Nested Dynamic Condition Response Graphs

Thomas Hildebrandt[1], Raghava Rao Mukkamala[1], and Tijs Slaats[1⋆]

IT University of Copenhagen, Rued Langgaardsvej 7, 2300 Copenhagen, Denmark,
{hilde, rao, tslaats}@itu.dk,

Abstract. We present an extension of the recently introduced declarative process model *Dynamic Condition Response Graphs* (DCR Graphs) to allow *nested subgraphs* and a new *milestone relation* between events. The extension was developed during a case study carried out jointly with our industrial partner Exformatics, a danish provider of case and workflow management systems. We formalize the semantics by giving first a map from Nested to (flat) DCR Graphs with milestones, and then extending the previously given mapping from DCR Graphs to Büchi-automata to include the milestone relation.

1 Introduction

Declarative process models have been suggested by several research groups [1–5,15,16, 18, 19] as a good approach to describe case management and other non-rigid business and workflow processes where it is generally allowed to redo or skip activities, and even dynamically adapt the set of activities and constraints. The rationale is that if a strict sequencing is the exception, then the implicit specification of control flow in declarative models is more appropriate than notations based on explicit control flows such as the (typical use of) Business Process Model and Notation (BPMN) 2.0 [13].

A drawback of the declarative approaches in general, however, is that the implicit definition of the state and control flow makes it more complex to perceive the state and execute the process. To find out what are the next possible activities it is necessary to evaluate a set of constraints defined relatively to the history of the execution.

This motivates finding an expressive declarative process language that allows for a simple run-time scheduling which is easily visualized for the case worker. As a candidate for such a language we recently introduced in [7, 11] a declarative process model called Dynamic Condition Response Graphs (DCR Graphs). The model is a generalization of the classic event structure model for concurrency [20] and is inspired by the Process Matrix model [10, 12] developed by one of our industrial partners Resultmaker, a Danish provider of workflow and case-management systems.

The core DCR Graphs model consists of a set of events and four binary relations between the events: The *dynamic inclusion* and *dynamic exclusion* relations, and the *condition* and *response* relations. The dynamic inclusion and exclusion relations generalize the usual symmetric conflict relation of event structures by splitting it in two

⋆ This research is supported by the Danish Research Agency through a Knowledge Voucher granted to Exformatics (grant #10-087067, www.exformatics.com), the Trustworthy Pervasive Healthcare Services project (grant #2106-07-0019, www.trustcare.eu) and the Computer Supported Mobile Adaptive Business Processes project (grant #274-06-0415, www.cosmobiz.dk).

asymmetric relations: If an event A excludes an event B, written $A \rightarrow\!\!\% \ B$, then B can not happen until after the occurrence of an event C that includes event B, which is written $C \rightarrow\!\!+ \ B$. Similarly, the condition and response relations generalize the usual causal order relation of event structures by splitting it in two relations: If an event B has event A as condition, written $A \rightarrow\!\!\bullet \ B$, then event A must either be currently excluded or have happened for B to happen. Dually, if an event A has event B as response, written $A \bullet\!\!\rightarrow B$, then event B must eventually happen or always eventually be excluded after an occurrence of event A. To express that events are executed by actors with different roles the core model is extended with roles assigned to the events.

In [7] we show that the run-time state of DCR Graphs can be represented as a marking consisting of three sets of events, recording respectively the *executed events*, the *currently included events*, and the *pending response events*, i.e. events that must eventually happen or be excluded. From the marking, it is easy to evaluate if an event can happen (by checking if all its conditions are either executed or excluded) and to verify if the graph is in a completed state (by checking if the set of included pending responses is empty). It is also easy to update the state when executing an event by adding it to the set of executed events, remove the event from the pending response set and add new response events according to the response relation, and include/exclude events in the set of currently included events according to the include/exclude relations. In [7, 11] we express the acceptance condition for infinite runs (no pending response is continuously included without being executed) by giving a map to a Büchi automaton.

In the present paper we describe how to extend the model to allow for *nested sub-graphs* as is standard in most state-of-the art modelling notations. The work was carried out during a case study, in which we are applied *Nested DCR Graphs* in the design phase of the development of a distributed, inter-organizational case management system carried out by our industrial partner, Exformatics, a company that specializes in solutions for knowledge sharing, workflows and document handling.

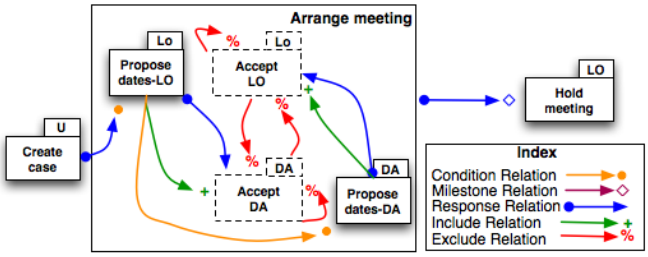

Fig. 1. Nested DCR Graphs with Arrange meeting sub-graph

Fig. 1 shows the graphical notation for nested DCR graphs and illustrates the use of nested sub-graphs in a sub part of the model arising from our case study. The Arrange meeting event represents the arrangement of a meeting between two of the organizations (DA and LO) using the distributed case management system being developed. It has been refined to a sub-graph including four sub events for proposing and accepting dates for the meeting. The dashed boxes indicate that the events Accept DA and Accept LO for accepting meeting dates are initially excluded. Described briefly, when the organization (U) creates a case, it triggers as a response the event Propose dates-LO, representing LO proposing dates for a meeting. This event triggers as a response and includes the event Accept DA, representing DA accepting

the dates. But it also enables that DA can propose other dates, represented by the event Propose Dates-DA. Now, this event triggers as a response and includes the event Accept LO, representing LO accepting the dates. Again, LO may do this, or again propose dates. The proposal of dates may continue forever, and as long as no one accepts there will be a pending response on at least one of the accept events. As soon as one of the accept events happen, they will both be excluded, and there will be none of the included events in the sub-graph having pending responses. This corresponds to the accepting condition for finite runs of DCR graphs [7], and thus intuitively reflects that the sub-graph is in a completed state. Now, we want to express that the event Hold meeting can only be executed when this is the case. To do this, we introduced a new core relation between events called the *milestone* relation. If an event A is a milestone for an event B, written $A \rightarrow\!\!\diamond\ B$, then B can not happen if A is included and required to be executed again (i.e. as a response). The new milestone relation allow us to define nesting as simply a tree structure on events that can be flattened to (flat) DCR Graphs by keeping all atomic events (i.e. events with no sub-events) and letting them inherit the relations defined for their super-events. In particular, the flattening does not introduce new events (in fact it removes all super events) and at most introduce an order of n^2 new relations. Thus, we need not define a new operational semantics for nested DCR Graphs, instead we can make the much simpler extension of the semantics for (flat) DCR Graphs to consider the new milestone relation. It is worth noting that while the milestone relation makes it very direct to express completion of subgraphs, we conjecture that it does not add expressiveness to DCR Graphs.

Related work: Our approach is closely related to the work on ConDec [18, 19]. The crucial difference is that we allow nesting and a few core constraints making it possible to describe the state of a process as a simple marking. ConDec does not address nesting (nor dynamic inclusion/exclusion), but allows one to specify any relation expressible within Linear-time Temporal Logic (LTL). This offers much flexibility with respect to specifying execution constraints. In particular the condition and response relations are standard verification patterns and also considered in [18, 19] (the condition relation is called precedence), and we have used the same graphical notation. However, the execution of a process expressed as LTL (which typically involves a translation to a Büchi-automaton) is more complex and the run-time state is difficult to relate to the original ConDec specification. Moreover, we conjecture that DCR Graphs are as expressive as Büchi-automata, and thus more expressive than LTL. Finally, Nested DCR Graphs relates to the independent (so far unpublished) work on the declarative *Guard-Stage-Milestone* model by Hull, presented in invited talks at WS-FM 2010 and CASCON 2010.

Structure of paper: In Sec. 2 we define Nested DCR Graphs formally, motivated by the case study, and define the mapping to flat DCR Graphs with milestones. In Sec. 3 we then define the lts semantics and the mapping from flat DCR Graphs with milestones to Büchi-automata. The two maps together define the semantics of Nested DCR Graphs. Due to space limitations we refer to [8] and the full version [9] for a detailed description of the case study and tool support. We conclude in Sec. 4 and give pointers to future work.

2 Nested DCR Graphs and Milestones

We now give the formal definition of the Nested DCR Graph model described informally above, which extends the model in our previous work [7] with nesting and the new milestone relation $\rightarrow\diamond$ between events.

Definition 1. *A* Nested Distributed dynamic condition response graph with milestones *is a tuple* $(\mathsf{E}, \rhd, \mathsf{M}, \rightarrow\bullet, \bullet\rightarrow, \rightarrow\diamond, \pm, \mathsf{Act}, l, \mathsf{R}, \mathsf{P}, \mathsf{as})$*, where*

(i) E *is the set of* events
(ii) $\rhd : \mathsf{E} \rightharpoonup \mathsf{E}$ *is a partial function mapping an event to its super-event (if defined), and we also write* $e \rhd e'$ *if* $e' = \rhd^k(e)$ *for* $0 < k$*, referred to as the* nesting relation
(iii) $\mathsf{M} = (E, R, I) \subseteq \mathsf{atoms}(\mathsf{E}) \times \mathsf{atoms}(\mathsf{E}) \times \mathsf{atoms}(\mathsf{E})$ *is the* marking*, containing sets of currently executed events* (E)*, currently pending responses* (R)*, and currently included events* (I)*.*
(iv) $\rightarrow\bullet \subseteq \mathsf{E} \times \mathsf{E}$ *is the* condition *relation*
(v) $\bullet\rightarrow \subseteq \mathsf{E} \times \mathsf{E}$ *is the* response *relation*
(vi) $\rightarrow\diamond \subseteq \mathsf{E} \times \mathsf{E}$ *is the* milestone *relation*
(vii) $\pm : \mathsf{E} \times \mathsf{E} \rightharpoonup \{+, \%\}$ *is a partial function defining the* dynamic inclusion and exclusion *relations by* $e \rightarrow + e'$ *if* $\pm(e, e') = +$ *and* $e \rightarrow \% e'$ *if* $\pm(e, e') = \%$
(viii) Act *is the set of* actions
(ix) $l : \mathsf{E} \rightarrow \mathsf{Act}$ *is a* labeling *function mapping events to actions.*
(x) R *is a set of* roles*,*
(xi) P *is a set of* principals *(e.g. persons or processors) and*
(xii) $\mathsf{as} \subseteq (\mathsf{P} \cup \mathsf{Act}) \times \mathsf{R}$ *is the* role assignment *relation to principals and actions.*

where $\mathsf{atoms}(\mathsf{E}) = \{e \mid \forall e' \in \mathsf{E}. \rhd (e') \neq e\}$ *is the set of* atomic *events.*

We require that the nesting relation $\rhd \subset E \times E$ *is acyclic and that there are no infinite sequence of events* $e_1 \rhd e_2 \rhd \ldots$*. We will write* $e \trianglerighteq e'$ *if* $e \rhd e'$ *or* $e = e'$*, and* $e \trianglelefteq e'$ *if* $e' \rhd e$ *or* $e = e'$*. We require that the nesting relation is consistent with respect to dynamic inclusion/exclusion in the following sense: If* $e \rhd e'$ *or* $e' \rhd e$ *then* $\pm(e, e'') = +$ *implies* $\pm(e', e'') \neq \%$ *and* $\pm(e, e'') = \%$ *implies* $\pm(e', e'') \neq +$*.*

The new elements are the nesting relation $\rhd \subset E \times E$ and the milestone relation $\rightarrow\diamond \subseteq \mathsf{E} \times \mathsf{E}$. The consistency between the nesting relation and the dynamic inclusion/exclusion is to ensure that when we map a nested DCR Graph to the corresponding flat DCR Graph as defined in Def. 2 below, no atomic event both includes and excludes another event. That is, if an event e includes (excludes) another event e'', then any of its super or sub events e' can not exclude (include) the event e''.

The new elements conservatively extend the DCR Graphs defined in [7] in the sense that given a Nested dynamic condition response graph as defined in Def. 1, the tuple $(\mathsf{atoms}(\mathsf{E}), \mathsf{M}, \rightarrow\bullet, \bullet\rightarrow, \pm, \mathsf{Act}, l, \mathsf{R}, \mathsf{P}, \mathsf{as})$ is a (Distributed) dynamic condition response graph as defined in [7]. In particular, the semantics will be identical if both the $\rhd$ map and the milestone relation are empty.

A nested distributed dynamic condition response graph can be mapped to a flat distributed dynamic condition response graph with at most the same number of events. Essentially, all relations are extended to sub events, and then only the atomic events

are preserved. The labelling function is extended by labelling an atomic event e by the sequence of labels labelling the chain of super events starting by the event itself: $e \rhd e_1 \dots e_k \not\rhd$. The role assignment is extended to sequences of actions by taking the union of roles assigned to the actions.

Definition 2. *For a Nested DCR Graph* $G = (\mathsf{E}, \rhd, \mathsf{M}, \rightarrow\bullet, \bullet\rightarrow, \rightarrow\diamond, \pm, \mathsf{Act}, l, \mathsf{R}, \mathsf{P}, \mathsf{as})$ *define the underlying flat DCR Graph as*

$$G^\flat = (\mathsf{atoms}(\mathsf{E}), \mathsf{M}, \rightarrow\bullet^\flat, \bullet\rightarrow^\flat, \rightarrow\diamond^\flat, \pm^\flat, \mathsf{Act}^+, l^\flat, \mathsf{R}, \mathsf{P}, \mathsf{as}^\flat),$$

where $rel^\flat = \trianglerighteq rel \trianglelefteq$ *for some relation* $rel \in \{\rightarrow\bullet, \bullet\rightarrow, \rightarrow\diamond\}$ *and* $\pm(e, e')^\flat = \pm(e_s, e'_s)$ *if* $\pm(e_s, e'_s)$ *is defined and* $e \trianglerighteq e_s$ *and* $e' \trianglelefteq e'_s$ *and* $l^\flat(e_0) = a_0.a_1.a_2 \dots a_k$ *if* $e_0 \rhd e_1 \rhd e_2 \dots \rhd e_k$ *and* $l(e_i) = a_i$ *for* $0 \leq i \leq k$ *and* $\mathsf{as}^\flat(a_0.a_1.a_2 \dots a_k) = \{\mathsf{as}(a_i) \mid 0 \leq i \leq k\}$ *and* $as^\flat(p) = p$ *for* $p \in \mathsf{P}$.

It is easy to see that the size of the relations may increased by an order of at most n^2 where n is the number of atomic events.

3 Semantics

Below we define the semantics of DCR Graphs with milestones by giving a labelled transition semantics and a mapping to Büchi-automata.
Notation: For a set A we write $\mathcal{P}(A)$ for the power set of A. For a binary relation $\rightarrow \subseteq A \times A$ and a subset $\xi \subseteq A$ of A we write $\rightarrow \xi$ and $\xi \rightarrow$ for the set $\{a \in A \mid (\exists a' \in \xi \mid a \rightarrow a')\}$ and the set $\{a \in A \mid (\exists a' \in \xi \mid a' \rightarrow a)\}$ respectively.

Definition 3. *For a dynamic condition response graph with milestones* $G = (\mathsf{E}, \mathsf{M}, \rightarrow\bullet, \bullet\rightarrow, \rightarrow\diamond, \pm, l, \mathsf{Act}, \mathsf{R}, \mathsf{P}, \mathsf{as})$*, we define the corresponding labelled transition systems* $T(G)$ *to be the tuple* $(S, \mathsf{M}, \rightarrow \subseteq S \times \mathsf{Act} \times S)$ *where* $S = \mathcal{P}(\mathsf{E}) \times \mathcal{P}(\mathsf{E}) \times \mathcal{P}(\mathsf{E})$ *is the set of markings of* G *and* $\mathsf{M} = (R, I, E) \in S$ *is the initial marking,* $\rightarrow \subseteq S \times \mathsf{E} \times (\mathsf{P} \times \mathsf{Act} \times \mathsf{R}) \times S$ *is the transition relation given by* $\mathsf{M}' \xrightarrow{(e,(p,a,r))} \mathsf{M}''$ *where*

(i) $\mathsf{M}' = (E', R', I')$ *is the marking before transition*
(ii) $\mathsf{M}'' = (E' \cup \{e\}, R'', I'')$ *is the marking after transition*
(iii) $e \in I$, $l(e) = a$, $p \,\mathsf{as}\, r$, *and* $a \,\mathsf{as}\, r$,
(iv) $\rightarrow\bullet e \cap I' \subseteq E'$,
(v) $\rightarrow\diamond e \cap I' \cap R' = \emptyset$,
(vi) $I'' = (I' \cup e\rightarrow+) \setminus e\rightarrow\%$,
(vii) $R'' = (R' \setminus \{e\}) \cup e\bullet\rightarrow$,
(viii) $E'' = E' \cup \{e\}$

We define a run $(e_0, (p_0, a_0, r_0)), (e_1, (p_1, a_1, r_1)), \dots$ *of the transition system to be a sequence of labels of a sequence of transitions* $\mathsf{M}_i \xrightarrow{(e_i,(p_i,a_i,r_i))} \mathsf{M}_{i+1}$ *starting from the initial marking. We define a run to be accepting if* $\forall i \geq 0, e \in R_i.\exists j \geq i.(e = e_j \vee e \notin I_j)$. *In words, a run is accepting if no response event is included and pending forever, i.e. it must either happen at some later state or become excluded.*

Condition *(iii)* in the above definition expresses that, only events e that are currently included, can be executed, and to give the label (p, a, r) the label of the event must be a, p must be assigned to the role r, which must be assigned to a. Condition *(iv)* requires that all condition events to e which are currently included should have been executed previously. Condition *(v)* states that the currently included events which are milestones to event e must not be in the set of pending responses (R'). Conditions *(vi)*, *(vii)* and *(viii)* are the updates to the sets of included events, pending responses and executed events respectively. Note that an event e' can not be both included and excluded by the same event e, but an event may trigger itself as a response.

If one considers only finite runs then the acceptance condition degenerates to requiring that no pending response is included at the end of the run. If infinite runs are also of interest (as e.g. for reactive systems and LTL) the acceptance condition can be captured by a mapping to a Büchi-automaton with τ-event defined as follows.

Definition 4. *A* Büchi-automaton with τ-event *is a tuple* $(S, s, Ev_\tau, \rightarrow \subseteq S \times Ev_\tau \times S, F)$ *where* S *is the set of* states*,* $s \in S$ *is the* initial state*,* Ev_τ *is the set of* events *containing the special event* τ*,* $\rightarrow \subseteq S \times Ev_\tau \times S$ *is the* transition relation*, and* F *is the set of* accepting states*. A (finite or infinite) run is a sequence of labels not containing the* τ *event that can be obtained by removing all* τ *events from a sequence of labels of transitions starting from the initial state. The run is* accepting *if the sequence of transitions passes through an accepting state infinitely often.*

Since we at any given time may have several pending responses we must make sure in the mapping to Büchi-automata that all of them are eventually executed or excluded. To do this we assume any fixed order of the finite set of events E of the given dynamic condition response graph. For an event $e \in E$ we write $rank(e)$ for its rank in that order and for a subset of events $E' \subseteq E$ we write $min(E')$ for the event in E' with the minimal rank.

Definition 5. For a finite distributed dynamic condition response graph $G = (\mathsf{E}, \mathsf{M}, \rightarrow\bullet, \bullet\rightarrow, \rightarrow\diamond, \pm, \mathsf{Act}, l, \mathsf{R}, \mathsf{P}, \mathsf{as})$ where $\mathsf{E} = \{e_1, \ldots, e_n\}$, marking $\mathsf{M} = (E, R, I)$ and $rank(e_i) = i$, we define the corresponding Büchi-automaton with τ-event to be the tuple $B(G) = (S, s, \rightarrow \subseteq S \times Ev_\tau \times S, F)$ where

- $S = \mathcal{P}(\mathsf{E}) \times \mathcal{P}(\mathsf{E}) \times \mathcal{P}(\mathsf{E}) \times \{1, \ldots, n\} \times \{0, 1\}$ is the set of states,
- $Ev_\tau = (\mathsf{E} \times (\mathsf{P} \times \mathsf{Act} \times \mathsf{R})) \cup \{\tau\}$ is the set of events,
- $s = (\mathsf{M}, 1, 1)$ if $I \cap R = \emptyset$, and $s = (\mathsf{M}, 1, 0)$ otherwise
- $F = \mathcal{P}(\mathsf{E}) \times \mathcal{P}(\mathsf{E}) \times \mathcal{P}(\mathsf{E}) \times \{1, \ldots, n\} \times \{1\}$ is the set of accepting states and
- $\rightarrow \subseteq S \times Ev_\tau \times S$ is the transition relation given by $(\mathsf{M}', i, j) \xrightarrow{\tau} (\mathsf{M}', i, j')$ where

(a) $\mathsf{M}' = (E', R', I')$ is the marking
(b) $j' = 1$ if $I' \cap R' = \emptyset$ otherwise $j' = 0$.

and $(\mathsf{M}', i, j) \xrightarrow{(\mathrm{e,(p,a,r)})} (\mathsf{M}'', i', j')$ where

(i) $\mathsf{M}' = (E', R', I') \xrightarrow{(\mathrm{e,(p,a,r)})} (E'', R'', I'') = \mathsf{M}''$ is a transition of $T(D)$.

(ii) For $M = \{e \in I' \cap R' \mid rank(e) \rangle\ i\}$ let $j' = 1$ if
  (a) $I'' \cap R'' = \emptyset$ or
  (b) $min(M) \in (I' \cap R' \backslash (I'' \cap R'')) \cup \{e\}$ or
  (c) $M = \emptyset$ and $min(I' \cap R') \in (I' \cap R' \backslash (I'' \cap R'')) \cup \{e\}$
  otherwise $j' = 0$.
(iii) $i' = rank(min(M))$ if $min(M) \in (I' \cap R' \backslash (I'' \cap R'')) \cup \{e\}$ or else
(iv) $i' = rank(min(I' \cap R'))$ if $M = \emptyset$ and $min(I' \cap R') \in (I' \cap R' \backslash (I'' \cap R'')) \cup \{e\}$ or else
(v) $i' = i$ otherwise.

We prove that the mapping from the labelled transition semantics to Büchi-automata is sound and complete in the full version of the paper [9].

The formal semantics of DCR graphs mapped to Büchi-automata enabled us to perform model checking and formal verification of processes specified in DCR graphs. The prototype implementation allows us to perform verification of both safety and liveness properties using the SPIN [17] model checker and only verification of safety properties using the ZING [14] model checker. The prototype has also been extended to support runtime verification, for monitoring of properties specified using Property Patterns [6].

4 Conclusion and Future Work

We have given a conservative extension of the declarative process model Distributed DCR Graphs [7] to allow for nested sub-graphs motivated and guided by a case study carried out jointly with our industrial partner. A detailed description of the case study and tool support for DCR Graphs can be found in [8]. The main technical challenge was to formalize the execution and in particular *completion* of sub-graphs. We do this by introducing a new *milestone* relation $A \rightarrow\!\!\diamond\ B$, which blocks the event B as long as there are events in A required to be executed (i.e. required responses). We believe this is the right notion of completeness of nested sub-graphs. First of all, it coincides with the definition of acceptance of finite runs in DCR Graphs [7] recalled in Sec. 3 above. Secondly, its formalization is a simple extension of the labelled transition semantics given in [7, 11] since it is a condition on the set of pending responses already included in the states. Finally, it allows for a nested sub-graph to alternate between being completed and not completed, as is often the case in ad hoc case management. This is not possible in the related ad-hoc sub-process activity in BPMN 2.0. Future work within the TrustCare and CosmoBiz projects, which are the context of the work, includes exploring the expressiveness of DCR Graphs, extending the theory and tools for analysis, verification and model-driven engineering, extending the model to be able to express other relevant features such as multi-instance sub-graphs, time, exceptions, data, types and run-time adaption, i.e. dynamic changes of the model.

References

1. Kamal Bhattacharya, Cagdas Gerede, Richard Hull, Rong Liu, and Jianwen Su. Towards formal analysis of artifact-centric business process models. In *LNCS*, volume 4714, pages 288–304, 2007.

2. Christoph Bussler and Stefan Jablonski. Implementing agent coordination for workflow management systems using active database systems. In *Research Issues in Data Engineering, 1994. Active Database Systems. Proceedings Fourth International Workshop on*, pages 53–59, Feb 1994.
3. David Cohn and Richard Hull. Business artifacts: A data-centric approach to modeling business operations and processes. *IEEE Data Eng. Bull.*, 32(3):3–9, 2009.
4. Hasam Davulcu, Michael Kifer, C. R. Ramakrishnan, and I.V. Ramakrishnan. Logic based modeling and analysis of workflows. In *Proceedings of ACM SIGACT-SIGMOD-SIGART*, pages 1–3. ACM Press, 1998.
5. Alin Deutsch, Richard Hull, Fabio Patrizi, and Victor Vianu. Automatic verification of data-centric business processes. In *Proceedings of the 12th International Conference on Database Theory*, ICDT '09, pages 252–267, New York, NY, USA, 2009. ACM.
6. Matthew B. Dwyer, George S. Avrunin, and James C. Corbett. Property specification patterns for finite-state verification. In *Proceedings of the second workshop on Formal methods in software practice*, FMSP '98, pages 7–15, New York, NY, USA, 1998. ACM.
7. Thomas Hildebrandt and Raghava Rao Mukkamala. Declarative event-based workflow as distributed dynamic condition response graphs. In *Programming Language Approaches to Concurrency and communication-cEntric Software 2010 (PLACES10)*. EPTCS, 2010.
8. Thomas Hildebrandt, Raghava Rao Mukkamala, and Tijs Slaats. Designing a cross-organizational case management system using dynamic condition response graphs. In *Accepted for IEEE International EDOC Conference*, 2011.
9. Thomas Hildebrandt, Raghava Rao Mukkamala, and Tijs Slaats. Designing a cross-organizational case management system using nested dynamic condition response graphs. Technical Report TR-2011-141, IT University of Copenhagen, 2011.
10. Karen Marie Lyng, Thomas Hildebrandt, and Raghava Rao Mukkamala. From paper based clinical practice guidelines to declarative workflow management. In *Proceedings ProHealth 08 workshop*, 2008.
11. Raghava Rao Mukkamala and Thomas Hildebrandt. From dynamic condition response structures to büchi automata. In *Proceedings of 4th IEEE International Symposium on Theoretical Aspects of Software Engineering (TASE 2010)*, August 2010.
12. Raghava Rao Mukkamala, Thomas Hildebrandt, and Janus Boris Tøth. The resultmaker online consultant: From declarative workflow management in practice to LTL. In *Proceeding of DDBP*, 2008.
13. Object Management Group BPMN Technical Committee. Business Process Model and Notation, version 2.0, 2010. http://www.omg.org/cgi-bin/doc?dtc/10-06-04.pdf.
14. Microsoft Research. Zing model checker. Webpage, 2010. http://research.microsoft.com/en-us/projects/zing/.
15. Pinar Senkul, Michael Kifer, and Ismail H. Toroslu. A logical framework for scheduling workflows under resource allocation constraints. In *VLDB*, pages 694–705, 2002.
16. Munindar P. Singh, Greg Meredith, Christine Tomlinson, and Paul C. Attie. An event algebra for specifying and scheduling workflows. In *Proceedings of DASFAA*, pages 53–60. World Scientific Press, 1995.
17. Spin. On-the-fly, ltl model checking with spin. Webpage, 2008. http://spinroot.com/spin/whatispin.html.
18. Wil M. P. van der Aalst, Maja Pesic, and Helen Schonenberg. Declarative workflows: Balancing between flexibility and support. *Computer Science - R&D*, 23(2):99–113, 2009.
19. Wil M.P van der Aalst and Maja Pesic. A declarative approach for flexible business processes management. In *Proceedings of DPM 2006*, LNCS. Springer Verlag, 2006.
20. Glynn Winskel. Event structures. In Wilfried Brauer, Wolfgang Reisig, and Grzegorz Rozenberg, editors, *Advances in Petri Nets*, volume 255 of *Lecture Notes in Computer Science*, pages 325–392. Springer, 1986.