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, Chalmers University of Technology, 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, and 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, complete or otherwise. In earlier work, we have developed contract automata [20], 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 but not the actual real-time elapsed between them e.g. enabling one to write an anti-money laundering regulation of the form *"The owner of a bank account is prohibited from withdrawing more than $1000 unless they first provide source-of-income documentation"*, but not *"The payment service provider is obliged to close down an account if its owner has not submitted source-of-income documentation within 30 days of opening it"*.

Since then, various real-time deontic logics and calculi have been proposed as ways of dealing with actual time in a deontic setting, e.g., [13,16,12,17] (see Section 2 for a discussion of such work), allowing the expressing of real-time clauses in normative documents. In this paper we borrow from work done in real-time automata-based formalisms,

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

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 [20] enriched with real-time constraints, or as timed automata [1] enriched with deontic notions. In order to evaluate the effectiveness of timed contract automata to express real-time normative models, we present a use-case from the literature describing an airline-passenger agreement [13].

2. Related Work

The starting point of our work are *contract automata* [2,3], 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]. A timed automaton is a finite automaton extended with clock variables which increase their value as time elapses (all at the same rate). Clocks can be used as guards on transitions and as invariants on locations, thus restricting the timed behaviour of the automaton. Clocks can be reset to zero when taking a transition. Essentially, our formalism may be seen as combination of contract automata with timed automata.

Different partial formalisations of normative specifications with time have been given by Governatori et al., for instance in [14,15,16], in the context of *defeasible logic*. In [14,15] a classification is presented concerning 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.

Related to our operational approach are C-O Diagrams, first introduced by Martinez et al. [19] and further extended in [18,10,11,12], with the goal to provide a formal visual representation of normative systems. The formalism has a timed automata semantics, via a complex encoding. The basic component of a C-O diagram is a *box* for each norm (obligation, permission, prohibition) or action, containing a (timed) enabling condition, a timed constraint for the norm or action to happen and, if needed, a pointer to a reparation. (Reparations are also "boxes".) Complex C-O diagrams may be built through the composition of simpler boxes via *refinements*: sequence, conjunction and choice. The formalism has been used to specify (and partially verify) normative documents using a CNL [8,9,7]. The main difference with our approach is that we take a more operational perspective.

Themulus [13,5,6] is a *timed contract-calculus* where contracts are viewed as first class objects amenable to formal reasoning, independent of the system they are enacted in. The calculus includes permissions, prohibitions and obligations as well as reparations in case of violations. There is a recursive operator allowing to express recurrent norms. Themulus provides an operational bisimulation-based approach to contract analysis and comparison. In contrast, our semantic approach is more explicitly modal, allowing for ephemeral and persistent norms which can be convoluted to express in a more structured logic or calculus.

The timed dyadic deontic logic TDDL was introduced by Kharraz et al. [17] as an extended timed version of a dyadic deontic logic with conditional obligations, permis-

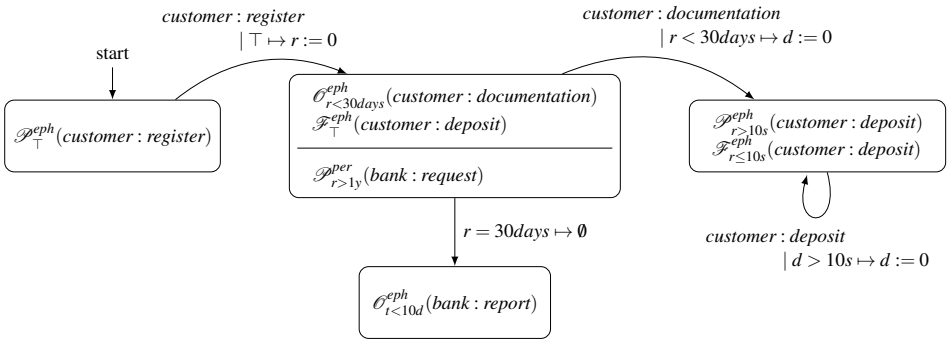
sions, and obligations, and with a reparation operator for representing contrary-to-duties and contrary-to-prohibitions. The logic includes a sequence operator allowing the definition of sequences of individual norms. Timed intervals allows to express deadlines of norms. The logic has a trace semantics capturing both satisfaction and violation of norms. The main difference with our approach is its denotational character (ours being operational) and the use of ephemeral norms and multiple clocks.

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.

3. Timed Contract Automata

Timed contract automata regulate the behaviour of multiple parties over time. They bring together deontic notions from *contract automata* [21] and real-time notions from *timed automata* [1] in order to express the way the parties' rights evolve over time based on the actions performed by the different parties.

A small example of a timed contract automaton is shown below, depicting the regulations imposed on a bank and its customer, in which: (i) Initially, the customer is permitted to register, upon which they must provide source-of-income documentation within 30 days. Until they provide such documentation, they are prohibited from depositing funds, and if they do not submit this information, the bank is obliged to report it within 10 days of the period elapsing. (ii) Customers with a verified account are permitted to deposit funds but not more frequently than once every 10 seconds. (iii) The bank is permitted to request updated documentation 1 year after registration.



The notation will be explained in detail in the rest of this section, but note that the states of the automaton depict modes of the agreement and are annotated with temporal deontic norms of two forms: ephemeral (*eph*) which hold as long as we are in that state and persistent (*per*) which persist even if we leave that state. Norms (permission $\mathscr{P}$, prohibition $\mathscr{F}$ and obligation $\mathscr{O}$) are tagged by the temporal constraint over clocks (with $\top$ indicating no temporal constraint), the party and the action. The transitions are annotated as $p : a \mid \tau \mapsto \rho$ — party p, action a, temporal constraint τ and clock resets ρ. Timeout transitions are also allowed (see the transition pointing downwards), annotated by a clock and value when the timeout will trigger, and a set of resets to perform. In this example, three clocks r (reset when registration occurs), t (reset when the customer fails to provide documentation) and d (reset when a deposit occurs).

Time 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.

For example, the clock predicate τ which is satisfied when clock c_1 exceeds 10, and in which the value of clock c_2 is at least twice the value of c_3 is written as: $\tau(v) = v(c_1) > 10 \wedge v(c_2) \geq 2v(c_3)$. For convenience, we will write these making the valuation v implicit, i.e., $\tau = c_1 > 10 \wedge c_2 \geq 2c_3$.

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.

Actions 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 of parties involved $\mathbb{P}$ and actions $\mathbb{A}$.

In order to be able to identify violations to permissions, we will assume that attempts to perform an action are observable (as used, for instance, in [22]). 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 observed 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 $t_i < t_j$.

Norms 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 long been discussed in the literature (e.g., [4,6,14,12,10,17]), with various interesting semantics identified as useful. 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.

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

Buying a one month subscription to an online music service would give John the continuous permission, whereas buying a token which can be used to play a game once would correspond to one-time permission. Rather than replicate the discussion (already done elsewhere) 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 1 *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.*

For example, the obligation on John to pay before clock c exceeds 10 minutes would be written as: $\mathscr{P}_{c\leq 10}(John : pay)$. Although in theory we allow for any clock predicate, in practice we may want to make constraints on the type of predicate to allow e.g. it has to be a decidable function, it does not exhibit Zeno-like behaviour, it has no isolated points, etc.

3.1. 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 [20] 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 2 *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 $q \xrightarrow{p:a \mid \tau \mapsto v} q'$ *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* $\gamma \notin dom(\rho)$*; 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.

3.2. Timed Semantics

On the basis of the definition shown in the previous section, we can now give a formal semantics to timed contract automata. We start by giving a timed-trace semantics to the automata, which we will then extend in the next section to deal with the normative aspect.

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 3 *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 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 antecedent of the rule above, we progress remaining in the same state:

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

3.3. 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.

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

It is worth highlighting that we trigger a violation or satisfaction of a norm if the clocks predicate can no longer be satisfied bar with clock resets. For example, an obligation $\mathscr{O}_{c\leq 10}(p:a)$ is violated if enough time passes for c to reach time 10 (despite the fact that the clock could be reset at a later stage). Not doing so would lead to the anomalous situation in which obligations can never be violated and other norms can never be discharged.

Also note that we give a semantics which does not consider an attempt to perform a prohibited action to be a violation (unless the attempt is successful, of course) and analogously, neither does an attempt to perform an action sufficient to discharge an obligation to perform that action. In practice, we would have a strong and a weak form of obligation and prohibition to capture both possibile semantics. For space reasons, we give only one.

We can now define an additional progress relation, addressing configuration changes from a deontic, rather than a temporal one both in the case of a violation or otherwise.

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

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

Note that the *active* function 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 $\perp$ will not evolve further, i.e., $\perp \xRightarrow[temp]{p:a,t} \perp$ and $\perp \xRightarrow[norm]{p:a,t} \perp$.

3.4. Combining Temporal and Deontic Semantics

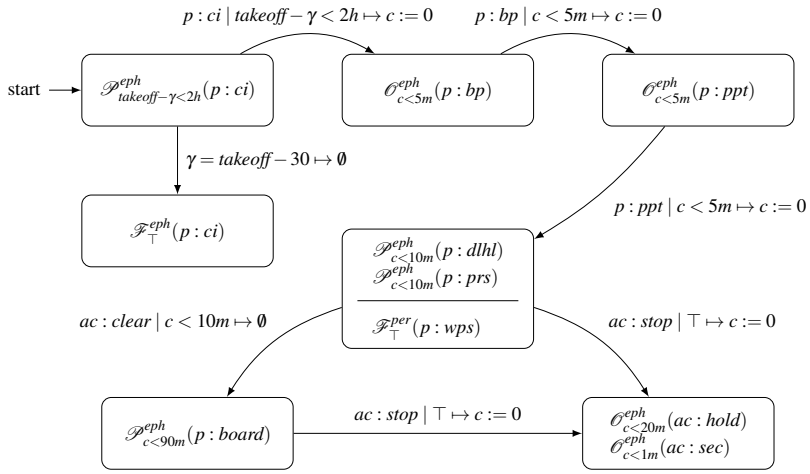
We can combine these relations by putting them in sequence, i.e., $c \stackrel{e}{\Rightarrow} c'$ is defined to mean that there exists configuration c'' such that $c \xRightarrow[norm]{e} c'' \xRightarrow[temp]{e} c'$. The residual configuration after a well-formed timed trace can be computed using 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 $conf_0 \stackrel{ts}{\Rightarrow} \perp$.

4. Use Case: Airport Regulations

In order to illustrate the use of timed contract automata, we consider a use case from the literature expressing airport regulations, and based on the Madrid Barajas airport regulations [13]. Due to space restrictions, we present a selection of the regulations in this paper, as shown in the agreement text below. In order to make the link with the formal description clearer, we annotate actions in the text using italics and follow them by an abbreviation that acts as the action symbol in the formal notation. The parties involved are (i) the passenger (denoted by p in the formal notation); and (ii) the airline company (denote by *ac* in the formal notation).

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*), as well as *present it to the staff* (*prs*). The passenger is also hereby 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 following timed contract automaton. 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.



It is worth noting that apart from the global clock γ we use just one additional clock c to keep track of relative time. In more complex use cases we have explored, particularly when one has clock guarded persistent norms, one typically has to use multiple clocks to keep track of different relative temporal starting points.

What can be noted from this example, is that the automaton description of the clauses uses much of the structure of the original text. On the other hand, the automaton provides a more operational view of how the agreement advances, and is more easily amenable to automated analysis. We are, in fact, currently looking at automated means of analysing timed contract automata for conflicts and for automated compliance verification with constant overheads.

5. 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 *Legal Knowledge and Information Systems - JURIX 2014: The Twenty-Seventh Annual Conference, Jagiellonian University, Krakow, Poland, 10-12 December 2014*, pages 49–54, 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] María Emilia Cambronero, Luis Llana, and Gordon J Pace. A calculus supporting contract reasoning and monitoring. *IEEE Access*, 5:6735–6745, 2017.

[6] María-Emilia Cambronero, Luis Llana, and Gordon J. Pace. Timed contract compliance under event timing uncertainty. In *JURIX'17*, volume 302 of *Frontiers in Artificial Intelligence and Applications*, pages 33–38. IOS Press, 2017.

[7] John J. Camilleri, Mohammad Reza Haghshenas, and Gerardo Schneider. A Web-Based Tool for Analysing Normative Documents in English. In *The 33rd ACM/SIGAPP Symposium On Applied Computing –Software Verification and Testing track (SAC-SVT'18)*. ACM, 2018. To appear.

[8] John J. Camilleri, Gabrielle Paganelli, and Gerardo Schneider. A cnl for contract-oriented diagrams. In *Fourth Workshop on Controlled Natural Language (CNL 2014)*, volume 8625 of *LNCS*, pages 135–146. Springer, 2014.

[9] John J. Camilleri and Gerardo Schneider. Modelling and analysis of normative documents. *Logical and Algebraic Methods in Programming*, 91:33–59, 2017.

[10] Gregorio Díaz, María Emilia Cambronero, Enrique Martínez, and Gerardo Schneider. Timed Automata Semantics for Visual e-Contracts. In *5th International Workshop on Formal Languages and Analysis of Contract-Oriented Software (FLACOS'11)*, volume 68 of *Electronic Proceedings in Theoretical Computer Science*, pages 7–21, Málaga, Spain, September 2011.

[11] Gregorio Díaz, María Emilia Cambronero, Enrique Martínez, and Gerardo Schneider. Specification and Verification of Normative Texts using C-O Diagrams. *IEEE Transactions on Software Engineering*, 99:1, 2013.

[12] Gregorio Díaz, María-Emilia Cambronero, Enrique Martínez, and Gerardo Schneider. Specification and verification of normative texts using C-O Diagrams. *Transactions on Software Engineering*, 40(8):795–817, 2014.

[13] Alberto García, María-Emilia Cambronero, Christian Colombo, Luis Llana, and Gordon J. Pace. Themulus: A timed contract-calculus. In *Proceedings of the 8th International Conference on Model-Driven Engineering and Software Development - MODELSWARD*, pages 193–204. INSTICC, SciTePress, 2020.

[14] Guido Governatori, Joris Hulstijn, Régis Riveret, and Antonino Rotolo. Characterising deadlines in temporal modal defeasible logic. In *AI 2007: Advances in Artificial Intelligence, 20th Australian Joint Conference on Artificial Intelligence, Gold Coast, Australia, December 2-6, 2007, Proceedings*, pages 486–496, 2007.

[15] Guido Governatori and Antonino Rotolo. Justice delayed is justice denied: Logics for a temporal account of reparations and legal compliance. In *Computational Logic in Multi-Agent Systems - 12th International Workshop, CLIMA XII, Barcelona, Spain, July 17-18, 2011. Proceedings*, pages 364–382, 2011.

[16] Guido Governatori, Antonino Rotolo, and Giovanni Sartor. Temporalised normative positions in defeasible logic. In *ICAIL'05*, pages 25–34, 2005.

[17] Karam Younes Kharraz, Martin Leucker, and Gerardo Schneider. Timed dyadic deontic logic. In *The 34th International Conference on Legal Knowledge and Information Systems (JURIX'21)*, volume 346 of *Frontiers in Artificial Intelligence and Applications*, pages 197–204. IOS Press, 2021.

[18] Enrique Martínez, María-Emilia Cambronero, Gregorio Díaz, and Gerardo Schneider. Timed automata semantics for visual e-contracts. In *Workshop on Formal Languages and Analysis of Contract-Oriented Software*, volume 68 of *Electronic Proceedings in Theoretical Computer Science*, pages 7–21, 2011.

[19] Enrique Martínez, Gregorio Díaz, María-Emilia Cambronero, and Gerardo Schneider. A Model for Visual Specification of E-contracts. In *The 7th IEEE International Conference on Services Computing (IEEE SCC'10)*, pages 1–8, Miami, USA, July 5–10 2010. IEEE Computer Society.

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

[21] Gordon J. Pace and Fernando Schapachnik. Types of rights in two-party systems: A formal analysis. In *Legal Knowledge and Information Systems - JURIX 2012: The Twenty-Fifth Annual Conference, University of Amsterdam, The Netherlands, 17-19 December 2012*, pages 105–114, 2012.

[22] Filipe A. A. Santos, Andrew J. I. Jones, and José Carmo. Action concepts for describing organised interaction. In *30th Annual Hawaii International Conference on System Sciences (HICSS-30), 7-10 January 1997, Maui, Hawaii, USA*, pages 373–382. IEEE Computer Society, 1997.