Chain of events: Modular Process Models for the Law

Søren Debois[1], Hugo A. López[2,4], Tijs Slaats[2], Amine Abbad Andaloussi[3], and Thomas T. Hildebrandt[2⋆]

[1] Department of Computer Science, IT University of Copenhagen, Denmark
debois@itu.dk
[2] Department of Computer Science, Copenhagen University, Denmark
{lopez,slaats,hilde}@di.ku.dk
[3] Technical University of Denmark, Kgs. Lyngby, Denmark
amab@dtu.dk
[4] DCR Solutions A/S, Denmark

Abstract. In this paper, we take technical and practical steps towards the modularisation of compliant-by-design executable declarative process models. First, we demonstrate by example how the specific language of timed DCR graphs is capable of modelling complex legislation, with examples from laws regulating the functioning of local governments in Denmark. We then identify examples of law paragraphs that are beyond these modelling capabilities. This incompatibility arises from subtle and—from a computer science perspective—non-standard interactions between distinct paragraphs of the law, which must then become similar interactions between model fragments. To encompass these situations, we propose a notion of *networks of processes*, where the processes are allowed to interact and regulate their interaction through the novel mechanisms of *exclusion* and *linking*. Networks are parametric in the underlying process formalism, allowing interactions between processes specified in arbitrary and possibly distinct trace-language semantics formalisms as the individual models. Technically, we provide a sufficient condition for a good class of network compositions to realise *refinement* of the constituent processes. Finally, parts of the theoretical framework (networks and exclusion) have been implemented by our industry partners, and we report on a preliminary evaluation suggesting that inter-model synchronisation is indeed both necessary and helpful in practical modelling scenarios.

Keywords: law, compliance by design, process modelling, refinement

1 Introduction

Casework is often governed by law, e.g, in municipal governments or in the finance sector. In these settings, adherence to the law—legal compliance—is an

⋆ Work supported by the Innovation Fund Denmark project *EcoKnow* (7050-00034A), the Danish Council for Independent Research project *Hybrid Business Process Management Technologies* (DFF-6111-00337), and the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement BehAPI No.778233.

essential part of "correctness". However, in systems supporting casework, the law is rarely a first-class object, and guarantees of compliance are hard to come by.

This problem is compounded by the practical difficulty that compliance is a property of a *collection* of IT system. A modern European municipal government of even a medium-sized city will have a system landscape rivalling enterprises in complexity and heterogeneity: Disparate systems acquired and updated at disparate schedules over decades. To reason formally about compliance in such a setting, it is not enough to know that any single system is in compliance, we must know that the composite overall system is in compliance.

The law itself is *also* a collection of interacting entities; typically paragraphs or sections. Research into formalisation of law has established as paramount the need for a trustworthy and understandable correspondence between the constructs in the formal notation on the one hand, and the natural language texts that express the rules in the original legal sources on the other [6, 14]. Such a correspondence is necessary to ensure that guarantees provided by the formal language will, quite literally, "hold up in court"; but also to allow for updating models when the law inevitably changes. The quest for such correspondences have given rise to the *isomorphism principle* [5,6] (see also discussion in [7]) that formal models of the law must be in one-one correspondence with the structure of that law—e.g., that each paragraph in a law text corresponds uniquely to a model fragment in the formal specification.

This paper studies models for the law with the aim of directly constructing declarative, executable workflow specifications from it. E.g., when the law states that "the parents must consent to a government interview with their child", the executable workflow specification must have activities "consent" and "interview", and we must be able to prove that in the model, the latter is always preceded by the former. We specifically consider the Danish Consolidation Act on Social Services [1], which regulates in minute detail the operations of Danish Municipalities. We formalise fragments of this law in the Timed Dynamic Condition Response graphs (DCR graphs) declarative modelling language [11, 18, 20], as they are already actively being used to to create executable models of law to for public digital case management systems [25].

We find that while individual paragraphs of the law are straightforward to model, *interactions* between paragraphs are difficult or impossible to model if one is to take the isomorphism principle seriously. To address this shortcoming, we propose the meta-formalism of "Networks" for expressing such interactions via the novel constructs of *linking* and *exclusion*. These constructs allow (a) one-to-many interactions between constraints in the underlying processes and (b) selectively disregarding such constraints in interactions.

The meta-formalism of Networks is *independent* of the exact formalisms used to specify individual process/paragraphs, i.e., it is a hybrid process notation [31]. It is only required that each component notation has a labelled transition system semantics. Thus, it is technically possible for a network to combine processes/paragraphs formalised in disparate notations, e.g., some as DCR, some as DECLARE [3, 30], some as finite automata, and some as BPMN [29].

As a key technical result, we give a sufficient condition for Networks to give rise to *refinement* in the sense of [11, 34]. This theorem has been verified in Isabelle/HOL; the formalisation is available online [32]. Definitions, Lemmas, and Theorems etc. in this paper that have been so verified are marked out with a filled-in, like the one at the end of this paragraph: ■

Moreover, parts of the theoretical framework (networks and exclusion) have been implemented by our industry partners, and we report on interviews with practitioners who find the notions of inter-model synchronisation indeed both necessary and helpful in practical modelling scenarios.

In summary, we make the following contributions.

1. We demonstrate the use of timed DCR graphs to model excerpts of a real law, showing examples of both sections that can be modelled straightforwardly and those that require interaction.
2. We define a notion of Networks with novel concepts of "exclusion" and "linking" tailored to the complex and unusual requirements that modelling the law under the isomorphism principle poses on compositionality.
3. We show how this notion of compositionality formally gives a syntactic means of achieving refinement in the sense introduced in [11] of models expressed in possibly distinct formalisms.
4. We report on a preliminary qualitative evaluation of an implementation of DCR networks with exclusion as part of the a process engine used to digitalise administrative processes in municipal governments.

Altogether, the present paper takes significant steps, both technical and practical, towards achieving compliant-by-design executable declarative process models of government workflows.

Related Work. We share motivation with the study of Compliant-by-Design business processes [15]. Here, formal languages expressing laws and regulations is an active line of research, and a variety of approaches exist, e.g., logics [15–17], Petri Nets [24], and declarative process languages [10]. We are unaware of Compliance-by-Design work that include references as language primitives.

The relationship between natural language specifications and (declarative) business processes has been recently studied in the BPM community with works for Declare [2], deontic logics [12] and DCR graphs [26]. While these works apply NLP techniques to identify rules between process activities, they do not consider the inter-dependencies between rules. The exception is [35], that identifies subsumption, redundancy and conflict between rules. The present work takes a different tack, by providing a mechanism to modularise rules.

An approach similar to linking has been proposed for Petri Net variants in [13, 22, 23]. Here process fragments, modelled as Petri nets, are loosely coupled through event and data dependencies. Our approach is different in that we employ a declarative process language (DCR graphs), we link event executions instead of data, and fragment composition is based on multicast synchronisation. Finally, several works in logic programming have studied modularity and

composition (see [8] for an overview). Networks and links resemble union and overriding union operators in modular logic programs.

2 Timed Dynamic Condition Response Graphs

We briefly recall Timed DCR graphs as introduced in [20]. Informally, a DCR graph comprises a set of events E, a *marking* assigning state to each event, and a set of inter-event relations. Together, the two determine (a) whether a given event is *enabled* for execution, (b) how such execution would update the marking; and (c) what events are required to happen within what deadlines.

Time is advanced in discrete steps called "ticks", and time spans are measured in integral numbers of such ticks. Deadlines in a timed DCR graph is measured in how many ticks may elapse before some event must happen; when that number is 0, time cannot advance any further without either executing the event or violating the semantics of the DCR graph.

Intuitively, the marking indicates for each event e when (if ever) it was last executed; when (if ever) it must eventually be executed or excluded—its deadline—; and whether the event is currently included or excluded. Excluded events cannot be executed, and are disregarded as obstacles to other events executing.

Similarly, the relations govern enabledness and marking update: A *timed condition* $(e, k, e') \in \rightarrow\bullet$ means that event e' can only execute if event e is excluded or it was previously executed and that the last execution was at least k time units ago. A *timed response* $(e, k, e') \in \bullet\rightarrow$ means that whenever event e executes, it imposes the requirement on e' to either become and stay excluded, or to execute within at most k time units. A *milestone* $(e, e') \in \rightarrow\diamond$ means that event e' can only execute if event e is not currently required to be executed or excluded. An *exclusion* (resp. *inclusion*) relation $(e, f) \in \rightarrow\%$ resp. $(e, f) \in \rightarrow+$ toggles the inclusion state of f to false resp. true whenever e is executed.

All in all, the meaning of a DCR graph is the set of sequences of event executions and time increments it is willing to allow.

We give a brief formal account of timed DCR graphs below; however, the reader who either knows DCR graphs already, or is satisfied to learn by example is invited to skip ahead to the next Section.

Notation Let ω be the set of finite natural numbers and zero. Let ∞ be the set $\omega \cup \{\omega\}$, where we refer to ω as infinity. We write $X \rightharpoonup Y$ for a partial function from X to Y. When $f : X \rightarrow Y$ is a (possibly partial) function, we write $f[x \mapsto y]$ for the function $f' : X \rightarrow Y$ identical to f except $f'(x) = y$. Finally, for a binary relation R, take $e\ R = \{f | (e, f) \in R\}$ and vice versa.

Definition 1. *A timed DCR Graph G is given by a tuple $(E, M, \rightarrow\bullet, \bullet\rightarrow, \rightarrow\diamond, \rightarrow+, \rightarrow\%, L, l)$ where*

1. *E is a finite set of events*
2. *$M \in (E \rightharpoonup \omega) \times (E \rightharpoonup \infty) \times \mathcal{P}(E)$ is the timed marking*
3. *$\rightarrow\bullet \subseteq E \times \omega \times E$, is the timed condition relation*

4. $\bullet\!\!\rightarrow \subseteq E \times \infty \times E$, *is the timed response relation*
5. $\rightarrow\!\!\diamond, \rightarrow\!\!+, \rightarrow\!\!\% \subseteq E \times E$ *are the milestone, include and exclude relations*
6. L *is the set of labels*
7. $l : E \to L$ *is a labelling function, which assigns to each event* e *a label* $l(e)$.

We write the components of a marking M *as* $M = (t_{ex}, t_{re}, In)$. *The* minimal included response deadline $minr_G$ *is defined by* $minr_G = min\{t_{re}(e) \mid t_{re}(e) \in \omega \wedge e \in In\}$.

The marking defines for each event e an integer $k = t_{ex}(r)$ indicating how long ago it was executed or $\bot = t_{ex}(r)$ if not; a deadline $t_{re}(r)$ for the event to be executed or $\bot$; and a boolean In indicating whether the event is "included".

Definition 2. *Let* G *be a timed DCR graph. We say that* the event e is enabled, *writing* $\mathsf{enabled}(M, e)$ *iff*

1. $e \in In$
2. $\forall e' \in In \,.\, (e', k, e) \in \rightarrow\!\!\bullet \implies t_{ex}(e') \neq \bot \wedge k \leq t_{ex}(e')$
3. $\forall e' \in In \,.\, e' \rightarrow\!\!\diamond\, e \implies t_{re}(e') = \bot$

We say that the time-step n is enabled, *writing* $\mathsf{enabled}(M, n)$ *when* $minr_G \geq n$.

That is, for e to be enabled, (1) it must be included; (2) whenever it is conditional upon an included event e' with delay k, then this e' was executed at least k time steps ago; and (3) every included milestone e' for e is not pending. A time-step n is enabled iff no included event has a deadline closer than n time units.

Definition 3. *Let* G *be a timed DCR graph. The effect of* executing *an enabled event* e *in* $M = (t_{ex}, t_{re}, In)$ *is a new marking given by:*

$$\mathsf{effect}_G(M, e) = (t_{ex}[e \mapsto 0],\ t'_{re},\ In \setminus (e \rightarrow\!\!\%) \cup (e \rightarrow\!\!+))$$

where $t'_{re}(f) = min\{k \mid (e, k, f) \in \bullet\!\!\rightarrow\}$ *when* $(e, k, f) \in \bullet\!\!\rightarrow$ *and* $t'_{re}(f) = t_{re}[e \mapsto 0](f)$ *otherwise. Similarly, the result of* advancing time by n time-units *is the new marking given by:*

$$\mathsf{effect}_G(M, n) = ((+n) \circ t_{ex}, (-n) \circ t_{re}, In)$$

where $(+n)$ *respectively* $(-n)$ *denote the function* $\omega_\bot \to \omega_\bot$ *which preserve* $\bot$ *and otherwise takes* k *to* $k + n$ *respectively* $\max(k - n, 0)$.

That is, executing e updates the marking by (i) setting the last-executed time $t_{ex}(e)$ of e to 0 (now); (ii) clearing any existing deadline of e, then setting new deadlines for events with responses from e; and (iii) making not-included all events excluded by e, then making included all events included by e. Similarly, when the time-step n is enabled, we "advance time" by adding n to all executed time-stamps, and subtracting n from all deadlines. (An equivalent variation of this semantics make Timed DCR Graphs finite, see [20] for details.)

Definition 4 (Labelled transition system). *An event or time step* $\alpha \in E \cup \mathbb{N}$ *has a* transition $M \xrightarrow{\alpha} M'$ *iff* $\mathsf{enabled}(M, \alpha)$ *and* $\mathsf{effect}_G(M, \alpha) = M'$*. A run of a graph* G *is a finite or infinite sequence of transitions*

$$G_0 \xrightarrow{\alpha_1} G_1 \xrightarrow{\alpha_2} G_2 \xrightarrow{\alpha_3} \cdots$$

We write $\mathsf{runs}(G)$ *for the set of all possible runs for a graph* G*. An* accepting run *is a run such that for all* $i \leq k$ *and all* $e \in E$*, if* $t^i_{re}(e) \in \omega$ *and* $e \in In^i$*, then there exists* $j > i$ *s.t. either* $e \notin In^j$ *or* $\alpha_i = e$ *Finally, a* trace *is a finite or infinite sequence* $\lambda_1 \lambda_2 \ldots$ *of labels and natural numbers, such that there exists and accepting run* $G_0 \xrightarrow{\alpha_1} G_1 \xrightarrow{\alpha_2} \cdots$ *where* $\lambda_i = l(\alpha_i)$ *or* $\lambda_i = \alpha_i = n \in \mathbb{N}$

Note that in this definition, a trace is a run where events have been replaced with their labels, but time advances (natural numbers) have been left in. The indirection of labels is a source of expressive power; see [11] for details.

We write DCR graphs as $[M]\ R$, where M is the marking and R is a list of relations separated by vertical bars. E.g.:

$$[\mathsf{A} : (7, \mathsf{t}, \bot), \mathsf{B} : (\bot, \mathsf{t}, 3)]\ \mathsf{A} \rightarrow\!\bullet\ \mathsf{B} \mid \mathsf{A} \bullet\!\xrightarrow{10}\ \mathsf{B}$$

Here, the marking $\mathsf{A} : (7, \mathsf{t}, \bot)$ that A was executed 7 time-steps ago, it is currently included (t), and there is no deadline for it ($\bot$). Conversely, in $\mathsf{B} : (\bot, \mathsf{t}, 3)$, we see that B was not executed, but does have a deadline of 3. Formally, the marking $\mathsf{A} : (7, \mathsf{t}, \bot)$ should be read as $t_{ex}(\mathsf{A}) = 7$, $\mathsf{A} \in \mathsf{In}$ is true, and $\mathsf{t}_{re}(\mathsf{A}) = \bot$.

While [20] did not allow multiple distinct deadlines between the same two events, the present notion of DCR graphs relaxes this limitation by preferring the minimum of multiple deadlines. This is to ensure that the above calculus-like notation is always well-defined, i.e., that one can freely write terms such as $\mathsf{A} \bullet\!\xrightarrow{5}\ \mathsf{B} \mid \mathsf{A} \bullet\!\xrightarrow{10}\ \mathsf{B}$.

3 Models of law

We now provide examples of modelling law fragments as DCR graphs. We shall see how DCR graphs neatly model individual sections of a real-world law. In Section 4, we re-use these models when considering references between sections.

As a real-world example, we shall consider fragments of the Danish Consolidation Act for Social Services [33] (CASS). Municipalities in Denmark have processed an average of 9.337,33 CASS cases in the last 3 years. Revising the outcome of these cases is standard procedure: In the first semester of 2018, 887 cases (9,5% of the total cases) were revised, and the outcome of 483 cases (5,1% of the total cases) was changed [27, 28].

3.1 A condition: CASS §63(1)

This paragraph describes the situations in which a municipal government must intervene to provide medical attention for a child:

> CASS §63(1): "*If the custodial parent fails to have a child or young person examined or treated for a life-threatening disease or a disease involving the risk of substantial and permanent impairment of function, the children and young persons committee may decide to undertake such examination or treatment.*"

To model this paragraph as a DCR graph, or in any event-based formalism, we have to understand from this description what are the *events* of the graph. The custodial parent "fail[ing] to have a child or young person examined or treated" is not an event happening at a particular moment in time but rather a continuous state of affairs. The key to modelling this situation is to recognise that the event is not the failure itself, but rather the *formal recognition* by the municipal government that this failure is indeed happening. That decision *is* an event: It happens at a specific moment in time where a document declaring such recognition is signed.

With that in mind, we find in 63(1) the events (that the municipal government formally recognises) a *"failure to undertake examination or treatment"* and *"compulsory examination or treatment"*. How are these events related? The phrasing of the paragraph indicates that only if there is such failure may the government step in: in process terms, the failure is a condition for the compulsory examination or treatment. On the other hand, the phrasing does not *require* the government to act. Altogether, we arrive at the following DCR graph:

$$P_{63(1)} \stackrel{\text{def}}{=} [\mathsf{failure}_{63(1)} : (\bot, \mathsf{t}, \bot),\ \mathsf{exam}_{63(1)} : (\bot, \mathsf{t}, \bot)]\ \mathsf{failure}_{63(1)} \rightarrow\!\bullet\ \mathsf{exam}_{63(1)}$$

In this graph, both events are marked as not executed ($\bot$), included (t) and not pending ($\bot$). The graph has a single condition constraint $\mathsf{failure}_{63(1)} \rightarrow\!\bullet\ \mathsf{exam}_{63(1)}$, indicating that the event $\mathsf{exam}_{63(1)}$ can execute only if $\mathsf{failure}_{63(1)}$ has previously executed. In this section, we shall not distinguish between an event and its label, formally taking $\ell(\mathsf{failure}_{63(1)}) = \mathsf{failure}_{63(1)}$ and $\ell(\mathsf{exam}_{63(1)}) = \mathsf{exam}_{63(1)}$.

Here, subscripts such as "63(1)" are simply part of the events name and do not have any special significance. They will become helpful in the next section, when we need to distinguish between near-identical events in distinct paragraphs/graphs.

Considering the possible runs of $P_{63(1)}$, we find among others the following:

$$\langle \mathsf{failure}_{63(1)}, \mathsf{exam}_{63(1)} \rangle \tag{1}$$

On the other hand the singleton $\mathsf{exam}_{63(1)}$ is not a run: The condition prohibits execution $\mathsf{exam}_{63(1)}$ without first executing $\mathsf{failure}_{63(1)}$.

3.2 Static obligations & inclusion state: CASS §50(3)

This paragraph describes how, during a so-called Child Protection Examination (CPE), the child being considered for protection must in fact be heard.

> CASS §50(3): "*The examination shall include a consultation with the child or young person. The consultation may be dispensed with if factors such as the*

> *maturity of the child or young person or the nature of the case strongly suggests that the decision should be made without prior consultation. If the consultation cannot be conducted, steps shall be taken to establish the views of the child or young person. [...]"*

Again we identify events: a "consultation with the child or young person" ($\mathsf{consult}_{50(3)}$); the declaration that "the consultation may be dispensed with" ($\mathsf{omit}_{50(3)}$); and the (formal documentation of) "the views of the young person or child", established by some other means than consultation ($\mathsf{views}_{50(3)}$).

The text describes a usual course of action of consulting the child, and an alternative for special cases (marked as *"steps shall be taken"*). These situations are usually modelled with an event indicating the declaration of special circumstances, which then excludes the common case and includes the special case:

$$P_{50(3)} \stackrel{\text{def}}{=} [\mathsf{consult}_{50(3)} : (\bot, \mathsf{t}, \omega),\ \mathsf{omit}_{50(3)} : (\bot, \mathsf{t}, \bot),\ \mathsf{views}_{50(3)} : (\bot, \mathsf{f}, \omega)]$$
$$\mathsf{omit}_{50(3)} \rightarrow\!\!\% \ \mathsf{consult}_{50(3)} \mid \mathsf{omit}_{50(3)} \rightarrow\!\!+ \ \mathsf{views}_{50(3)}$$

In $P_{50(3)}$, the marking (line 1) says that $\mathsf{consult}_{50(3)}$ and $\mathsf{views}_{50(3)}$ are initially required to happen eventually (ω). Event $\mathsf{views}_{50(3)}$ is initially not included. While not included it cannot be executed, so the requirement to eventually happen in the marking does not count. The relations (line 2) say that if event $\mathsf{omit}_{50(3)}$ happens, then (left) $\mathsf{consult}_{50(3)}$ is excluded and (right) $\mathsf{views}_{50(3)}$ is included, reversing that state of affairs: While both still technically pending, it is now $\mathsf{consult}_{50(3)}$ which is not included and considered irrelevant, whereas $\mathsf{views}_{50(3)}$ is included and relevant, and thus required to eventually happen.

3.3 Time & obligations: CASS §50(7)

Part of the requirements for the CPE process described in CASS §50 describes how quickly the municipal government should react to reports (typically from medical staff or school staff) that a child may be in need of special support:

> CASS §50(7): "*The examination must be completed within four (4) months after the municipal council has become aware that a child or young person may be in need of special support. Where, exceptionally, an examination cannot be completed within 4 months, the municipal council shall prepare a provisional assessment and complete the examination as soon as possible thereafter.*"

We find three events in this text: "the municipal council has become aware that a child or young person may be in need of special support" ($\mathsf{report}_{50(7)}$), the completion of the examination (line 1, $\mathsf{compl}_{50(7)}$), and the preparation of a provisional assessment (line 4-5, $\mathsf{prov}_{50(7)}$).

We model this paragraph as a graph with relations enforcing the obligation to either complete the examination, or produce a provisional assessment within

4 months from report's reception.

$$P_{50(7)} \stackrel{\text{def}}{=} [\mathsf{report}_{50(7)} : (\bot, \mathsf{t}, \bot), \mathsf{compl}_{50(7)} : (\bot, \mathsf{t}, \bot), \mathsf{prov}_{50(7)} : (\bot, \mathsf{t}, \bot)]$$
$$\mathsf{report}_{50(7)} \bullet\!\!\xrightarrow{\omega} \mathsf{compl}_{50(7)} \mid \mathsf{report}_{50(7)} \rightarrow\!\!\bullet\ \mathsf{prov}_{50(7)}$$
$$\mid \mathsf{report}_{50(7)} \bullet\!\!\xrightarrow{4m} \mathsf{prov}_{50(7)} \mid \mathsf{compl}_{50(7)} \rightarrow\!\!\% \ \mathsf{prov}_{50(7)}$$

That is, if a report is received ($\mathsf{report}_{50(7)}$ is executed), the examination must eventually (ω) be concluded. To model the special case of provisional assessments, we combine deadlines and exclusion: we require that a provisional assessment ($\mathsf{prov}_{50(7)}$) is produced within 4 months after receiving the report, but remove that requirement using an exclusion once the actual examination completes ($\mathsf{compl}_{50(7)}$).

4 Modelling references

We now take legal texts whose specifications introduce referential information.

> CASS §48(1): "*Before the municipal council makes a decision under sections 51, 52, 52a, 56, 57a, 57b, 58, 62 and 63, section 65(2) and (3) and sections 68–71 and 75, the child or young person must be consulted on these matters. The consultation may be dispensed with if the child or young person was consulted immediately beforehand in connection with the performance of a child protection examination, cf. section 50 below. [...]*"

The article continues by describing the circumstances for a consultation to be omitted, under which a guardian must be present etc. We will ignore these details for brevity, and focus on the formal relations between paragraph instead.

There are several such references. First, §48(1) requires a consultation before "making a decision" under a range of other paragraphs, including §63 (see sec. 3.1). Recall that §63(1) tasked the municipal government with undertaking medical examination or treatment for young persons if their custodian failed to do so. For §63, the "decision" referred to in §48(1) refers to the municipal government deciding to (unilaterally) undertake such exams or treatments, that is, executing the event $\mathsf{exam}_{63(1)}$.

Second, §48(1) explicitly states that if a child consultation was made under §50, the consultation otherwise required by §48(1) is not necessary. According to domain specialists, in the situation where both §48 and §50 takes effect, the various consultations required are all considered "the same".

These two kinds of references begets the question: How do we model such references in DCR graphs? We shall see in this section that the latter kind can be considered simply a renaming (since the activities are literally considered "the same"); however, the former kind requires special treatment.

For starters, let us ignore exactly how §48 will be connected to other paragraphs and make a straightforward model of the requirement that before making (certain) decisions about a child, that child must be consulted. In this case, it is straightforward to identify in §48(1) the events "make a decision" ($\mathsf{decide}_{48(1)}$)

and "consult the child" ($\mathsf{consult}_{48(1)}$). Notice how $\mathsf{consult}_{48(1)}$ is not the same as the $\mathsf{consult}_{50(3)}$ in $P_{50(3)}$—this is where the subscripts become helpful. In this case, the model simply contains a condition, stipulating the requirement that the consultation must come before the decision:

$$P_{48(1)} \stackrel{\text{def}}{=} [\mathsf{consult}_{48(1)} : (\bot, \mathsf{t}, \bot),\ \mathsf{decide}_{48(1)} : (\bot, \mathsf{t}, \bot)]\mathsf{consult}_{48(1)} \rightarrow\!\bullet\ \mathsf{decide}_{48(1)} \quad (2)$$

It is tempting to think that we can model this reference by simply *identifying* the event $\mathsf{decide}_{48(1)}$ in $P_{48(1)}$ with event $\mathsf{exam}_{63(1)}$ in $P_{63(1)}$. However, this will not be sufficient, as the decision in §48(1) must *also* be identified with other decisions in the other paragraphs listed (51, 52, 52a and so forth). By transitivity, we would identify them all, but that is non-sensical: the decision to remove a child from the home in §58 is obviously not identical the decision to conduct a medical examination in §63(1). *Those two things are not at all the same.*

However, if proceedings are underway for the same child for *both* of §58 and §63(1) simultaneously, then the consultation mentioned in §48(1) applies for both of them. That means that there should be only *one* such consultation, simultaneously catering to *all* the relevant proceedings.

Altogether, we find that we cannot identify all decisions mentioned in §48(1), however, we must identify the consultations for those decisions in order to maintain a strict correspondence with the law; in order to uphold the isomorphism principle [5, 6]. In DCR terms, we have a set of events in distinct graphs (the decisions), each of which is conditional on the same precondition, specified in a distinct other graph. To capture this idea, we introduce *networks*.

4.1 Networks

Networks formalise a notion of "synchronising process models". While we intend to use them with DCR graphs as the underlying process model—and this is how our industry partner is using them—they are intrinsically formalism agnostic: Any formalism with trace-based semantics can be used as the basic processes, and there is no requirement that all underlying processes are specified in the same formalism.

We abstract the underlying formalism into the following notion of a *process notation*. Assume a fixed universe $\mathcal{U}$ of actions.

Definition 5. *A* process notation $A = (\mathcal{P}, \mathsf{excluded}, \mathsf{step})$ *comprises a set* $\mathcal{P}$ *of process models; a function* $\mathsf{excluded} : \mathcal{P} \to 2^{\mathcal{U}}$*, and a function* $\mathsf{alph} : \mathcal{P} \to 2^{\mathcal{U}}$*; and a transition predicate* $\mathsf{step} : \mathcal{P} \times \mathcal{U} \times \mathcal{P}$*. We require that* $(P, l, Q) \in \mathsf{step}$ *implies both* $l \in \mathsf{alph}(P)$ *and* $\mathsf{alph}(P) = \mathsf{alph}(Q)$*, and if also* (P, l, Q') *then* $Q = Q'$*, that is,* step *is* action-deterministic*.*

Intuitively, alph gives a finite bound on the actions a process may exhibit, and we require this bound to be preserved by step-transitions. Similarly, excluded tells us which actions are excluded in a given process; this set is allowed to change as the process evolves.

$$
\begin{array}{rlll}
R, S ::= & P & \text{process notation} & \mid \quad l \triangleright l_1, \cdots, l_n.\ R \quad \text{link} \\
& \mid R \parallel S & \text{network parallel} & \mid \quad 0 \quad \text{unit} \\
\\
\beta ::= & l \mid \triangleright \beta & \text{(regular/limited) action} &
\end{array}
$$

Fig. 1. Syntax of Networks

DCR graphs with injective labelling is a process notation; in this notation "actions" are DCR trace labels.

Lemma 6. *Take $\mathcal{P}$ to be the set of timed DCR graphs with labels in $\mathcal{U}$ and injective labelling functions. Let* $\mathsf{excluded}$ *be the function which given a timed DCR graph G with events E, marking M, and labelling l returns the set of labels of events of E that are not in In, that is,* $\mathsf{excluded}\, G = \{l(e) \mid e \in E \setminus In\}$. *Finally take $(G, l, G') \in \mathsf{step}$ iff there exists some event $e \in E$ s.t. $\ell(e) = l$ and $G \xrightarrow{e}\!\!\!\!\rightarrow G'$. Then $(\mathcal{P}, \mathsf{excluded}, \mathsf{step})$ is a process notation.*

Note that because of the assumption that the labelling functions are injective, (1) the step predicate is action deterministic, and (2) it is not possible have distinct events e, f where $\ell(e) = \ell(f)$ yet $e \in In$ but $f \notin In$. That is, if $l \in \mathsf{excluded}\, G$, then the graph G has exactly one event labelled l, and that event is excluded.

Network themselves are vaguely reminiscent of CSP [21], and are similar to the notion of networks for DCR graphs of [19]. However, they differ radically from both with the introduction of limited actions, exclusion, and links.

The key features of Networks is synchronisation on limited and unlimited actions. Intuitively, an unlimited action is a "real" action, exhibited by an underlying process. Conversely, a limited action indicates that while the network does not wish to independently execute that action, it is willing to follow along if someone else does. Limited actions allow a network to deny actions to other networks, by refusing to engage in them.

We use this mechanism to formalise a notion of linking, where a single label exhibited by one process is considered a required synchronisation partner for *multiple distinct* actions in other processes, but will not independently exhibit that action. This construct will be helpful in modelling paragraphs of the law like §48(1), which imposes constraints on multiple other paragraphs.

Notation. Network actions are formed by tagging an underlying action $l \in \mathcal{U}$ as either "limited" or "unlimited". We write limited actions $\triangleright l$ and unlimited ones simply l. For either, we define the function γ to extract the underlying process action, $\gamma(l) = \gamma(\triangleright l) = l$. For two network actions β_1, β_2 with the same underlying action $\gamma(\beta_1) = \gamma(\beta_2) = l$ we define their combination $\triangleright l \sqcup \triangleright l = \triangleright l$, $\triangleright l \sqcup l = l$, $l \sqcup \triangleright l = l$, and $l \sqcup l = l$—that is, the unlimited action "wins".

The syntax of *Networks* is defined in Figure 1. A network R is a collection of possibly linked processes. We present the semantics Networks in Figure 3. The

$$\mathsf{alph}(N) = \begin{cases} \emptyset & \text{if } N = 0 \\ \mathsf{alph}(P) & \text{if } N = P \\ \{l_1, \ldots, l_n\} \cup (\mathsf{alph}(M) \backslash \{l\}) & \text{if } N = l \triangleright l_1, \ldots, l_n.\ M \\ \mathsf{alph}(N_1) \cup \mathsf{alph}(N_2) & \text{if } N = N_1 \parallel N_2 \end{cases}$$

$$\mathsf{actions}(N) = \mathsf{alph}(N) \cup \{\triangleright x \mid x \in \mathsf{alph}(N)\}$$

Fig. 2. Alphabet and labels of a DCR network

$$\frac{(P, l, Q) \in \mathsf{step}}{P \xrightarrow{l} Q}\ [\text{N-PROC}] \qquad \frac{l \in \mathsf{excluded}\, P}{P \xrightarrow{\triangleright l} P}\ [\text{N-EXCL}]$$

$$\frac{R \xrightarrow{\beta} S \qquad \gamma(\beta) = l \qquad 1 \leq i \leq n}{l \triangleright l_1, \ldots, l_n.\ R \xrightarrow{\triangleright l_i} l \triangleright l_1, \ldots, l_n.\ S}\ [\text{N-LINK}]$$

$$\frac{R \xrightarrow{\beta} S \quad \gamma(\beta) \notin \{l, l_1, \ldots, l_n\}}{l \triangleright l_1, \cdots, l_n.\ R \xrightarrow{\beta} l \triangleright l_1, \ldots, l_n.\ S}\ [\text{N-PASSTHRU}]$$

$$\frac{R_1 \xrightarrow{\beta} R_2 \quad S_1 \xrightarrow{\beta'} S_2 \quad \gamma(\beta) = \gamma(\beta')}{R_1 \parallel S_1 \xrightarrow{\beta \sqcup \beta'} R_2 \parallel S_2}\ [\text{N-SYNC}] \qquad \frac{R_1 \xrightarrow{\beta} R_2 \quad \gamma(\beta) \notin \mathsf{alph}(S)}{R_1 \parallel S \xrightarrow{\beta} R_2 \parallel S}\ [\text{N-PAR}]$$

Fig. 3. Transition semantics of networks (symmetric rule for [N-PAR] is elided.)

definition uses the auxiliary notion of the *alphabet* of a network, the set of labels it syntactically mentions, and its *actions*, which is just its alphabet lifted to both unlimited and limited actions. We give these auxiliary definitions in Figure 2.

We briefly explain the rules of Figure 3. In [N-PROC] we see that the network which is just a single process in some notation exhibits the actions of that process. In [N-EXCL] we see that this network *also* may exhibit a limited network action for an otherwise excluded underlying process action. Then, a network $l \triangleright l_1, \cdots, l_n.\ R$ has two ways to fire a transition: In [N-LINK], we assume that the underlying network R fires an action l. The linked network then fires, instead of l, any of the actions l_i. However, this linked action is limited, as indicated by the triangle. In [N-PASSTHRU], we assume instead that the action l' has nothing in common with neither l nor the linked actions $l_1, \ldots, l_n$; in this case, the linked network exhibits also the (unlimited) action l'. Finally, the synchronisation rule for parallel composition of networks $R_1 \parallel R_2$ is given in [N-SYNC] and [N-PAR]. In [N-SYNC], we require either both sides to exhibit an action, and the underlying process action of either to be the same. This allows a limited and unlimited action to synchronise, with the composite process exhibiting the "least limited" of the two actions. In [N-PAR], we allow the composite process to exhibit a network action when either does, provided the underlying process action does not occur syntactically in the other.

Definition 7 (Network LTS). *A Network R defines an LTS where states are networks, and there is a transition (R, l, R') whenever $R \xrightarrow{l} R'$. A run of R is a sequence*

$$R = R_1 \xrightarrow{\beta_1} R_2 \xrightarrow{\beta_2} \cdots R_{k-1} \xrightarrow{\beta_{k-1}} R_k$$

A trace $\mathsf{trace}(r)$ *of a run r is the sequence $\beta_1, \ldots, \beta_k$ of actions of the run. The language of the network R is defined as the set traces of those of its runs that are everywhere unlimited (where no $\beta_i = \triangleright l$ for any l), that is,*

$$\mathsf{lang}(R) = \{\mathsf{trace}(r) \mid r \text{ is an unlimited run of } R\}\,.$$

Note that we do not accept limited actions in traces: limited actions cannot happen independently, but require a corroborating un-limited action.

4.2 Modelling with Networks

Using Networks underpinned by timed DCR graphs, we can return to the question how to model inter-paragraph references using the models in Section 3. Note the subtle difference that in that Section 3 we were considering runs, whereas now we are considering traces. The difference is imperceptible since the models of Section 3 all had every event labelled by itself, that is $\ell(e) = e$. For this reason, we allow ourselves in this section to treat "labels" and "events" interchangeably, and we will speak only of events.

For modelling §48(1), we simply *link* the $\mathsf{decide}_{48(1)}$ event with the relevant events from other paragraphs:

$$R \stackrel{\text{def}}{=} P_{63(1)} \parallel \mathsf{decide}_{48(1)} \triangleright \mathsf{exam}_{63(1)}.\ P_{48(1)} \tag{3}$$

This R *does not* admit the trace $\langle \mathsf{failure}_{63(1)}, \mathsf{exam}_{63(1)} \rangle$ even though we saw in (1) that $P_{63(1)}$ does. In R, even if $P_{63(1)}$ allows the action $\mathsf{exam}_{63(1)}$, for the entirety of R to also allow that action, the right-hand side $\mathsf{decide}_{48(1)} \triangleright \mathsf{exam}_{63(1)}.\ P_{48(1)}$ must synchronise via either the [N-SYNC] or [N-PAR] rule. Since both sides of the parallel has $\mathsf{exam}_{63(1)}$ in their alphabet, only [N-SYNC] applies. This means that if the parallel were to have the action $\mathsf{exam}_{63(1)}$, also the right-hand side link would have either of the actions $\mathsf{exam}_{63(1)}$ or $\triangleright\mathsf{exam}_{63(1)}$. Looking at the link rules [N-LINK] and [N-PASSTHRU], we see that the right-side can exhibit $\mathsf{exam}_{63(1)}$ iff $P_{48(1)}$ can exhibit $\mathsf{exam}_{63(1)}$, but this is *not* possible because of the condition from $\mathsf{consult}_{50(3)}$ to $\mathsf{decide}_{48(1)}$ in that graph, see (2).

On the other hand, the network R *does* have the trace

$$\langle \mathsf{failure}_{63(1)}, \mathsf{consult}_{48(1)}, \mathsf{exam}_{63(1)} \rangle\ .$$

We mentioned briefly above §58 which under extreme circumstances allows the government to remove a child from the home. Assuming a process P_{58}, with the event remove_{58} signifying the decision to undertake such removal. We can then build the network where a child is both subject to proceedings §63 and §58.

$$R_2 \stackrel{\text{def}}{=} P_{63(1)} \parallel P_{58} \parallel \mathsf{decide}_{48(1)} \triangleright \mathsf{exam}_{63(1)}, \mathsf{remove}_{58}.\ P_{48(1)} \tag{4}$$

Again, this model would not admit any trace where $\mathsf{exam}_{63(1)}$ or remove_{58} happened without a prior $\mathsf{consult}_{48(1)}$. It would admit various interleavings of the §63 and §58 proceedings and the §48(1) requirements.

The dependency on §50(3). Returning to §50(3), we recall that §48 allowed use of a §50(3) consultation to replace its own, and that practitioners consider both events "the same". This has an obvious model: the one where we simply rename events so that those two identical consultations are identical. This is represented via the syntactical substitution for a free name, here written $P\{e/f\}$:

$$R_3 \stackrel{\text{def}}{=} P_{63(1)} \parallel P_{58} \parallel P_{50(3)} \parallel \mathsf{decide}_{48(1)} \triangleright \mathsf{exam}_{63(1)}, \mathsf{remove}_{58}.\ \left(P_{48(1)}\{\mathsf{consult}_{50(3)}/\mathsf{consult}_{48(1)}\}\right) \quad (5)$$

It is irrelevant whether the renaming happens inside or outside the link construct.

5 A theory of links and refinement

We now relate the networks to the notion of refinement originally introduced for DCR graphs [11] and later generalised to arbitrary process models with trace semantics [34]. Under the right circumstances, networks provide a syntactic mechanism for establishing refinements, thus providing a useful approximation for what in DCR graphs is a computationally hard problem.

Notation Given a sequence s, we define the projection onto a set X as $s|_X$ as simply the (possibly non-contiguous) sub-sequence of s for which each element is in X. We lift this notion to sets of sequences pointwise.

Definition 8 (Network Refinement). *Let R, S be DCR networks. We say that R is a* refinement *of S iff* $\mathsf{lang}(R)|_{\mathsf{alph}(S)} \subseteq \mathsf{lang}(S)$. ■

To establish refinement, we confine the set of actions that may become limited.

Definition 9. *Let N be a network and $X \subseteq \mathcal{U}$ a finite set of labels. We call X* unlimited for N *iff for all β with $\gamma(\beta) \in X$ and $N \xrightarrow{\beta} N'$ for some N' then β is unlimited. X* is globally unlimited for N *if X is unlimited for every M reachable from N.* ■

Lemma 10. *Let $P \in \mathcal{P}$ be a process, and let N be the network consisting exactly of P. Then X is unlimited for N iff in every P' reachable (under the process notation* step*-relation) from P, x is not excluded for all $x \in X$.* ■

We shall see in Lemma 11 below how an unlimited set for a network ensures the existence of an unlimited sub-trace on one side of a parallel composition of networks. The proof relies on action-determinacy of networks, which in turn necessitates that requirement of Definition 5. We conjecture that this requirement of action-determinacy is can be dispensed with at the cost of a somewhat more complicated proof development.

Lemma 11. *Let R_0, S_0 be networks and let $X \subseteq \mathcal{U}$ be a set of labels. Suppose that X is globally unlimited for R_0 and that $\mathsf{alph}(R_0) \cap \mathsf{alph}(S_0) \subseteq X$. Let r be a run of $R_0 \parallel S_0$:*

$$R_0 \parallel S_0 \xrightarrow{\beta_1} R_1 \parallel S_1 \xrightarrow{\beta_2} \cdots \xrightarrow{\beta_{k-1}} R_{k-1} \parallel S_{k-1} \xrightarrow{\beta_k} R_k \parallel S_k \tag{6}$$

Consider the sequence $(\beta_i, R_{i+1})_{1 \leq i < k}$ and take $i_1, \ldots, i_m$ to be the indices identifying a maximal subsequence of this sequence such that $\beta_{i_j} \in X$. Then this subsequence identifies a run r^1 of R_0:

$$R_0 = R_{i_1} \xrightarrow{\beta_{i_1}} R_{i_2} \xrightarrow{\beta_{i_2}} \cdots R_{i_{k'-1}} \xrightarrow{\beta_{i_{k'-1}}} R_{k'} \tag{7}$$

Moreover, $\mathsf{trace}(r^1) = \mathsf{trace}(r)|_{\mathsf{actions}(R_1)}$. ■

Theorem 12. *Let R be a network, assume that X is globally unlimited for R, and that $\mathsf{alph}(R) \cap \mathsf{alph}(S) \subseteq X$. Then $R \parallel S$ is a refinement of R.*

Proof. We must prove that for any trace $\mathsf{trace}(r) \in \mathsf{lang}(R \parallel S)$ we have also $\mathsf{trace}(r)|_{\mathsf{alph}(R)} \in \mathsf{lang}(R)$. By definition of language, every action in $\mathsf{trace}(r)$ is unlimited, so $\mathsf{trace}(r)|_{\mathsf{alph}(R)} = \mathsf{trace}(r)|_{\mathsf{actions}(R)}$. But $\mathsf{trace}(r)|_{\mathsf{actions}(R)}$ is a trace of R by Lemma 11; and projection preserves unlimited-ness, hence we must have $\mathsf{trace}(r)|_{\mathsf{actions}(R)} \in \mathsf{lang}(R)$. ■

Corollary 13. *Let P be a DCR Graph in which all events with a label in $l, \vec{l}$ are included in all reachable markings. Let R be a Network with $\mathsf{alph}(R) \cap \mathsf{alph}(P) = \{l\} \cup \vec{l}$. Then the network $P \parallel l \triangleright \vec{l}$. R refines P.*

What does Theorem 12 and corollary 13 mean for modelling? Looking at R_2 and R_3 from eqs. (4) and (5), it is straightforward to prove using Theorem 12 that both R_2 and R_3 are in fact refinements of $P_{63(1)}$.

Corollary 14. *R_2 and R_3 both refine $P_{63(1)}$.*

This confirms our intuition that (our model of) §48(1) does not in fact modify §63(1) *beyond* adding the requirement to have a consultation before deciding.

6 Implementation and Evaluation

A subset of networks with limited actions, exclusions and network composition but not the link construct, has been implemented by DCR Solutions A/S, a Danish vendor of adaptive case management systems, and used at Syddjurs Municipality (a Danish Municipal government) to implement a administrative processes compliant to CASS in DCR graphs. We report on a **qualitative evaluation** of this subset. The objective of the evaluation is to (a) determine whether DCR networks are *relevant* for practitioners; (b) estimate its *usability* as a modelling construct; and (c) discover its *limitations* as perceived by practitioners.

The evaluation comprises a structured 2 hour interview with a Syddjurs Municipality staff member ("the subject") responsible for developing executable DCR models supporting municipal casework and subsequent analysis of responses. The subject has 3 years experience modelling with DCR models, and had used the DCR Network implementation for at least 2 months. The interview was conducted on February 7th, 2020; interview script, answers, and analysis results are available on-line at [4].

We posed two sets of questions consecutively in a single session. With the first set, we inquired into the background of the expert and the relevance of the investigated approach (a). With the second set, we sought to compare law digitalisation before and after the introduction of DCR networks (b), and to examine the consequences of using the implementation (b,c). In the interview, the subject reflected on his past and current experience with modelling the law.

We analysed a recording of the interview using a qualitative inductive approach supported by grounded theory [9]. With the support of qualitative data analysis tool "Atlas.ti", we applied *initial coding* to identify the pertinent aspects in the interview. We then used *focused coding* to gather the open-codes into more abstract concepts based on their similarity traits. Finally, we used *axial coding* to establish the relationships between the identified codes.

Outcome The *relevance* (a) of DCR networks was justified by a set of domain requirements. The subject highlighted the presence of references in almost all law text and the need to model the interaction between distinct law paragraphs. When reflecting on *past* modelling experience, the subject mentioned the lack of mechanisms to model communication between process models representing distinct law paragraphs. In practice, these mechanisms are needed to automatically trigger related processes and model constraints between events in related models. In the absence of such mechanisms,case-workers must synchronise processes manually, incurring overhead and in some cases leading them to bypass the case management system altogether.

To investigate *usability* (b), the subject was guided to compare his past and current modelling experiences. We note that this interview cannot distinguish usability of the concept of DCR networks from usability of the tooling used by the subject. The subject described areas where the proposed implementation was helpful: the support to automate triggering of events, and for inter-model constraints between them. According to the subject, these mechanisms facilitate modelling the interplay between different processes, and also support process decomposition, making it possible to divide extant models into smaller fragments, each describing a specific law section.

With regards to *limitations* (c), the subject raised the lack of explicit mechanisms to visualise references between events of different processes, making it difficult to track and maintain dependencies between different models. Moreover, the subject felt limited by the absence of simulation tools for DCR networks. Last but not least, he underlined the necessity to extend the existing approach to support data flow between process models.

7 Conclusion

In this paper, we have taken technical and practical steps towards achieving compliant-by-design executable process descriptions. We demonstrated the use of timed DCR graphs to model excerpts of a real law, showing examples of both sections that can be modelled straightforwardly and those that required interaction between models. To that end we defined a notion of compositional Networks with novel concepts of "exclusion" and "linking" tailored to modelling the complex and unusual requirements that modelling the law under the isomorphism principle poses on compositionality. We then showed how this notion of compositionality formally provides a syntactic means of achieving refinement in the sense introduced in [11], here for models expressed in possibly distinct formalisms. This development has been verified in Isabelle/HOL, with theories available on-line [32]. Finally, we reported on a preliminary interview-based evaluation with practitioners, which confirms the necessity of treating references in models. Altogether, we have taken both technical and a practical step towards executable declarative process models of government workflows.

References

1. Bekendtgørelse af lov om social service (Aug 2017), Børne- og Socialministeriet
2. van der Aa, H., di Ciccio, C., Leopold, H., Reijers, H.A.: Extracting Declarative Process Models From Natural Language. In: CAiSE. Springer Heidelberg (2019)
3. Aalst, W.M.P.v.d., Pesic, M.: DecSerFlow: Towards a Truly Declarative Service Flow Language. In: WSFM. pp. 1–23. LNCS (Sep 2006)
4. Andaloussi, A.A.: Evaluation of dcr networks: Interview recordings and full analysis (Feb 2020), http://doi.org/10.5281/zenodo.3724874
5. Bench-Capon, T.J.M.: Deep models, normative reasoning and legal expert systems. pp. 37–45. New York, USA (1989)
6. Bench-Capon, T.J.M., Coenen, F.P.: Isomorphism and legal knowledge based systems. Artificial Intelligence and Law 1(1), 65–86 (Mar 1992)
7. Bench-Capon, T., Araszkiewicz, M., Ashley, K., Atkinson, K., Bex, F., Borges, F., Bourcier, D., Bourgine, P., Conrad, J.G., Francesconi, E., Gordon, T.F., Governatori, G., Leidner, J.L., Lewis, D.D., Loui, R.P., McCarty, L.T., Prakken, H., Schilder, F., Schweighofer, E., Thompson, P., Tyrrell, A., Verheij, B., Walton, D.N., Wyner, A.Z.: A history of AI and Law in 50 papers: 25 years of the international conference on AI and Law. Art. Int. and Law 20(3), 215–319 (Sep 2012)
8. Bugliesi, M., Lamma, E., Mello, P.: Modularity in Logic Programming. The Journal of Logic Programming 19-20, 443–502 (May 1994)
9. Charmaz, K.: Constructing Grounded Theory. Introducing Qualitative Methods series, SAGE Publications (2014)
10. Chesani, F., Mello, P., Montali, M., Riguzzi, F., Sebastianis, M., Storari, S.: Checking compliance of execution traces to business rules. In: BPM. pp. 134–145. Springer (2008)
11. Debois, S., Hildebrandt, T.T., Slaats, T.: Replication, refinement & reachability: complexity in dynamic condition-response graphs. Acta Informatica 55(6), 489–520 (Sep 2018)

12. Dragoni, M., Villata, S., Rizzi, W., Governatori, G.: Combining natural language processing approaches for rule extraction from legal documents. In: AICOL. LNCS, vol. 10791, pp. 287–300. Springer (2017)
13. Eberle, H., Unger, T., Leymann, F.: Process fragments. In: OTM Confederated International Conferences. pp. 398–405. Springer (2009)
14. Gordon, T.F., Governatori, G., Rotolo, A.: Rules and norms: Requirements for rule interchange languages in the legal domain. In: Rule Interchange and Applications. pp. 282–296. Springer Berlin Heidelberg, Berlin, Heidelberg (2009)
15. Governatori, G., Sadiq, S.: The journey to business process compliance. IGI Global (2009)
16. Governatori, G., Rotolo, A.: Norm Compliance in Business Process Modeling. In: Semantic Web Rules. pp. 194–209. LNCS, Springer, Berlin, Heidelberg (Oct 2010)
17. Hashmi, M., Governatori, G., Wynn, M.T.: Normative requirements for business process compliance. In: Australian Symposium on Service Research and Innovation. pp. 100–116. Springer (2013)
18. Hildebrandt, T., Mukkamala, R.R.: Declarative Event-Based Workflow as Distributed Dynamic Condition Response Graphs. In: PLACES. EPTCS, vol. 69, pp. 59–73 (2010)
19. Hildebrandt, T.T., Mukkamala, R.R., Slaats, T.: Safe distribution of declarative processes. In: SEFM 2011. LNCS, vol. 7041, pp. 237–252. Springer (2011)
20. Hildebrandt, T.T., Mukkamala, R.R., Slaats, T., Zanitti, F.: Contracts for cross-organizational workflows as timed dynamic condition response graphs. J. Log. Algebr. Program. 82(5-7), 164–185 (2013)
21. Hoare, C.A.R.: Communicating sequential processes. Communications of the ACM 21(8), 666–677 (1978)
22. Holfter, A., Haarmann, S., Pufahl, L., Weske, M.: Checking compliance in data-driven case management. In: BPM Workshops. pp. 400–411. Springer, Cham (2019)
23. Kindler, E., Petrucci, L.: Towards a standard for modular petri nets: A formalisation. In: Petri Nets. pp. 43–62. Springer (2009)
24. Lohmann, N.: Compliance by design for artifact-centric business processes. Information Systems 38(4), 606 – 618 (2013)
25. López, H.A., Debois, S., Slaats, T., Hildebrandt, T.T.: Business process compliance using reference models of law. In: Procs. of FASE. LNCS, Springer (2020), to appear
26. López, H.A., Marquard, M., Muttenthaler, L., Strømsted, R.: Assisted declarative process creation from natural language descriptions. In: EDOC Workshops. pp. 96–99. IEEE (2019)
27. National Social Appeals Board (Ankestyrelsen): Annual report for the 2018 case process. `https://ast.dk/publikationer/arsopgorelse-2018` (May 2019)
28. National Social Appeals Board (Ankestyrelsen): Appeals Board decisions on the Services Act in Q2 to Q4 2018 (...). `https://bit.ly/3glQOBK` (May 2019)
29. Object Management Group BPMN Technical Committee: Business Process Model and Notation, Version 2.0 (2013)
30. Pesic, M., Schonenberg, H., Aalst, W.M.P.v.d.: DECLARE: Full Support for Loosely-Structured Processes. In: EDOC. pp. 287–287 (Oct 2007)
31. Slaats, T., Schunselaar, D.M.M., Maggi, F.M., Reijers, H.A.: The semantics of hybrid process models. In: CoopIS 2016. pp. 531–551
32. Søren Debois: Formalisation: Modular Process Models for the Law. `https://www.itu.dk/people/debois/thys/ifm20` (Jun 2019)
33. The Danish Ministry of Social Affairs and the Interior: Consolidation Act on Social Services (Sep 2015), `http://english.sm.dk/media/14900/consolidation-act-on-social-services.pdf`, Executive Order no. 1053

34. Tijs Slaats, Søren Debois, Thomas Hildebrandt: Open to Change: A Theory for Iterative Test-Driven Modelling. In: BPM. LNCS, vol. 11080, pp. 31–47. Springer (2018)
35. Winter, K., Rinderle-Ma, S.: Deriving and Combining Mixed Graphs from Regulatory Documents Based on Constraint Relations. In: Procs. of CAiSE. vol. 11483, pp. 430–445. Springer International Publishing, Cham (2019)