and Deontic Constraints in Contracts

The primitive temporal concept introduced in 2.2.1 and various more complex temporal expressions that involve combination of these primitive concepts can be used for time characterisation of actions in communities, such as their duration and temporal relationships between different actions. In addition, they can be used to determine temporal consistency of these actions such as ensuring that an action is prohibited in certain time interval, but not in another one, as in parking restrictions in cities.

Furthermore, in the context of a community, the actions in a community are attributed to the roles that the community consists of. Hence, the temporal characterisation of actions can be associated with the roles in a community. This is indeed more of interest when analysing union of temporal and deontic constraints in a community. We note that as policies are defined by a community, so are the temporal constraints defined by the community - in fact, in many cases temporal constraints can be regarded as an integral part of policy statements, as in obligation to execute some action by some absolute point in time.

When considering a contract as a specification of roles in a community, their mutual obligations and other policies applicable to the roles (such as those arising from the community's outer scope), there are several areas where temporarily-enriched deontic expressions can be of particular importance. They can be used to formally define consistent (both temporal and deontic) behaviour of trading partners to a contract. This formal specification can be then used to facilitate negotiation between parties to the contract, ensuring a valid contract from the outset (both in terms of feasibility and legal validity). It can be also used as an input to some automated monitoring tools that can be able to interpret policies and thus detect a behaviour of a party to the contract that is non-consistent to the contract specification. In this paper, we limit our discussion to verification of temporal and deontic constraints.

## 3. Towards formal modeling of e-contracts

To formally model an e-contract, we use the building blocks introduced in the previous sections of this paper.

### 3.1. Visualisation of deontic constraints

A contract is represented as a set of deontic constraints. Thus the first step is specification of deontic constraints including specification of roles and their permissions, obligations and prohibitions. For that purpose we use formal statements introduced in Section 2.3.

To visualize deontic constraints and corresponding temporal constraints assigned to a role we use a concept of a role window (as depicted in Figure 5). A role window depicts *temporal constraints within deontic context*. Note that a contract specifies a community and thus role windows are always used within the same community.

The role window is divided into 3 different areas that correspond to obligations (O), permissions (P) and prohibitions (F) assigned to that particular role. Within each area parallel time lines are constructed (one per action). Each timeline has the corresponding time interval during which an action must or should occur as defined by the corresponding hard and soft temporal constraints respectively (as represented in the first area), could occur (as represented in the second area) or must not occur (as represented in the third area). The actual duration of each action is in fact shorter than the corresponding time interval represented in a role window. This is because an action is expected to occur within that interval. Also note that all timelines are limited on the left and right side by Cb and Ce (i.e. period of contract validity).

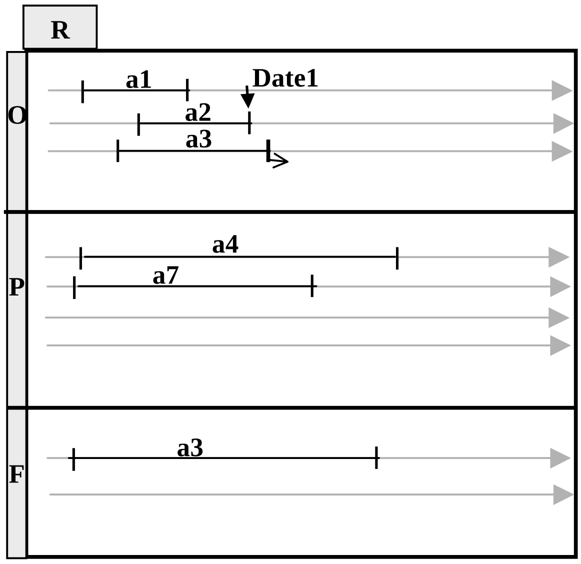


**Figure 5. A role window for R**

The same concept can be further generalised to provide a "summary" of all role windows for the same contract as depicted in Figure 6. This summary window is projection of deontic constraints associated with the same role across different communities (i.e. contracts) where this role belongs to. This summary window can be used for cross-comparison and various analysis of temporal constraints. Similarly the same concept can be extended to represent deontic constraints for a single role across different contracts C1, C2 and C4 as depicted in Figure 6.

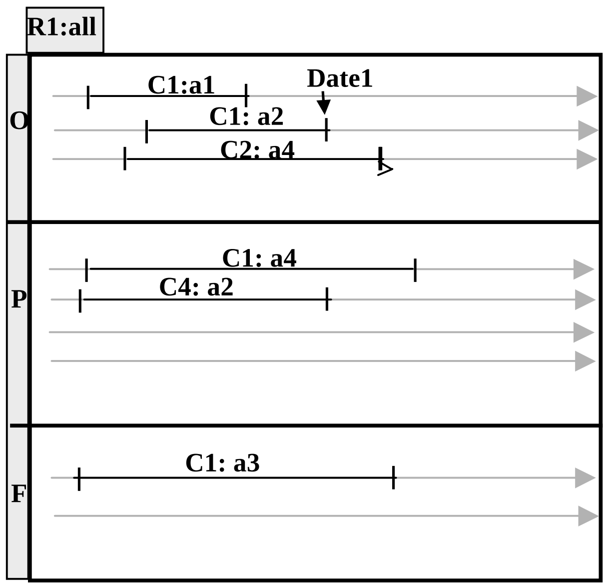


**Figure 6 A summary window for a single role across different contracts**

The summary windows can be used during contract execution for monitoring purposes.

### 3.2. Verification of deontic consistency

After all deontic constraints are specified it is necessary to perform verification of their temporal consistency especially when dealing with contracts with large number of constraints. Verification is based on deontic logic rules as follows:

The first case of deontic inconsistency arises when the same role is both obliged and forbidden to do the same action within the same time interval. In other words periods of validity of these two deontic constraints overlap. Observe that the concept of time is crucial here, because the same role can be permitted to do an action and then forbidden. However, this situation will not result in deontic inconsistency as their corresponding time intervals do not overlap.

Hence, the following two deontic constraints

$$O(Ri, ai, b, \leq, Date1, -, t1, t2)$$

$$F(Ri, ai, b, >, Date2, -, t3, t4)$$

will result in deontic inconsistency if the following time intervals: (t1, Date1) and (Date2, t4) overlap.

Similarly the following two deontic constraints:

$$\mathrm{O(Ri, ai, e, =, -, d, t1, t2)}$$

$$F(Ri, ai, b, >, Date2, -, t3, t4)$$

are mutually inconsistent if the following two time intervals: (t1+d, t2) and (t3,t4) overlap.

Another case of deontic inconsistency arises when the same role is both permitted and forbidden to do the same action during the same period of time. Thus the following two deontic constraints:

$$P(Ri, ai, b, \leq, Date1, -, t1, t2)$$

$$F(Ri, ai, b, >, Date2, -, t3, t4)$$

are mutually inconsistent if the following two time intervals: (t1, Date1) and (Date2, t4) overlap.

Similarly, it is possible to verify mutual inconsistency of obligations and permissions associated with the same role.

Obviously, the existence of a large number of deontic constraints can make the problem of manual verification of their mutual inconsistency time consuming and error-prone because it is necessary to compare all possible pair combinations of deontic constraints for the same action (e.g. prohibitions with obligations etc.) We propose a simple, yet very effective visual mechanism for verification of deontic inconsistency based on the introduced concept of a role window. After a role window is constructed for each role, visual verification of temporal constraints can start. For that purpose it is necessary to take the first area (that corresponds to obligations) and determine all referential time points (where an interval start or finish).

After all referential time points are determined in the first area it is possible to construct a vertical partitions across all three areas at each referential point (as shown in Figure 7).

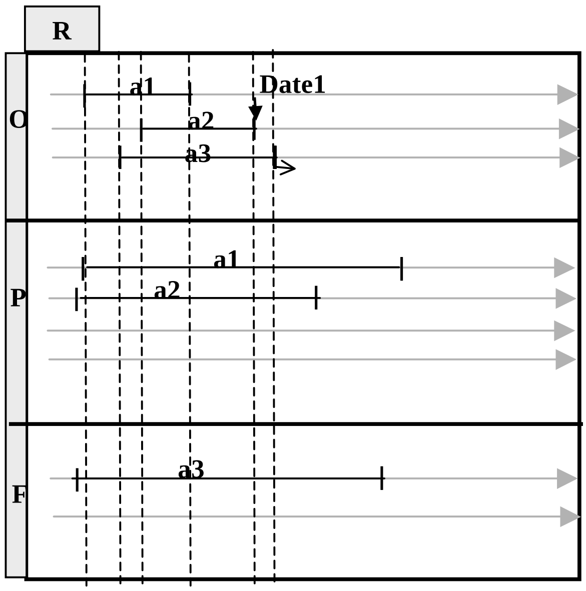


**Figure 7: Verification of deontic consistency**

So, in order to verify deontic inconsistency instead of the above manual method, it is necessary to scan the complete role window partition by partition. This is a more user-friendly way of verification of deontic constraints that can be easily automated. If the same action is detected in the first and third area that corresponds to prohibition – an inconsistency is detected.

Similar procedure can be used to detect other type of inconsistency that could occur between the second and third areas of the role window (that correspond to permissions and prohibitions). However, in that case referential points will be determined in the second area (that corresponds to permissions).

### 3.3. Verification of temporal consistency and scheduling of actions

In addition to temporal constraints and estimates generated by deontic constraints, it is necessary to take into account other temporal constraints such as relative deadlines as well as temporal estimates. Note that the relative deadline constraints can be imposed by various resource constraints i.e. a resource cannot be shared and has to be used by a single action at the time.

To visualise temporal constraints and estimates we propose a simple concept of a time map (as depicted by Figure 8). Time map depicts *temporal constraints applicable to roles in the community*. Nodes of this map correspond to the time reference points such as beginning and end points of individual actions. Arcs are labeled by a temporal operator and a relative time value that correspond to the time distance between two nodes. Some nodes have a deadline constraint defined. Arcs used to represent temporal constraints are visualised as darker than temporal estimates. The following depicts an example of a time map.

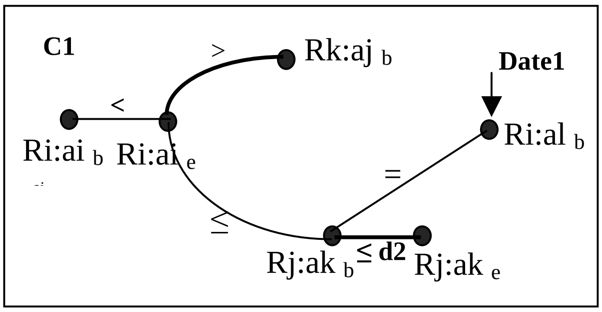


**Figure 8 An example of time map for contract C1**

The next step in contract preparation is to schedule individual actions i.e. to determine their expected/prescribed beginning and end time and duration of individual actions. This step is very important because if a schedule cannot be found that means that some temporal and deontic constraints cannot be satisfied. Note that the role window does specify the time period during which an action must/should or could start, however it does not specify when exactly within that time period the action will occur. Thus, role windows are not sufficient for scheduling of individual actions.

In a very simple contract a schedule can be easily determined manually. For more complex contracts it is necessary to use algorithms such as Floyd-Warschall all pair shortest algorithm introduced in [4].

After the e-contract is prepared i.e. all temporal and deontic contraints are specified and verified and a schedule is determined the next step is contract negotiation. In this process deontic constraints as well as temporal constraints and estimates can be changed (by the negotiating parties).

Thus, role windows (both individual and summary) as well as time maps can be used as decision support tools for if-then analysis. Because every time when a value of a temporal attribute is changed, or a role is assigned a different action, it is necessary to repeat the process of verification of deontic and temporal consistency and scheduling of individual actions. Note that the above introduced concepts of role windows and time maps can be also used for monitoring purposes during contract execution. However, monitoring is out of the scope of this paper.

## 4. Related Work

A B2B Enterprise Model introduced in [5] is used as a basis of e-contracting architecture in this paper. Key elements of the original enterprise model are: *contract repository* (used to store standard contract forms and templates), *contract notary* used to store signed instances of standard contracts forms), *contract* monitor (that enables monitoring of the business interactions governed by a contract) and *contract enforcer* (used to ensure the compliance with contract terms). This model is currently being implemented using BizTalk technology and XML messaging (for more details see [1]).

In order to support formal modeling of contracts as described in this paper, we argue that the above architecture has to be extended to include an additional component called *contract verifier.* This decision support component needs to provide tools for construction and analysis of role windows and time maps, verification of temporal and deontic consistency and automatic scheduling of individual actions according to the contract specification.

In the area of policy-based management for distributed systems, the related work includes Role-based Management framework by (Lupu and Sloman, 1999). The authors also use time when specifying policies, however we consider more types of temporal constraints. Furthermore, the authors consider modality conflicts to detect inconsistencies in policy specification which may arise when two or more policies with modalities of opposite sign (e.g. authorized and forbidden) refer to the same subjects, targets and actions. In our work, to verify deontic consistency, we take into account not only different modalities, roles and actions but also the associated temporal constraints. Because it is important to verify whether the same role is both obliged and prohibited to perform the same action within the *same time interval*.

Other related work in the area of e-contracting includes EU-funded COSMOS project (see [7]) that provides the set of services that facilitate the use of e-contracts. Much of the system deals with lower-level communication and representation issues rather than more contract-specific issues.

## 5. Conclusion

E-contracting is becoming increasingly needed as more and more business are moving on-line. As technologies for contract management are becoming available, the focus is shifting from technology to modeling issues.

The main objective of this paper was to describe some aspects of formal modeling of e-contracts. This process consists of formal modeling and verification of deontic constraints, verification of deontic consistency of an e-contract, formal modeling and visualisation of temporal constraints and estimates, verification of temporal consistency of an e-contract and finally scheduling of the required actions. The paper also introduced visualisation concepts such as role windows and time maps that can be used not only for verification and scheduling but also as decision support tools during contract negotiation.

Our current and future work includes several extensions and applications of the proposed formalism. We plan to include support for resource modeling and management issues. We also plan to utilize this formalism to facilitate automated monitoring and decision support during contract execution. For this purpose, the concepts of role window and time maps introduced in this paper will be further extended.

## ACKNOWLEDGMENT

The work reported in this paper has been funded in part by the Cooperative Research Centres Program through the Department of the Prime Minister and Cabinet of the Commonwealth Government of Australia.

## 12. References

[1] Herring, C. and Milosevic, Z. (2001), “Implementing B2B Contracts Using BizTalk”, Proc. of *HICSS-34 Conference,* Hawaii, Honolulu.

[2] ISO/IEC WD 15414. (1998) Open Distributed Processing – Reference Model – Enterprise Viewpoint.

[3] Cole, J. et al. (2001), “Author Obliged to Submit Paper before 4 July: Policies in an Enterprise Specification”, *Policy2001 workshop*, Bristol, UK, January.

[4] Dechter, R., Meiri, I., Pearl, J. (1991), “Temporal constraint networks”, *Artificial Intelligence*, 49, 61-95.

[5] Milosevic, Z. and Bond, a. (1995), “Electronic Commerce on the Internet: What is still missing?”, *Proc. of the 5th Conference of the Internet Society*, pg. 245-254, Honolulu.

[6] Lupu, E. and Sloman, M. (1999) “Conflicts in Policy-based Distributed Systems Management”, *IEEE Transactions on Software Engineering - Special Issue on Inconsistency Management.*

[7] Griffel, F. et al. (1998), “Electronic Contracting with COSMOS – How to Establish, Negotiate and Execute Electronic Contracts on the Internet”, *EDOC’98 Workshop*, La Jolla, California, USA.