Legal Knowledge and Information Systems
E. Francesconi et al. (Eds.)

doi:10.3233/FAIA220460

An Automata-Based Formalism for Normative Documents with Real-Time

Stefan CHIRCOP [a,1], Gordon J. PACE [a] and Gerardo SCHNEIDER [b]
[a] *Department of Computer Science, University of Malta, Msida, Malta*
[b] *Department of Computer Science and Engineering, University of Gothenburg, Gothenburg, Sweden*

Abstract. Deontic logics have long been the tool of choice for the formal analysis of normative texts. While various such logics have been proposed many deal with time in a qualitative sense, i.e., reason about the ordering but not timing of events, it was only in the past few years that real-time deontic logics have been developed to reason about time quantitatively. In this paper we present timed contract automata, an automata-based deontic modelling approach complementing these logics with a more operational view of such normative clauses and providing a computational model more amenable to automated analysis and monitoring.

Keywords. real-time logic, deontic logic, normative systems, legal contracts

1. Introduction

The duality of automata-based and logic-based formalisms has long been acknowledged in computer science. The former excelling on providing a visual and operational model, while the latter provide a more compositional and denotational view, the two approaches complement each other. Automata-based approaches are ideal for describing models and for automated analysis, logic-based approaches for writing specifications. In earlier work, we have developed contract automata [8], an automata-based approach to a class of deontic logics and proved equivalence of expressiveness between the two [3]. The class of logics contract automata addressed was that of logics which have a qualitative notion of time, i.e., caring about the ordering of action occurrence but not the actual real-time elapsed between them.

Since then, various real-time deontic logics and calculi have been proposed as ways to express real-time clauses in normative documents. In this paper we borrow from work done in real-time automata-based formalisms, particularly timed automata [1] to extend our previous work on contract automata to deal with real-time aspects and norms. We present *timed contract automata*, which can be seen either as contract automata [8] enriched with real-time constraints, or as timed automata [1] enriched with deontic notions. In order to evaluate the effectiveness of our approach, we present a use-case describing an airline-passenger agreement [5].

[1]Corresponding Author: Department of Computer Science, University of Malta; E-mail: stefan.chircop.15@um.edu.mt.

Related work. The starting point of our work are *contract automata* [2], an untimed operational approach to the formalisation of normative systems. Contract automata are finite state machines where states are annotated with permissions, prohibitions and obligations over single actions, and transitions are labelled with actions. The formalism allows for the encoding of reparations. In our work we take contract automata and extend it with clocks, inspired by *timed automata* [1], resulting in a combination of the two.

Different partial formalisations of normative specifications with time have been given by Governatori et al., for instance in [6,7], in the context of *defeasible logic*, classifying timed deontic actions depending on their duration and scope: *achievement*, *maintenance* and *punctual*. In our work we consider *achievement* obligations, *maintenance* permissions and treat prohibitions as *maintenance* obligations to *avoid* an event.

Finally, we refer the reader to the paper [4] for a discussion on the difficulties and challenges of defining and monitoring a formal timed normative language.

2. Timed Contract Automata

Timed contract automata regulate the behaviour of multiple parties over time. They bring together deontic notions from *contract automata* [9] and real-time notions from *timed automata* [1] in order to express how the actions performed by the different parties over time reflect their expected behaviour moving forward.

The underlying notion of time we will use throughout the paper is a continuous one, ranging over the non-negative reals $\mathbb{R}^+ \cup \{0\}$, and denoted by $\mathbb{T}$. In keeping with timed automata, we allow automata to use multiple clocks from a denumerable set $\mathbb{C}$. We will allow for the resetting of clocks, all of which will be assumed to run at the same rate. We will assume throughout the existence of a global clock (which cannot be reset) $\gamma \in \mathbb{C}$. A *clock valuation*, of type $\mathbb{C} \to \mathbb{T}$, gives a snapshot of the values carried by the clocks. We use $val_{\mathbb{C}}$ to denote the set of of all clock valuations. We will need to write conditions over the values of clocks, for which we will use *clock predicates* ranging over $val_{\mathbb{C}} \to \mathbb{B}$ and which, given a clock valuation, return whether the predicate is satisfied. We use $pred_{\mathbb{C}}$ to refer to the set of all such predicates.

We will write $v \gg \delta$ to denote the advancement of clock values in v by δ to be defined as $\lambda c \cdot v(c) + \delta$. A valuation v is said to exceed the latest satisfaction of a clock predicate τ, written $v > \max(\tau)$, if for any non-negative progress in time $\delta \in \mathbb{T}$, the predicate is not satisfied $\neg(\tau(v+\delta))$.[2] Finally, we also defined the overriding of a clock valuation by another $v \oplus v'$ to be defined to be the clock valuation which returns the value given by v' when defined, or that given by v otherwise.

The underlying deontic approach used in these automata is a party-aware and action-based one, i.e., they talk about what, for instance, a *party ought-to-do* as opposed to the *state the party ought-to-be in*. Timed contract automata will be parametrised by the set $\mathbb{P}$ of parties involved and actions $\mathbb{A}$.

In order to identify violations to permissions, we will assume that attempts to perform an action are observable. In order to do so, given a set of actions $\mathbb{A}$, we will write $\mathbb{A}_{attempted}$ to denote the enriched alphabet $\mathbb{A} \cup \{a_{attempt} \mid a \in \mathbb{A}\}$.

Our semantics will be based on an observed timed trace of actions (and attempted ones). A *timed trace* over a set of parties $\mathbb{P}$ and alphabet $\mathbb{A}$ is a finite sequence of ob-

[2] This does not take into account the possibility of clocks being reset.

served events — where an *event* is an action with associated party and timestamp (as per the global clock): $seq(\mathbb{P} \times \mathbb{A}_{attempted} \times \mathbb{T})$ such that the timestamp progresses with each observed action, i.e., given timed trace ts, then for any i and j such that $i < j$, if $ts(i) = (p_i, a_i, t_i)$ and $ts(j) = (p_j, a_j, t_j)$ then $t_i < t_j$.

In keeping with the action-based approach, norms refer to particular parties and actions. Timed contract automata allow a range of norms to be expressed, ranging over obligations, prohibitions and permissions. Real-time norms have been discussed in the literature with various useful semantics. For instance, consider granting John the permission to access a digital resource over the coming 10 minutes. Two possible semantics can be given to this permission — a one-time semantics, i.e., John can access the resource over the coming 10 minutes, but uses up that permission in doing so, or a continuous semantics, i.e., John can access that resource any number of times over the coming 10 minutes. Neither is intrinsically correct (or incorrect), since it depends on what sort of permission one intends to give John. Rather than replicate the discussion of which norms are appropriate in the real-time context, we limit ourselves to a number of norms which can, however, be extended if we want to adopt other norms as part of timed contract automata in the future.

Definition. The norms we will use are parametrised by: (i) the party on which the norm applies; (ii) temporal constraints when the norm applies; and (iii) the action on which the norm applies. For a given set of actions $\mathbb{A}$, a set of clocks $\mathbb{C}$ and parties $\mathbb{P}$, the set of possible norms, denoted by $\mathbb{D}$, covers objects of the form: $\mathscr{N}_\tau(p : a)$ where: (i) $\mathscr{N}$ is norm ranging over permission $\mathscr{P}$, prohibition $\mathscr{F}$ and obligation $\mathscr{O}$; (ii) $p \in \mathbb{P}$ is the party to whom the norm applies; (iii) $\tau \in pred_{\mathbb{C}}$ is a clock predicate indicating when the norm applies; and (iv) $a \in \mathbb{A}$ is the action which the norm refers to.

Syntax. Time contract automata can be seen as a combination of timed automata [1] in that they allow the use of real-time clocks and clocked events, and contract automata [8] in that they use states as norm-carrying modes. Similar to timed automata, (i) we allow for multiple clocks (all progressing at the same rate); (ii) transitions are guarded by clock conditions; and (iii) transitions may reset any number of clocks to particular values.

Similar to contract automata, states are associated with a number of norms. However, the temporal modality offered by the automaton can interrupt deontic norms. For instance, being in a particular state may prohibit John from reading a file as long as clock c does not exceed 10 minutes. However, a transition is taken from that state when c still reads 2 minutes, thus exiting that state. Whether the prohibition is discarded (since we are no longer in that state) or persists (since the temporal constraint has not yet run out) is a choice one has to make. On one hand, we can see the norms in the states as being *active* as long as we are in the state, or as being *enacted* when we enter the state. Both forms can be useful, and we keep both forms of *ephemeral* and *persistent* norms.

Definition. A *timed contract automaton* C, over parties $\mathbb{P}$, actions $\mathbb{A}$ and that uses clocks $\mathbb{C}$, is a tuple $\langle Q, q_0, \rightarrow, \rightarrow_{timeout}, pers, eph \rangle$ where: (i) Q is the set of states, with $q_0 \in Q$ being the initial state; (ii) $\rightarrow \subseteq Q \times (\mathbb{P} \times \mathbb{A} \times pred_{\mathbb{C}} \times val_{\mathbb{C}}) \times Q$ is the transition relation labelling each transition with a party and action which trigger it, a clock predicate which guards it, and a (possibly partial) clock valuation to reset any number of clocks upon taking the transition; (iii) $\rightarrow_{timeout} \subseteq Q \times (\mathbb{C} \times \mathbb{T} \times val_{\mathbb{C}}) \times Q$ is the timeout transition relation with resets, enabling leaving a state when a particular timer reaches a particular value and resetting any number of clocks; and (iv) $pers$, $eph \in Q \rightarrow 2^{\mathbb{D}}$ are functions, which given a state, return the sets of persistent and ephemeral norms active when in that

state. We will write $q \xrightarrow{p{:}a \mid \tau \mapsto \rho} q'$ to denote $(q, (p, a, \tau, \rho), q') \in \rightarrow$ and $q \xrightarrow{c=t \mapsto \rho}_{timeout} q'$ to denote $(q, (c, t, \rho), q') \in \rightarrow_{timeout}$.

A timed contract automaton is *well-formed* if (i) the global clock is never reset, i.e., if $q \xrightarrow{p{:}a \mid \tau \mapsto \rho} q'$, then $\gamma \notin \mathrm{dom}(\rho)$; and (ii) the automaton is *deterministic*, i.e., an observed action only allows for one transition to fire: if $q \xrightarrow{p{:}a \mid \tau_1 \mapsto \rho_1} q_1$ and $q \xrightarrow{p{:}a \mid \tau_2 \mapsto \rho_2} q_2$, then either $q_1 = q_2$ and $\rho_1 = \rho_2$, or for any clocks valuation v, $\neg(\tau_1(v) \wedge \tau_2(v))$. In the rest of the paper we will assume that timed contract automata are well-formed.

Timed Semantics. In order to define the semantics, we start by defining the configuration of a timed contract automaton. This stores all relevant information about the automaton during an execution, namely (i) current state; (ii) current value of clocks; and (iii) active persistent and ephemeral deontic norms.

Definition. A *configuration* of a timed contract automaton $M = \langle Q, q_0, \rightarrow, pers, eph \rangle$ has type: $Q \times val_{\mathbb{C}} \times \mathbb{D} \times \mathbb{D}$. We write $Conf_M$ to denote the set of all configurations, leaving out M when clear from the context. The initial configuration $conf_0$ is $(q_0, \lambda c \cdot 0, pers(q_0), eph(q_0))$.

Based on this, we can define the temporal progression of configurations upon observing a new event (p,a,t). Recall that the time t of the event in the trace will be according to the global clock γ. We define the configuration relation $conf \xRightarrow{p:a,t} conf'$ showing how a configuration evolves, breaking it down into (i) a temporal step $conf \xRightarrow[temp]{p:a,t} conf'$; and (ii) a deontic step $conf \xRightarrow[norm]{p:a,t} conf'$. Firstly, we allow progression along a matching timeout transition using the following rule:

$$\frac{q \xrightarrow{c=t' \mapsto \rho}_{timeout} q' \qquad (q', (v \gg \delta) \oplus \rho, P \cup pers(q'), eph(q')) \xRightarrow[temp]{p:a,t} C}{(q, v, P, E) \xRightarrow[temp]{p:a,t} C} \; \delta = t' - v(c),\ t' - v(\gamma) > \delta$$

Note that if a timeout transition fires before the event time, that transition is taken, and we must move to the destination state of the timeout transition, updating the persistent and ephemeral norms accordingly. If no timeout transition matches the antecedent of the rule above, we can consume the event as per the following rule:

$$\frac{q \xrightarrow{p{:}a \mid \tau \mapsto \rho} q'}{(q, v, P, E) \xRightarrow[temp]{p:a,t} (q', (v \gg \delta) \oplus \rho, P \cup pers(q'), eph(q'))} \; \delta = t - v(\gamma),\ \tau(v \gg \delta)$$

If no transition matches the rule above, we progress by remaining in the same state:

$$\frac{}{(q, v, P, E) \xRightarrow[temp]{p:a,t} (q', v \gg \delta, P, E)} \; \delta = t - v(\gamma)$$

Deontic Semantics. We can now turn to the deontic aspect of the semantics of timed contract automata. The semantics of the individual norms is characterised using a satisfaction and a violation predicate which decides how an observed action interacts with that norm, allowing to extend the progress relation to address configuration changes from a deontic both in the case of a violation or otherwise.

$$vio(\mathscr{P}_\tau(p:a), (p:a_{attempt}, v)) \stackrel{df}{=} \tau(v) \qquad sat(\mathscr{P}_\tau(p:a), (p':a', v)) \stackrel{df}{=} v > \max(\tau)$$
$$vio(\mathscr{F}_\tau(p:a), (p:a, v)) \stackrel{df}{=} \tau(v) \qquad sat(\mathscr{F}_\tau(p:a), (p':a', v)) \stackrel{df}{=} v > \max(\tau)$$
$$vio(\mathscr{O}_\tau(p:a), (p':a', v)) \stackrel{df}{=} v > \max(\tau) \qquad sat(\mathscr{O}_\tau(p:a), (p:a, v)) \stackrel{df}{=} \tau(v)$$

$$\frac{\exists n \in P \cup E \cdot vio(n,\ (p : a, v \gg \delta))}{(q,\ v,\ P,\ E) \xrightarrow[norm]{p:a,\ t} \bot}\ \delta = t - v(\gamma)$$

$$\frac{\neg\exists n \in P \cup E \cdot vio(n,\ (p : a, v \gg \delta))}{(q,\ v,\ P,\ E) \xrightarrow[norm]{p:a,\ t} (q,\ v,\ active(P,\ (p : a,\ v)),\ active(E,\ (p : a,\ v)))}\ \delta = t - v(\gamma)$$

Note that *active* removes satisfied norms given an observed event, i.e., $active(N,\ (p : a,\ v))$ is defined to be $\{n \in N \mid \neg sat(n,\ (p : a,\ v))\}$. In addition, we will have rules to ensure that a violation $\bot$ will not evolve further, i.e., $\bot \xrightarrow[temp]{p:a,\ t} \bot$ and $\bot \xrightarrow[norm]{p:a,\ t} \bot$.

Combining Temporal and Deontic Semantics. We can combine these relations by putting them in sequence, i.e., $c \overset{e}{\Rightarrow} c'$ is defined to mean that there exists configuration c'' such that $c \xrightarrow[norm]{e} c'' \xrightarrow[temp]{e} c'$. The residual configuration after a well-formed timed trace can be computed using the transitive closure of this combined relation, starting from the initial configuration $conf_0$. A timed trace *ts violates* the timed contract automaton if and only if $conf_0 \overset{ts}{\Rightarrow} \bot$.

3. Use Case: Airport Regulations

We consider a use case from the literature expressing airport regulations, and based on the Madrid Barajas airport regulations [5]. Due to space restrictions, we only present a selection of the regulations, as shown below. The parties involved are (i) the passenger p; and (ii) the airline company ac.

1. The passenger is permitted to *check in* (*ci*) 2 hours before take-off. However, the check in desk is closed half an hour before take-off, and the passenger is prohibited from checking in from that point onwards.
2. The passenger is then obliged to *present their boarding pass* (*bp*) within 5 minutes, after which they have another 5 minutes to *produce their passport* (*ppt*).
3. Having done so, the passenger is permitted 10 minutes to *dispose of any liquids in their hand luggage* (*dlhl*), and *present it to the staff* (*prs*). The passenger is also prohibited from *carrying any weapons* (*wps*).
4. In the meantime, should the airline company find reason to *stop the passenger* (*stop*), then they must put their *hand luggage in the hold* (*hold*) within 20 minutes, as well as *call security* (*sec*) within 1 minute.
5. Should the staff *find no issues* (*clear*), then the passenger is permitted to *board the plane* (*board*) within 90 minutes since producing the passport.

We can express this snippet of the regulations using the timed contract automaton shown in Fig. 1. Note that we label transitions as $p : a \mid \tau \mapsto \rho$ to denote the transition tagged by party p, action a, clock constraints τ and resets ρ. Also note that we write $\top$ for the clock constraint which always returns true, and we express resets as assignments. Ephemeral and persistent norms are tagged individually for clarity.

Note that the automaton uses much of the structure of the original text. On the other hand, it provides a more operational view of the agreement, and is more amenable to automated analysis.

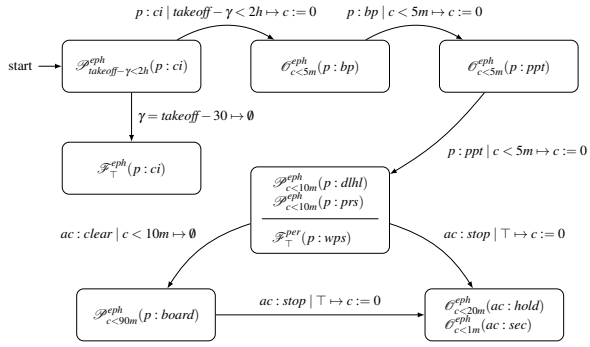

Figure 1. Automaton for the airport regulation use case.

4. Conclusions

In this paper we have presented timed contract automata, combining contract automata with timed automata to enable the operational modelling of real-time normative agreements. We do not envisage such automata as the specification language in which agreements can be modelled. Logic-based deontic approaches are more effective in that they provide better structure. Instead, we see timed contract automata as the operational model in which one can reason more effectively about real-time agreements. We are currently looking at formally correct compilation from deontic logics into timed contract automata, and algorithms for efficient analysis of timed contract automata. We already inherit many decidability (and non-decidability) results from timed automata, and the interesting question is how far we can push analysis such as conflict analysis and model checking of timed contract automata, and their use in runtime verification.

References

[1] Rajeev Alur and David L. Dill. A Theory of Timed Automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

[2] Shaun Azzopardi, Gordon J. Pace, and Fernando Schapachnik. Contract automata with reparations. In *JURIX'14*, pages 49–54. IOS Press, 2014.

[3] Shaun Azzopardi, Gordon J. Pace, Fernando Schapachnik, and Gerardo Schneider. Contract automata - an operational view of contracts between interactive parties. *Artif. Intell. Law*, 24(3):203–243, 2016.

[4] Shaun Azzopardi, Gordon J. Pace, Fernando Schapachnik, and Gerardo Schneider. On the specification and monitoring of timed normative systems. In *RV'21*, volume 12974 of *LNCS*. Springer, 2021.

[5] Alberto García, María-Emilia Cambronero, Christian Colombo, Luis Llana, and Gordon J. Pace. Themulus: A timed contract-calculus. In *MODELSWARD'20*, pages 193–204. SciTePress, 2020.

[6] Guido Governatori, Joris Hulstijn, Régis Riveret, and Antonino Rotolo. Characterising deadlines in temporal modal defeasible logic. In *AI'07*, pages 486–496, 2007.

[7] Guido Governatori and Antonino Rotolo. Justice delayed is justice denied: Logics for a temporal account of reparations and legal compliance. In *CLIMA XII*, pages 364–382, 2011.

[8] Gordon J. Pace and Fernando Schapachnik. Contracts for Interacting Two-Party Systems. In *FLACOS'12*, volume 94 of *ENTCS*, 2012.

[9] Gordon J. Pace and Fernando Schapachnik. Types of rights in two-party systems: A formal analysis. In *JURIX'12*, pages 105–114, 2012.