An Operational Semantics for the Extended Compliance Rule Graph Language

David Knuplesch, Manfred Reichert

Ulmer Informatik-Berichte
Nr. 2014-06
Juni 2014
Abstract

A challenge for any enterprise is to ensure conformance of its business processes with imposed compliance rules. Usually, the latter may constrain multiple perspectives of a business process, including control flow, data, time, resources, and interactions with business partners. Like for process modeling, intuitive visual languages have been proposed for specifying compliance rules. However, business process compliance cannot completely be decided at design time, but needs to be monitored during run time as well. In previous work we introduced the extended Compliance Rule Graph (eCRG) language that enables the visual monitoring of business process compliance regarding the control flow, data, time, and resource perspectives as well as the interactions a process has with business partners. This technical report introduces an operational semantics of the eCRG language. In particular, the state of a visual compliance rule is reflected through markings and annotations of an eCRG. The proposed operational semantics not only allows detecting compliance violations at run-time, but visually highlights their causes as well. Finally, it allows providing recommendations to users in order to proactively ensure for a compliant continuation of a running business process.

This work was done within the research project C³Pro funded by the German Research Foundation (DFG) under project number RE 1402/2-1.
1 Introduction

The correctness of business process models has been intensively discussed in literature for more than a decade [1–3]. While earlier work focused on syntactical correctness and soundness constraints (e.g., absence of deadlocks and lifelocks), the compliance of business processes with semantic constraints has been increasingly considered during the last years [4, 5]. Usually, compliance rules stem from domain-specific constraints, e.g., referring to corporate standards, legal regulations or evidenced best practices [6, 7], and need to be ensured in all phases of the process life cycle [8, 9].

Approaches dealing with the compliance of business processes during their execution are covered by the notion of compliance monitoring. In order to detect and report run-time violations of compliance rules, events of running business process instances need to be considered (cf. Fig. 1). Note that two kinds of monitoring need to be distinguished: reactive and proactive monitoring. Regarding reactive monitoring, the process-aware information system only reports a compliance violation once it has occurred. In turn, proactive monitoring aims to proactively prevent potential compliance violations that might occur during the further course of process execution; e.g. by suggesting appropriate tasks that still need to be executed to meet the compliance rule.

A multitude of approaches focusing on compliance monitoring at run-time were introduced during the last years [11–13]. While early suggestions focused on the control flow perspective, later proposals for monitoring compliance considered additional process perspectives as well [14, 15]. In particular, the data, resource, and time perspectives have been addressed. Other approaches, in turn, have focused on the traceability of compliance violations [10, 16]. Furthermore, [5] proposed 10 fundamental compliance monitoring functionalities (CMFs) that may be used to compare existing approaches for monitoring business process compliance. In this context, the authors stated that existing approaches do not provide a satisfactory solution that combines an expressive language with full traceability capabilities [5].

To close this gap, this report introduces an operational semantics for the extended Compliance Rule Graph (eCRG) language. The latter has been introduced in [17, 18] and enables...
the visual monitoring of business process compliance. In particular, the operational semantics annotates eCRGs with text, colors and symbols. In order to deal with compliance rules, which are triggered multiple times during the execution of a business process instance, the operational semantic creates and annotates multiple instances of an eCRG in parallel. The annotated instances of an eCRG not only indicate compliance violations, but may be also utilized for recommending the next process steps (i.e. activities), whose execution will ensure compliance. Furthermore, they provide a suitable basis for compliance metrics. Note that the eCRG language is a powerful visual notation for compliance rules that adequately supports the control flow, data, time, and resource perspectives as well as interactions with business partners. Consequently, this report not only provides the operational basis for monitoring control flow constraints, but enables the monitoring of the latter perspectives as well. Altogether the provided operational semantics provides the basis for

– utilizing the eCRG language for monitoring business process compliance during process execution.
– monitoring the conformance with compliance rules related to any process perspective; i.e., the control flow, data, time, resource perspectives as well as the interactions a process has with business partners.
– dealing with compliance rules, which are are triggered multiple times.
– reasoning about the violation of compliance rules – reactively & proactively.
– specifying compliance metrics and measures.

The remainder of this report is structured as follows: Section 2 discusses related work. Section 3 provides a motivating example. Backgrounds on the extended Compliance Rule Graph (eCRG) language are introduced in Section 4. Section 5 provides an operational semantics for the eCRG language. In particular this operational semantics formally specifies the transitions between the states of compliance rules. In turn, Section 6 evaluates and classifies the compliance of an eCRG. Furthermore, the specification of compliance metrics and measures is introduced as well as the provision of recommendations to users. Finally, Section 7 concludes the paper and provides an outlook on future research.
2 Related Work

For a decade, business process compliance has increasingly gained attention and several surveys have been published in the meantime (e.g., [19, 8, 20, 21, 5]). Along this trend, interest in compliance monitoring and continuous auditing [12] has grown as well. For example, [11] enriches process models with a semantic layer of internal controls. In [13, 22], in turn, the detailed architectures of an online auditing tools (OLAT) are described. An OLAT allows monitoring the operations of an organization in detective, corrective and preventive modes. The spectrum of techniques applied for compliance monitoring is wide spread: Behavioural profiles [23] utilize ordering relations in this context, whereas Supervisory Control Theory [24] prevents users from performing actions leading to compliance violations. Furthermore, visual declarative constraints [25], which are transformed into Event Calculus and Linear Temporal Logic (LTL), have shown increasing popularity. Fuzzy conformance checking [26] calculates an evaluation score that indicates how much the observed process instance complies instead of providing a simple yes/no answer.

In order to enable fine-grained compliance diagnostics at run-time, Compliance Rule Graphs [10] and colored automata [16] have been suggested. However, both mainly focus on control flow perspective. In turn, [5] compares approaches for monitoring business process compliance based on 10 compliance monitoring functionalities (CMF). In particular, it is emphasized that existing approaches do not provide a satisfactorily solution that combines an expressive language (CMF 1-5) with full compliance traceability (CMF 8+9). Table 1 summarizes the results from [5].

<table>
<thead>
<tr>
<th>Approach</th>
<th>CMF 1 time</th>
<th>CMF 2 data</th>
<th>CMF 3 resources</th>
<th>CMF 4 non-atomic</th>
<th>CMF 5 life-cycle</th>
<th>CMF 6 multi-instance</th>
<th>CMF 7 reactive mgmt</th>
<th>CMF 8 proactive mgmt</th>
<th>CMF 9 root cause</th>
<th>CMF 10 compl. degree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mubicon LTL [16]</td>
<td>+/-</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+/-</td>
</tr>
<tr>
<td>Mubicon EC [25, 15]</td>
<td>+</td>
<td>+/-</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>+/-</td>
<td>+/-</td>
</tr>
<tr>
<td>ECE Rules [26]</td>
<td>+</td>
<td>+/-</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>+/-</td>
<td>+/-</td>
</tr>
<tr>
<td>SCT [24]</td>
<td>+/-</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SeaFlows [10]</td>
<td>+/-</td>
<td>+/-</td>
<td>+/-</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>+/-</td>
<td>+/-</td>
</tr>
</tbody>
</table>

As opposed to the approaches described above, [27] monitors performance measures in the context of artifact-centric process models. In turn, [28, 14, 29] provide techniques to a posteriori verify the compliance of execution logs with a set of constraints. Some of these approaches not only focus on the control flow perspective, but take the time perspective [14] or resource perspective [28] into account as well.

A priori or design time compliance checking has been addressed by a multitude of approaches for a long time. Most of them apply model checking techniques (e.g., [30–33]). In addition, some of these design time approaches use visual compliance rules and not only consider the control flow perspective, but the data, interaction or time perspectives as well. Other compliance checking approaches, in turn, are based on Petri-Nets [34] and Mixed-Integer Programming [35].

Finally, there are few frameworks, which address the integration of business process compliance throughout the entire process lifecycle [9, 36, 8, 37, 38].
3 Motivating Example

This section introduces a motivating example, which refers to the event log from Fig. 2 and deals with an order-to-delivery process. The latter is subject to three compliance rules, which stem from internal guidelines.

Note that compliance rule $c_1$ is satisfied in one case, but violated in another. In particular, the depicted log refers to two different request items related to customers Mr. Smith and Mrs. John. These items, in turn, trigger two different instances of compliance rule $c_1$. In both cases, the amount is greater than 10,000€ and hence a solvency check is required. However, the latter was only performed for the request item of Mr. Smith, but not for the one of Mrs. John (i.e., $c_1$ is violated in the latter case). Besides the violation of $c_1$, compliance rule $c_2$ is violated twice as well. While the violated instance of rule $c_1$ will never be successfully completed, the violations of $c_2$ still may be healed by informing the agent, who sent the requests, about the results of the approvals.

The compliance rule examples further indicate that solely monitoring control flow dependencies between tasks is not sufficient in order to ensure compliance at run-time. In addition, constraints in respect to the data, time and resource perspectives of a business process as well as the interactions this process has with partner processes must be monitored [39, 21, 17, 18]. For example, the data perspective of compliance rule $c_1$ is addressed by the request item and its amount. In turn, receiving the request item (cf. $c_1$) represents an interaction with a business partner. The phrase ”by different staff members” deals with the resource perspective, whereas the condition ”at max three days” refers to the time perspective. To meet practical demand, compliance monitoring must not abstract from these process perspectives.

<table>
<thead>
<tr>
<th>#</th>
<th>date</th>
<th>time</th>
<th>type</th>
<th>id</th>
<th>details</th>
</tr>
</thead>
<tbody>
<tr>
<td>37</td>
<td>1/7/2013</td>
<td>15:27</td>
<td>receive</td>
<td>124</td>
<td>Request</td>
</tr>
<tr>
<td>38</td>
<td>1/7/2013</td>
<td>15:27</td>
<td>write</td>
<td>124</td>
<td>customer = Mr. Smith</td>
</tr>
<tr>
<td>39</td>
<td>1/7/2013</td>
<td>15:27</td>
<td>write</td>
<td>124</td>
<td>amount = 15.000€</td>
</tr>
<tr>
<td>40</td>
<td>1/7/2013</td>
<td>15:27</td>
<td>end</td>
<td>124</td>
<td>Request</td>
</tr>
<tr>
<td>55</td>
<td>1/7/2013</td>
<td>16:03</td>
<td>receive</td>
<td>592</td>
<td>Request</td>
</tr>
<tr>
<td>56</td>
<td>1/7/2013</td>
<td>16:03</td>
<td>write</td>
<td>592</td>
<td>customer = Mrs. John</td>
</tr>
<tr>
<td>57</td>
<td>1/7/2013</td>
<td>16:03</td>
<td>write</td>
<td>592</td>
<td>amount = 27.000€</td>
</tr>
<tr>
<td>67</td>
<td>1/7/2013</td>
<td>18:03</td>
<td>end</td>
<td>592</td>
<td>Request</td>
</tr>
<tr>
<td>77</td>
<td>2/7/2013</td>
<td>15:43</td>
<td>start</td>
<td>234</td>
<td>SolvencyCheck (Mrs. Brown)</td>
</tr>
<tr>
<td>78</td>
<td>2/7/2013</td>
<td>15:43</td>
<td>read</td>
<td>234</td>
<td>customer = Mr. Smith</td>
</tr>
<tr>
<td>79</td>
<td>2/7/2013</td>
<td>15:54</td>
<td>write</td>
<td>234</td>
<td>rating = high</td>
</tr>
<tr>
<td>80</td>
<td>2/7/2013</td>
<td>15:55</td>
<td>end</td>
<td>234</td>
<td>SolvencyCheck</td>
</tr>
<tr>
<td>91</td>
<td>2/7/2013</td>
<td>18:13</td>
<td>start</td>
<td>453</td>
<td>Approval (Mr. Muller)</td>
</tr>
<tr>
<td>92</td>
<td>2/7/2013</td>
<td>18:14</td>
<td>read</td>
<td>453</td>
<td>customer = Mr. Smith</td>
</tr>
<tr>
<td>93</td>
<td>2/7/2013</td>
<td>18:14</td>
<td>read</td>
<td>453</td>
<td>rating = high</td>
</tr>
<tr>
<td>94</td>
<td>2/7/2013</td>
<td>18:17</td>
<td>write</td>
<td>453</td>
<td>result = granted</td>
</tr>
<tr>
<td>95</td>
<td>2/7/2013</td>
<td>18:18</td>
<td>end</td>
<td>453</td>
<td>Approval</td>
</tr>
<tr>
<td>96</td>
<td>2/7/2013</td>
<td>18:19</td>
<td>start</td>
<td>642</td>
<td>Approval (Mrs. Brown)</td>
</tr>
<tr>
<td>97</td>
<td>2/7/2013</td>
<td>18:20</td>
<td>read</td>
<td>642</td>
<td>customer = Mrs. John</td>
</tr>
<tr>
<td>98</td>
<td>2/7/2013</td>
<td>18:23</td>
<td>write</td>
<td>642</td>
<td>result = granted</td>
</tr>
<tr>
<td>99</td>
<td>2/7/2013</td>
<td>18:23</td>
<td>end</td>
<td>642</td>
<td>Approval</td>
</tr>
</tbody>
</table>

Compliance rules

- $c_1$: When a request item with an amount greater than 10,000€ is received from an agent, the request must not be approved unless the solvency of the respective customer was checked. The latter task must be started at max three days after the receipt. Further, task approval and task solvency check must be performed by different staff members.

- $c_2$: After approval of a request item, the agent must be informed about the result within one days.

- $c_3$: After starting the production related to a particular order the latter may only be changed by the head of production.

Fig. 2: Event log of order-to-delivery processes and compliance rules
4 Fundamentals of Extended Compliance Rule Graphs

This paper utilizes the extended Compliance Rule Graph (eCRG) language for compliance monitoring. Since this language is based on the Compliance Rule Graph (CRG) language, we first introduce CRGs before presenting eCRG fundamentals.

4.1 Compliance Rule Graphs

The Compliance Rule Graph (CRG) language was introduced in [40, 10, 41]. It allows for the visual modeling of compliance rules that focus on the control flow perspective (i.e., sequence flow) of business processes. A CRG corresponds to an acyclic graph that consists of an antecedence pattern as well as one or multiple related consequence patterns. Both kinds of patterns are modeled using occurrence and absence nodes, which either express the occurrence or absence of events (e.g. events related to the execution of a particular task). Furthermore, the edges connecting these nodes express control flow dependencies.

As illustrated in Fig. 3, an event trace (i.e., a finite sequence of events related to the same process instance) is considered as compliant with a CRG iff for each match of the antecedence pattern there is at least one corresponding match for one of the consequence patterns. In turn, a trace is considered as trivially compliant iff there is no match of the antecedence pattern at all. As example consider the CRG from Fig. 3. It expresses that for each B not preceded by an A, a D must occur. Further, there must be no C that precedes B and D.

![Fig. 3: CRG example and semantics [8]](image)

4.2 Extended Compliance Rule Graph

The CRG language focuses on the control flow perspective of compliance rules, but ignores other perspectives. In [17, 18], therefore, we introduced the extended Compliance Rule Graph (eCRG) as a visual language for modeling compliance rules not only covering the control flow perspective, but providing integrated support for the resource, data and time perspectives as well as for interactions with business partners. To cover these various perspectives, the eCRG language allows for attachments in addition to nodes and connectors (i.e. edges). Thereby, nodes refer to entities (e.g. a data object) or events, whereas edges and attachments are used to refine the nodes or edges they are affiliated to. Furthermore, an eCRG may contain instance nodes referring to particular objects, which exist independently from the respective rule (e.g. Mr. Smith, postnatal ward, physician). Hence, instance nodes are neither part of...
the antecedence nor the consequence pattern, but constitute the instance pattern. Fig. 4 gives an overview of the elements of the eCRG language.

The elements of the eCRG language are partitioned into the control flow, data, time, resource and interaction perspectives that will be described in the following in more detail (cf. Fig. 4).

Control flow perspective. Modeling the control flow perspective of compliance rules is supported through four kinds of task nodes, i.e., antecedence occurrence, antecedence absence, consequence occurrence, and consequence absence task nodes. Based on these nodes it can be expressed whether or not particular tasks shall be executed. In addition, two kinds of sequence flow connectors are provided that allow constraining the execution sequence of tasks. Note that the absence of a sequence flow indicates parallel flow. Furthermore, exclusive connectors express mutual exclusion of the tasks they refer to. Finally, alternative connectors express that at least one of the connected tasks must occur.

Interaction perspective. The interaction perspective supports the exchange of messages with business partners. According to task nodes, four kinds of sending and four kinds of receiving message nodes are provided, i.e., antecedence occurrence, antecedence absence, consequence occurrence, and consequence absence nodes.

Time perspective. The eCRG language offers the following elements for modeling the time perspective: point-in-time nodes, time condition attachments, and time distance connectors. Point-in-time nodes express a particular date or point-in-time (e.g. 26th October 2014). Time conditions, in turn, may be attached to task nodes and sequence flow connectors to constrain the duration of a task or the time distance between task nodes, message nodes and point-in-time nodes. Finally, time distance connectors allow constraining the time distance without implying a particular sequence.
Data perspective. Data container nodes and data object nodes support the modeling of the data perspective in eCRGs. Furthermore, data flow connectors, data relation connectors and data condition attachments are provided. Data container nodes refer to process data elements or global data stores. By contrast, data object nodes refer to particular data values and data object instances. Both kinds of data nodes may be part of the antecedence or consequence pattern, or represent a particular data container and data object, respectively. Data flow connectors define which process tasks read or write which data objects or data containers. To constrain data containers, data objects and data flow, data conditions may be attached. In turn, data relation connectors allow comparing data objects.

Resource perspective. For modeling the resource perspective of compliance rules, resource nodes are provided, i.e., staff member, role, group, and organizational unit nodes. Similar to task nodes, resource nodes may be part of the antecedence or consequence pattern. Alternatively, they may represent a particular resource instance (e.g. Mr. Smith, postnatal ward, physician). To specify dependencies among resources, resource relation connectors are provided. In turn, resource condition attachments constrain a particular resource node. Finally, the performing relation indicates the performer of a task node.

Fig. 5 applies the eCRG language in order to model the compliance rules from our motivating example in Section 3, which have been presented in verbalized form in Fig. 2. In particular, Fig. 2(c1) addresses all process perspectives, i.e., the control flow, data, time and resource perspectives as well as interactions with business partners. In turn, Fig. 2(c2) does not refer to the resource perspective, whereas time and interaction perspectives are not addressed in Fig. 2(c3).
A formal specification of an eCRG is provided in Def. 1 (for a more detailed definition of eCRGs including a formal definition of $\phi$ see [42]).

**Definition 1 (Extended Compliance Rule Graph (eCRG)).**

Let $T$ be the set of points in time, $R$ be the set of human resources, and $\Omega$ be the set of all data objects. Then: An extended Compliance Rule Graph (eCRG) is a tuple $\Psi = (N, E, \phi, \text{type}, \text{src}, \text{tgt}, \text{at}, \text{pt})$ with

- $N := T \cup MN \cup O \cup C \cup R \cup P$ being the set of nodes that may be partitioned into the sets of task nodes $T$, message nodes $MN$, data object nodes $O$, data container nodes $C$, resource nodes $R$, and point-in-time nodes $P \subset T$.
- $E := s^f \cup d^f \cup p^m \cup r^r$ being the set of edges that may be partitioned into the sets of sequence flow edges $s^f$, data flow edges $d^f$, performing relations $p^m$, and resource relations $r^r$.
- $\phi := d^c \cup t^c \cup r^r$ being the set of attachments that may be partitioned into the sets of data conditions $d^c$, time conditions $t^c$, and resource conditions $r^r$.
- $\text{type} : T \cup MN \cup d^f \rightarrow T$ mapping each task node (message node, data flow edge) to a task type (message type, parameter name),
- $\text{src} : E \rightarrow N$ (tgt : $E \rightarrow N$) mapping each edge to its source (target) node,
- $\text{at} : \phi \rightarrow N \cup E$ mapping each attachment to the underlying node or edge it is attached to, and
- $\text{pt} : N \cup E \cup \phi \rightarrow \{AO, AA, CO, CA, I\}$ mapping each element of an eCRG to the corresponding pattern.

Further:

- $\Lambda_{\Psi} := N \cup E \cup \phi$ is the set of all elements,
- $\Gamma_{\Psi} := T \cup MN$ is the set of task and message nodes,
- $\phi_{\Psi} : \Lambda_{\Psi} \rightarrow \Lambda_{\Psi}$ maps each element to its affiliation; i.e., the element to which it is affiliated$^3$.
- For each set $X \subset \Lambda_{\Psi}$ of elements of an eCRG and each pattern $y \in \{AO, AA, CO, CA, I\}$, we define $X^y := \{x \in X | pt(x) = y\}$ as the pattern $y$ of $X$.

Function $pt$ partitions the elements of an eCRG in five patterns, which are the instance pattern ($I$) as well as the antecedence occurrence ($AO$), antecedence absence ($AA$), consequence occurrence ($CO$), and consequence absence ($CA$) patterns. Dependencies among these patterns are shown in Fig. 6, whereas the lower ones depend on the upper ones they

![Diagram](image-url)
are connected to. Based on the latter, Def. 2 formally specifies how connected elements constrain and depend on each other. In particular, a node, edge or attachment $\lambda_1$ depends on another element $\lambda_2$, if they are connected and the pattern of $\lambda_1$ depends on the one of $\lambda_2$. For example, task node production depends on its outgoing sequence flow edge in Fig. 5(c3). In turn, in Fig. 5(c3), message node request does not depend on both outgoing sequence flows. Instead, the latter depend on the message node.

**Definition 2 (Pattern Dependency Order).**

Let $\Psi = (N, E, \circ, \text{type}, \text{src}, \text{tgt}, \text{at}, \text{pt})$ be an eCRG and let $\lambda_1, \lambda_2 \in \Lambda_\Psi$ be two elements of the eCRG. Then:

- $\triangleright$ defines the partial dependency order over the set of patterns as specified in Fig. 6 (e.g. $I \triangleright AA$),
- We say $\lambda_1$ has a higher dependency level than $\lambda$ (i.e., $\lambda_1 \triangleright \lambda_2$), iff the pattern of $\lambda_1$ is greater than the one of $\lambda_2$ according to the dependency order $\triangleright$, i.e.,
  \[ \lambda_1 \triangleright \lambda_2 \Leftrightarrow \text{pat}(\lambda_1) \triangleright \text{pat}(\lambda_2) \]
- and $\preceq$, $\succeq$, $\triangleleft$ and $\trianglerighteq$ are defined accordingly.
5 eCRG Operational Semantics

This section introduces the operational semantics of the eCRG language that enables visually monitoring business process compliance at run-time. As discussed in Section 1, the latter is based on event streams that occur during the execution of business processes. In particular, compliance monitoring shall detect or, if possible, prevent compliance violations. For this purpose, first of all, Section 5.1 introduces the different events that are supported. Second, the fundamental states of a compliance rule are introduced in Section 5.2. Third, Section 5.3 specifies markings that annotate the elements of an eCRG with text, colors and symbols. Finally, Section 5.4 specifies the processing of events and discusses how the latter evolve and unfold markings of an eCRG (cf. Section 5.4).

5.1 Events

As this report addresses compliance monitoring in respect to multiple process perspectives, we not only monitor events that refer to the start and end of tasks. In addition, we consider events that correspond to the sending and receiving of messages as well as data flow events that log how activities read from and write to data sources. Furthermore, events may include temporal information as well as information about involved resources.

Table 2 summarizes the event types supported by our approach. Note that each event includes the time it occurs as well as a unique id. The latter enables us to identify correlations between the start, end and data flow events related to the same task or message.

<table>
<thead>
<tr>
<th>Event Type</th>
<th>Event Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task events</td>
<td>start(time, id, tasktype, performer)</td>
</tr>
<tr>
<td></td>
<td>end(time, id, tasktype, performer)</td>
</tr>
<tr>
<td>Message events</td>
<td>send(time, id, message)</td>
</tr>
<tr>
<td></td>
<td>receive(time, id, message)</td>
</tr>
<tr>
<td></td>
<td>end(time, id, message)</td>
</tr>
<tr>
<td>Data flow events</td>
<td>write(time, id, value → source)</td>
</tr>
<tr>
<td></td>
<td>read(time, id, value ← source)</td>
</tr>
</tbody>
</table>

Based on the events from Table 2, Def. 3 formally specifies event logs and streams.

**Definition 3 (Event Log or Event Stream).** Let $\mathcal{E}$ be the set of events (cf. Table 2) and let $\mathcal{T}$ be the discrete set of points in time. Then:

- $\mathcal{L} : \mathbb{N} \rightarrow \mathcal{E} \cup \{\varnothing\}$ is an event stream or event log.
- $\text{time} : \mathcal{E} \rightarrow \mathcal{T} : \text{time}(\text{event}(\vartheta, \ldots)) := \vartheta$ maps each event to the corresponding point-in-time.

Note that we assume that event streams are correct; i.e., they do not deviate (cf. [43]) from the real process, ids are unique, and events are provided in an ascending order (i.e., $\forall i, j \in [1..\text{pos}_\mathcal{L}] : i < j \Rightarrow \text{time}(\mathcal{L}(i)) \leq \text{time}(\mathcal{L}(j))$).

5.2 States of Compliance Rules

When monitoring the compliance of running process instances, compliance rules take varying states [10, 8]. Fig. 7 outlines the states that are supported by our approach. The most
fundamental state is \texttt{NOTActivated}, i.e., the compliance rule does not concern the running process instances until now. The opposite state \texttt{Activated} means that the compliance rule affects the process instance and includes the sub-states \texttt{TempSatisfied} and \texttt{TempViolated}. \texttt{TempSatisfied} is further partitioned into \texttt{Violable} and \texttt{Satisfied}, whereas \texttt{TempViolated} includes the sub-states \texttt{Pending} and \texttt{Violated}. As explained by our motivating example in Section 3, business process can multiply trigger (i.e. activate) compliance rule. Hence, a compliance rule can be in state \texttt{Activated} multiple times as indicated by superscript "+". Note that each of these activations of a compliance rule can take a different sub-state. For instance, the event log of our motivating example activates compliance rule $c_1$ twice (cf. Fig. 2). The first activation is \texttt{Satisfied}, whereas the second activation remains in state \texttt{Violated}.

![Fig. 7: States of compliance rules](image)

5.3 eCRG Markings

To monitor the state of a compliance rule, we annotate and mark the elements of an eCRG (cf. Section 4, [17, 18]) with symbols, colors and text (cf. Figs. 8). Such a marking of an eCRG (i.e., annotated eCRG) highlights whether or not the events corresponding to a node have occurred so far. Further, it describes whether conditions corresponding to edges and attachments are satisfied, violated, or still will be evaluated (cf. Figs. 8). Since there may be different activations and instantiations of a particular compliance rule, Def. 4 defines the state of an eCRG in terms of a set of markings, which, in turn, can be utilized to calculate the corresponding states of compliance as introduced in Section 5.2.

![Fig. 8: Annotations of eCRG elements](image)
Def. 4 (Markings and State of an eCRG).

Let $I$ be a set of unique identifiers. Let further $\Sigma$ be the set of points in time, $R$ be the set of human resources, and $\Omega$ be the set of all data objects, whereas $\epsilon$ is the empty value, i.e., the placeholder for not yet set identifiers, points in time, human resources, and data objects. Furthermore, let $\Psi = (N, E, o, type, src, tgt, at, pt)$ be an eCRG. Then: A marking $M$ of $\Psi$ is a tuple $M = (m, res, val, id, t_s, t_e)$ where

- $m : A_{\Psi} \rightarrow \{\Box, \triangle, \triangleright, \checkmark, \times\}$ marks each element of the eCRG with either NOT_ACTIVATED/NOT_MARKED ($\Box$), Activated ($\triangle$), Running ($\triangleright$), Completed/ Satisfied ($\checkmark$), or Skipped ($\times$).
- $res : O \cup R \cup pm \rightarrow \{\epsilon\} \cup R$ assigns a resource to each task node, resource node and performing relation edge,
- $val : O \cup df \rightarrow \{\epsilon\} \cup \Omega$ assigns a value to each data object node and data flow edge,
- $id : T \cup MN \cup C \cup df \rightarrow \{\epsilon\} \cup \Sigma$ assigns an unique identifier to each task node, message node, data container node, or data flow edge,
- $t_s : T \cup MN \cup P \rightarrow \{\epsilon\} \cup \Sigma$ (and $t_e : T \cup MN \cup P \rightarrow \{\epsilon\} \cup \Sigma$ respectively) assigns a starting (ending) time to each task node, message node, and point-in-time node.

Further, we define

- $M_{\Psi}^{in} := (m, res, val, id, t_s, t_e)$ as the initial marking of eCRG $\Psi$, where
  - $m(x) = \Box$ marks every element $x$ with $\Box$,
  - $t_s(x) := t_e(x) := \epsilon$ sets starting and ending times to $\epsilon$, and
  - $res(x) := \epsilon$, if $x \notin I$ val($x$) := $\begin{cases} \epsilon, & \text{if } x \notin I \\ x, & \text{else} \end{cases}$ id($x$) := $\begin{cases} \epsilon, & \text{if } x \notin I \\ x, & \text{else} \end{cases}$
    - only assign resources, values and identifiers to elements of the instance pattern.
- $M_{\Psi}$ as the set of all markings of $\Psi$,
- $M_{\Psi}^{\xi} \subseteq M_{\Psi}$ as the state of eCRG $\Psi$ after processing event stream $\xi$ until entry $i \in \mathbb{N}$, and
- $M_{\Psi}^{in} := \{M_{\Psi}^{in}\}$ as the initial state of $\Psi$.

5.4 Operational Semantics

In order to enable the processing of events in the context of the extended Compliance Rule Graph (eCRG) language, this section introduces the eCRG operational semantics. In particular Defs. 5-10 specify how the different event types are processed (cf. Fig. 9). First, all markings must be updated to the point-in-time of the event occurred (cf. Def. 5). Second, effects of the update (i.e., the updated annotations) are propagated to adjacent elements (cf. Def. 10). Third, Defs. 6-9 specify the actual handling of an event based on its type. Finally, annotations are propagated once again before completing the processing of an event. Note that the first two steps may be skipped if time does not differ from the one of the event directly processed before. In turn, these two steps may be applied without the subsequent ones in order to calculate the current state of a compliance rule at any point-in-time between two events.

5.4.1 Update Marking

Def. 5 specifies the update of a marking to the current point-in-time. In particular, the annotation of point-in-time nodes changes from NOT_MARKED $\Box$ to Completed $\checkmark$, if the
Fig. 9: Processing of start, message, data and end events

points in time to which the nodes refer have passed (t1). Furthermore, time conditions on running task nodes or sequence flow edges will be SKIPPED (×) if they are no longer satis-
fiable (t2).

Definition 5 (Update Marking).
Let $M := (m, res, val, id, ts, te) \in M_{\Psi}$ be a marking of eCRG $\Psi$ and let $\vartheta \in T$ be a point-in-time. Then: $\text{upd}^\vartheta : M_{\Psi} \rightarrow M_{\Psi}$ with $\text{upd}^\vartheta(M) := (m', res, val, id, ts, te)$ calculates the update of marking $M$ to point-in-time $\vartheta$, where

$$m'(\lambda) := \begin{cases} \checkmark, & \text{if } \lambda \in P \land m(\lambda) = \varnothing \land \lambda \leq \vartheta, \\ \times, & \text{if } \lambda \in \varsigma t c \land m(\lambda) = \varnothing \land (t := \text{at}(\mu t c) \in T) \\ \land (\forall \varsigma \in T : (\varsigma \geq \vartheta) \Rightarrow \neg \text{at}(t_{\varsigma}(t), \varsigma)), \\ \times, & \text{if } \lambda \in \varsigma t c \land m(\lambda) = \varnothing \land (\nu(1, n_{2}) := \text{at}(\mu t c) \in \Psi f) \\ \land (\forall \varsigma \in T : (\varsigma \geq \vartheta) \Rightarrow \neg \text{at}(t_{\varsigma}(n_{1}), \varsigma)), \\ m(\lambda), & \text{else} \end{cases}$$

(t1)

(t2)

Fig. 10 updates marking $B$ of compliance rule $c_{1}$ from our motivating example (cf. Fig. 2) to the point in time 4/7/2013 16:05. According to Def. 5(t2), the time condition on the sequence flow edge from message node is marked as skipped, because it can not be satisfied any more.

Fig. 10: Update of markings
5.4.2 Start and Message Event Handling

Def. 6 (Def. 7) non-deterministically handles start events (message events), in order to deal with different instantiations of a compliance rule. In particular, the set of matching and activated task nodes (message nodes) is determined first. Then, for each subset a new marking is instantiated, which solely changes the marking of the selected nodes from ACTIVATED (Δ) to RUNNING (■), and, accordingly, sets the identifiers, resources and starting times (cf. Fig. 11). Node that the empty set is among the subsets and the result hence includes the unchanged original marking as well.

Definition 6 (Start Event Handling).
Let \( M = (m, res, val, id, t_s, t_e) \in \mathcal{M}_\varPsi \) be a marking of eCRG \( \varPsi \) and \( \sigma = \text{start}(\vartheta, \iota, \tau, \rho) \) be a start event. Then:

\[ \Delta^\sigma(M) := \{ \lambda \in T | m(\lambda) = \Delta \wedge \text{type}(\lambda) = \tau \} \]

is the set of matching and activated task nodes,

\[ \text{For each subset } \delta \subseteq \Delta^\sigma(M), hdl^\sigma_\vardelta : \mathcal{M}_\varPsi \rightarrow \mathcal{M}_\varPsi \text{ with } hdl^\sigma_\vardelta(M) := (m^\delta, res^\delta, val^\delta, id^\delta, t_s^\delta, t_e) \]

calculates the handling of start event \( \sigma \) by node set \( \delta \) and marking \( M \):

\[ m^\delta(\lambda) := \begin{cases} \vartheta, & \text{iff } \lambda \in \delta \\ m(\lambda), & \text{else} \end{cases} \]

\[ id^\delta(\lambda) := \begin{cases} \iota, & \text{iff } \lambda \in \delta \\ id(\lambda), & \text{else} \end{cases} \]

\[ res^\delta(\lambda) := \begin{cases} \rho, & \text{iff } \lambda \in \delta \\ res(\lambda), & \text{else} \end{cases} \]

\[ t_s^\delta(\lambda) := \begin{cases} \vartheta, & \text{iff } \lambda \in \delta \\ t_s(\lambda), & \text{else} \end{cases} \]

\[ t_e^\delta(\lambda) := \begin{cases} \iota, & \text{iff } \lambda \in \delta \\ id(\lambda), & \text{else} \end{cases} \]

\[ hdl^\sigma : \mathcal{M}_\varPsi \rightarrow 2^{\mathcal{M}_\varPsi} \text{ with } hdl^\sigma(M) = \{hdl^\sigma_\vardelta(M) | \delta \in \Delta^\sigma(M)\} \]

calculates all handlings of \( \sigma \) by \( M \).

Definition 7 (Message Event Handling).
Let \( M = (m, res, val, id, t_s, t_e) \in \mathcal{M}_\varPsi \) be a marking of eCRG \( \varPsi \) and let \( \sigma = \text{send}(\vartheta, \iota, \tau) \) (and \( \text{receive}(\vartheta, \iota, \tau) \) respectively) be a message event. Then:

\[ \Delta^\sigma(M) := \{ \lambda \in MN | m(\lambda) = \Delta \wedge \text{type}(\lambda) = \tau \} \]

is the set of matching and activated message nodes,

\[ \text{For each subset } \delta \subseteq \Delta^\sigma(M), hdl^\sigma_\vardelta : \mathcal{M}_\varPsi \rightarrow \mathcal{M}_\varPsi \text{ with } hdl^\sigma_\vardelta(M) := (m^\delta, res^\delta, val^\delta, id^\delta, t_s^\delta, t_e) \]

calculates the handling of message event \( \sigma \) by node set \( \delta \) and marking \( M \):

\[ m^\delta(\lambda) := \begin{cases} \vartheta, & \text{iff } \lambda \in \delta \\ m(\lambda), & \text{else} \end{cases} \]

\[ id^\delta(\lambda) := \begin{cases} \iota, & \text{iff } \lambda \in \delta \\ id(\lambda), & \text{else} \end{cases} \]

\[ res^\delta(\lambda) := \begin{cases} \vartheta, & \text{iff } \lambda \in \delta \\ res(\lambda), & \text{else} \end{cases} \]

\[ t_s^\delta(\lambda) := \begin{cases} \iota, & \text{iff } \lambda \in \delta \\ id(\lambda), & \text{else} \end{cases} \]

\[ t_e^\delta(\lambda) := \begin{cases} \vartheta, & \text{iff } \lambda \in \delta \\ t_e(\lambda), & \text{else} \end{cases} \]

\[ hdl^\sigma : \mathcal{M}_\varPsi \rightarrow 2^{\mathcal{M}_\varPsi} \text{ with } hdl^\sigma(M) = \{hdl^\sigma_\vardelta(M) | \delta \in \Delta^\sigma(M)\} \]

calculates all handlings of \( \sigma \) by \( M \).

Fig. 11 illustrates the handling of a start and message events. In particular, the receipt of the message \textit{request} starts the corresponding message node of marking \( \vartheta \) in the upper example; i.e., the node is marked as RUNNING (■) and labeled with the receive time of the message. The lower example handles the start-event of task solvency check. The corresponding task node is marked as RUNNING. Furthermore, its the start time and performer of the task are specified.
5.4.3 End Event Handling

Since we use unique identifiers for start, message and end events, the latter can be processed deterministically (cf. Def. 8). In particular, the annotations of all nodes assigned to the identifier of the end event and marked with RUNNING (✓), are changed to COMPLETED (√) and their ending time is set accordingly (cf. Fig. 12).

**Definition 8 (End Event Handling).**

Let \( M = (m, res, val, id, t_s, t_e) \in \mathcal{M}_{\varphi} \) be a marking of eCRG \( \Psi \) and \( \sigma = \text{end}(\vartheta, \iota, \tau, \rho) \) (and \( \text{end}(\vartheta, \iota, \tau) \)) respectively be an end event. Then: \( \text{hdl}^\sigma : \mathcal{M}_{\varphi} \rightarrow 2^{\mathcal{M}_{\varphi}} \) with \( \text{hdl}^\sigma(M) := \{(m', res, val, id, t_s, t'_e)\} \) calculates the handling of end event \( \sigma \) by marking \( M \):

\[
m' := \begin{cases} 
  \sqrt{\cdot}, & \text{iff } id(\lambda) = \iota \\
  m(\lambda), & \text{else}
\end{cases}

\[
t'_e(\lambda) := \begin{cases} 
  \vartheta, & \text{iff } id(\lambda) = \iota \\
  id(\lambda), & \text{else}
\end{cases}
\]

Fig. 12 shows the handling of an end event. In particular, the processing of the message request is finished and the the corresponding message node of marking a5 is marked as COMPLETED (√).
5.4.4 Data Event Handling

We can also process data events deterministically, since the combination of the used unique identifiers and the used parameter names clearly refers to the data flow edges concerned (cf. Def. 9). In particular, the latter is marked as Satisfied (√) and annotated with the data value and the identifier of the data source or data target; i.e., the identifier of a data container (cf. Fig. 13).

Definition 9 (Data Event Handling).
Let \( M := (m, res, val, id, t_s, t_e) \in \mathcal{M}_\Psi \) be a marking of eCRG \( \Psi \). Further, let \( \sigma = \text{write}(\vartheta, \iota, \upsilon \par \omega) \) (and \( \text{read}(\vartheta, \iota, \upsilon \par \omega) \)) respectively be a data event. Then: \( \text{hdl}^\sigma : \mathcal{M}_\Psi \to 2^{\mathcal{M}_\Psi} \), with \( \text{hdl}^\sigma(M) := \{(m', res, val', id', t_s, t_e)\} \) calculates the handling of data event \( \sigma \) by marking \( M \):

\[
\begin{align*}
m'(\lambda) &:= \begin{cases} 
\vartheta, & \text{if } \lambda = (n, t) \in d_f^\top \wedge m(\lambda) = \varnothing \wedge id(n) = \iota \wedge m(n) = \top \wedge \text{type}(\lambda) = \text{par} \\
m(\lambda), & \text{else}
\end{cases} \\
val'(\lambda) &:= \begin{cases} 
\upsilon, & \text{if } \lambda = (n, t) \in d_f^\top \wedge m(\lambda) = \varnothing \wedge id(n) = \iota \wedge m(n) = \top \wedge \text{type}(\lambda) = \text{par} \\
val(\lambda), & \text{else}
\end{cases} \\
id'(\lambda) &:= \begin{cases} 
\omega, & \text{if } \lambda = (n, t) \in d_f^\top \wedge m(\lambda) = \varnothing \wedge id(n) = \iota \wedge m(n) = \top \wedge \text{type}(\lambda) = \text{par} \\
id(\lambda), & \text{else}
\end{cases}
\end{align*}
\]

Fig. 13 illustrates the handling of a data events. The writing data flow event amount of the message request is handled in the upper example. In particular, the corresponding outgoing data flow edge of message node request is marked as Satisfied (√) and annotated with the amount of 15.000. The lower example handles the reading data flow event customer that marks the incoming data flow edge of task solvency check as Satisfied (√) and annotates it with the customer Mr. Smith. Note that the latter annotation Mr. Smith does not meet the value Mrs. John of the data object customer; i.e., the handling of data flow events allows for conflicts. Furthermore, note that we omit the identifiers of data sources (i.e., data containers) in our examples for the sake of simplicity.
5.4.5 Propagation of effects

After each update or event handling, we propagate the corresponding effects (i.e., changes to annotations) on adjacent nodes and edges in order to ensure correct annotations (e.g., activation of subsequent task nodes). Further, we detect contradictory annotations related to the data and resource perspectives (cf. Def. 10).

In particular, resources are propagated from task nodes to Not_MARKED (□), dependent resource nodes (x2) by following resource edges (r1). In turn, data values and the identifiers of data containers are propagated from data flow edges to Not_MARKED (□), dependent data object nodes (d1) as well as data container nodes (d2). Next, the edge and its target node are marked with Satisfied (√) (x3).

Note that the aforementioned propagation steps are not performed, if resources, data values or identifiers of target resource nodes, data objects or data containers were already set to another value before. To highlight such conflicts, the corresponding data flow and resource edges will be marked with Skipped (×) (x4). Not_MARKED (□) data flow edges will be also marked with Skipped (×) if the corresponding task or message node is completed (d5). Afterwards, all conditions (x6) and relations are reevaluated (x7). If any element of the eCRG, affiliated to a dependent task or message node, is now Skipped (×), the corresponding task or message node will be marked with Skipped (×) as well (x8).

Next, the outgoing sequence flows of completed nodes are marked with Satisfied (√) (cf1). In turn, Not_MARKED (□), incoming sequence flow edges of already started nodes are marked with Skipped (×). Further, sequence flow edges from and to Skipped (×) nodes

Fig. 13: Handling of data events
are marked with SKIPPED (⊥) as well (cf2). Task and message nodes become ACTIVATED (△) when all incoming sequence flows, on which it depends, are marked as SATISFIED (✓) satisfied (cf3). In turn, task or message nodes will be marked with SKIPPED (⊥) when they depend on incoming sequence flows being SKIPPED (⊥) before (cf4). Note that the latter might trigger (cf2) again.

**Definition 10 (Effect Propagation).**

Let \( M := (m, \text{res}, \text{val}, \text{id}, t_s, t_e) \in \mathbb{M}_\Psi \) be a marking of eCRG \( \Psi \). Then: \( \text{prop} : \mathbb{M}_\Psi \rightarrow \mathbb{M}_\Psi \), \( \text{prop}(M) := (m^{(\ast)}, \text{res}^{(1)}, \text{val}^{(1)}, \text{id}^{(1)}, t_s, t_e) \) propagates effects on \( M \), whereby

\[
\begin{align*}
\text{res}^{(1)}(\lambda) := & \begin{cases} 
\rho, \text{ if } (\lambda = (t, r) \in pfm \land m(\lambda) = \square \land \text{res}(t) = \rho) \\
\vee (\lambda \in R \land \text{res}(\lambda) = \epsilon \land (\exists e := (t, \lambda) \in pfm \land \lambda \neq e \land \text{res}(e) = \rho)) 
\end{cases} \quad (r1) \\
\text{val}^{(1)}(\lambda) := & \begin{cases} 
\nu, \text{ if } \lambda \in O \land \text{val}(\lambda) = \epsilon \land (\exists e := (n, \lambda) \in df : \lambda \neq e \land \text{res}(e) = \nu) \\
\text{val}(\lambda), \text{ else}, 
\end{cases} \quad (d1) \\
\text{id}^{(1)}(\lambda) := & \begin{cases} 
\omega, \text{ if } \lambda \in C \land \text{id}(\lambda) = \epsilon \land (\exists e := (n, \lambda) \in df : \lambda \neq e \land \text{id}(e) = \omega) \\
\text{id}(\lambda), \text{ else}, 
\end{cases} \quad (d2) \\
\end{align*}
\]

\[
\begin{align*}
m^{(1)}(\lambda) := & \begin{cases} 
\checkmark, \text{ if } (\lambda \in R \land m(\lambda) = \square \land \text{res}^{(1)}(\lambda) \neq \epsilon) \\
\vee (\lambda \in O \land m(\lambda) = \square \land \text{val}^{(1)}(\lambda) \neq \epsilon) \\
\vee (\lambda \in C \land m(\lambda) = \square \land \text{id}^{(1)}(\lambda) \neq \epsilon) \\
\vee (\lambda = (t, r) \in pfm \land m(\lambda) = \square \land \text{res}^{(1)}(r) = \text{res}^{(1)}(\lambda)) \\
\vee (\lambda = (n, o) \in df \land o \in O \land m(\lambda) = \square \land \text{val}^{(1)}(o) = \text{val}^{(1)}(\lambda)) \\
\vee (\lambda = (n, c) \in df \land c \in C \land m(\lambda) = \square \land \text{id}^{(1)}(c) = \text{id}^{(1)}(\lambda)) \\
\end{cases} \quad (x3) \\
m^{(2)}(\lambda) := & \begin{cases} 
\checkmark, \text{ if } (\lambda \in o \land m^{(1)}(\text{at}(\lambda)) = \checkmark \land \lambda(\text{at}(\lambda))) \\
\vee (\lambda = (r_1, r_2) \in \Psi R \land m^{(1)}(r_1) = \checkmark \land m^{(1)}(r_2) = \checkmark \land \lambda(r_1, r_2)) \\
\vee (\lambda = (o_1, o_2) \in df \land \text{val}(\lambda) = \checkmark \land m^{(1)}(o_2) = \checkmark \land \lambda(o_1, o_2)) \\
\end{cases} \quad (x7) \\
m^{(3)}(\lambda) := & \begin{cases} 
\checkmark, \text{ if } (\lambda = (r_1, r_2) \in \Psi R \land m^{(1)}(r_1) = \checkmark \land m^{(1)}(r_2) = \checkmark \land \lambda(r_1, r_2)) \\
\vee (\lambda = (o_1, o_2) \in df \land \text{val}(\lambda) = \checkmark \land m^{(1)}(o_2) = \checkmark \land \lambda(o_1, o_2)) \\
\end{cases} \quad (x7) \\
m^{(4)}(\lambda) := & \begin{cases} 
\checkmark, \text{ if } (\lambda = (n_1, n_2) \in df \land m(\lambda) = \square \land m^{(3)}(n_1) = \checkmark \land \lambda(n_1, n_2)) \\
\text{id}(\lambda), \text{ else}, 
\end{cases} \quad (c f 1) \\
m^{(5)}(\lambda) := & \begin{cases} 
\Delta, \text{ if } \lambda \in T \cup MN \land m^{(4)}(\lambda) = \square \\
\land (\forall e := (n, \lambda) \in df : e \neq \lambda \Rightarrow m^{(4)}(e) = \checkmark) \\
\end{cases} \quad (c f 3)
\end{align*}
\]
Figs. 14-17 highlight the propagation after updates as well as after the handling of start, message, end, and data events. In particular, Fig. 14 transfers the annotation **Skipped** (×) of the time condition on the corresponding sequence flow edge that was marked with **Satisfied** (√) before. Based on this change, the subsequent task node **solvency check** is marked with **Skipped** (×) as well and its **Non_Marked** (□), incoming and outgoing sequence flow edges. Finally, the annotation **Skipped** (×) is propagated to the consequence absence task node **approval** as well as the consequence occurrence task node **approval**. Hence, the final marking Fig. 14(B′′) shows a temporal conflict.

In Fig. 15, the effects of a start event are propagated. In particular, the adjacent resource edge and resource node become **Completed** (√) and are annotated with resource **Mrs. Brown**. Further, the **Non_Marked** (□), incoming sequence flow edge of the just started task node **solvency check** is marked with **Skipped** (×). Finally, the consequence absence task node **approval** becomes also **Skipped** (×).

Fig. 16 propagates the effects of an end event. In particular, the outgoing sequence flow edges of the just completed message node **request** are marked with **Satisfied** (√). Furthermore, the subsequent task nodes **solvency check** and **approval** become **Activated** (∆).

The effects of data events are propagated in Fig. 17. In the upper example, the value 15,000 of the data flow edge used is written on the target data object **amount** that also becomes **Satisfied** (√). Further, the antecedence data condition the respective data flow is successfully evaluated and hence also marked as satisfied. As opposed to this, the comparison of the data object **customer** and its just written outgoing data flow edge fails. Hence, the latter data flow edge changes from **Satisfied** (√) to **Skipped** (×). Due to this change, the subsequent task node **solvency check** is marked with **Skipped** (×) as well and its outgoing
sequence flow edge. Finally, the annotation Skipped (✓) is also propagated on the following consequence occurrence task node approval. Hence, Fig. 14(D) highlights a data conflict.

Finally, Def. 11 formally specifies the update of the state of an eCRG to the current point in time as well as the processing of a particular event based on Defs. 5-10 and Fig. 9.

**Definition 11 (Event processing procedure).**

Let $\mathcal{M}_{\psi}^{i,\vartheta}$ be the state of an eCRG $\Psi$ after processing event stream $\mathcal{L}$ until the $i$th position ($i \in \mathbb{N}$), and let $\sigma := \mathcal{L}(i + 1)$ be the upcoming event on the following position of $\mathcal{L}$ and $\vartheta := \text{time}(\sigma) \in \Sigma$ be the current point in time. Then:

- $\mathcal{M}_{\psi}^{i,\vartheta} := \{M'|M' = \text{prop}(\text{upd}^\vartheta(M)) \land M \in \mathcal{M}_{\psi}^{i-1}\}$ is the state of $\Psi$ after processing event stream $\mathcal{L}$ until position $i$ and after the update to the current point in time $\vartheta$ and
- $\mathcal{M}_{\psi}^{i+1,\vartheta} := \{\text{prop}(M'|M' \in \text{hdl}^\vartheta(M) \land M \in \mathcal{M}_{\psi}^{i,\vartheta})\}$ is the state of $\Psi$ after processing $\sigma$, i.e. after processing event stream $\mathcal{L}$ until position $i + 1$.

Table 3 outlines the set of markings that results when completely processing the event stream from Fig. 1 for compliance rule $c_1$. Note that marking $F$ ensures that $c_1$ is satisfied for the request of Mr. Smith as highlighted in Fig. 18. In turn, Figs. 14(B′′′), 17(D), 18(I), and 18(J) highlight conflicts regarding the data, control flow, time, and resource perspectives.
Fig. 17: Effect propagation after handling data events and data conflict

Note that such conflicts only indicate why the considered events do not constitute a solution of a particular compliance rule. However, there might be another set of events that provide a satisfaction. In turn, Fig. 15(d2) indicates, which data values shall be read by task \textit{solvency check} and how task \textit{approval} shall be performed afterwards in order to satisfy $c_1$. Hence, Fig. 15(d2) may be utilized as recommendation to users in order to proactively ensure compliance.

<table>
<thead>
<tr>
<th>#</th>
<th>Request</th>
<th>Approval</th>
<th>Solvency</th>
<th>Approval</th>
<th>Customer</th>
<th>Amount</th>
<th>Rating</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>124</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>592</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>C</td>
<td>124</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>D</td>
<td>592</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>E</td>
<td>124</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>F</td>
<td>592</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>G</td>
<td>124</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>H</td>
<td>124</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>I</td>
<td>592</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>J</td>
<td>124</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>
Fig. 18: Fulfilling marking as well as control-flow and resource conflicts
6 Compliance Analytics

Another fundamental requirement of any compliance monitoring approach is to provide detailed user feedback. In this context, Section 6.1 introduces well-defined compliance assessments that not only highlight compliance violations, but also identify their causes. In addition, Section 6.2 shows how compliance metrics can be specified and how corresponding measurements can be obtained. Finally, Section 6.3 provide recommendations that support users in selecting the next process steps, whose execution will ensure compliance.

6.1 Compliance Assessments

Providing compliance assessments requires the analysis of the state of an eCRG and the markings it contains in order to clearly determine the corresponding states of compliance; i.e., either Not_Activated, Violable, Satisfied, Pending, or Violated (cf. Section 5.2). For this purpose, first of all, Def. 12 introduces a partial order over markings. The latter allows clustering the markings that correspond to each other as well as to specify and distinguish between different activations of an eCRG in Def. 13. Based on the latter, Def.14 finally provides compliance assessments; i.e., calculates the state of compliance (cf. Section 5.2) in order to enable simple and intuitive classifications.

**Definition 12 (Extensions of Markings).**

Let $\Psi = (N, E, \circ, \text{src}, \text{tgt}, \text{at}, \text{pt})$ be an eCRG and $M_\Psi$ the current state of compliance of $\Psi$. Further, let $M = (m, \text{res}, \text{val}, \text{id}, t_s, t_e)$ and $M' = (m', \text{res}', \text{val}', \text{id}', t'_s, t'_e)$ be two markings of $\Psi$; i.e., $M, M' \in M$. Then: We say

- $M$ is extended by $M'$ and write $M \mathbin{\text{extends}} M'$, iff $M'$ only differs from $M$ in markings that are neither Running ($\triangleright$) nor Completed ($\checkmark$) in $M$, i.e.,

$$M \mathbin{\subseteq} M' \Leftrightarrow \forall n \in T \cup M : m(n) \not\in \{\triangleright, \checkmark\} \lor \text{id}(n) = \text{id}'(n)$$

**Definition 13 (Activated Markings).**

Let $\Psi = (N, E, \circ, \text{type}, \text{src}, \text{tgt}, \text{at}, \text{pt})$ be an eCRG and $M_\Psi$ be the current state of compliance of $\Psi$. Further, let $M = (m, \text{res}, \text{val}, \text{id}, t_s, t_e)$ be a marking of $\Psi$; i.e., $M, M' \in M$. Then: We say

- $M$ is activated or an activation of $\Psi$ (written as $\text{Activated}(M)$), iff $M$ marks each element of the antecedence occurrence pattern as satisfied, but does not satisfy any element of the antecedence absence pattern. Furthermore, neither extends $M$ another activated marking nor exists a marking in $M$ that extends $M$ and satisfies a condition of the antecedence absence pattern.

$$\text{Activated}(M) \Leftrightarrow \left( \forall \lambda \in A^\circ : m(\lambda) = \checkmark \right) \land \left( \forall \alpha \in I^\circ_{\Psi} : m(\alpha) \in \{\square, \bigtriangleup, \times\} \land \text{id}(\alpha) = \epsilon \right) \land \left( \forall M_2 \in M \text{ with } M \mathbin{\subseteq} M_2 : \forall \alpha \in I^\circ_{\Psi} : m_2(\alpha) \neq \checkmark \right) \land \left( \forall M_3 \in M \text{ with } M_3 \mathbin{\subseteq} M : \lnot \text{Activated}(M_3) \right)$$
Definition 14 (Compliance Assessments).

Let $\Psi = (N,E,\phi,\text{type},\text{src},\text{tgt},\text{at},\text{pt})$ be an eCRG and $M_\phi$ be the current state of $\Psi$. Further, let $M \in M$ be an activated marking; i.e. $\text{Activated}(M)$.

Then: We say

- $M$ is **temporally satisfied** (written as $\text{TempSatisfied}(M)$), iff there exists a marking $M_2$ that extends $M$ and satisfies the consequence; i.e. $M_2$ marks each element of the consequence occurrence pattern with $\text{Satisfied}(\checkmark)$ and does not satisfy any element of the consequence absence pattern. Further, there exists no marking in $M$ that extends $M_2$ and satisfies a condition of the consequence absence pattern.

\[
\text{TempSatisfied}(M) :\Leftrightarrow \left( \exists M_2 \in M \text{ with } M \leq M_2 : (\forall \lambda \in A^CO_\phi : m_2(\lambda) = \checkmark) \land \left( \forall \alpha \in IG^A_\phi : m_2(\alpha) \in \{\square, \triangle, \times\} \land \text{id}(\alpha) = \epsilon \right) \land \left( \forall M_3 \in M \text{ with } M \leq M_2 \leq M_3 : \forall \alpha \in IG^A_\phi : m_3(\alpha) \neq \checkmark \right) \right)
\]

- $M$ is **violable (violably satisfied)** (written as $\text{Violable}(M)$), iff $M$ is temporarily satisfied, but for each $M_2$ extending $M$ and satisfying the consequence pattern, there remains at least one consequence absence node that has not been marked with $\text{Skipped}(\times)$ yet; i.e. that node might be executed in the following steps.

\[
\text{Violable} :\Leftrightarrow \text{TempSatisfied}(M) \land \left( \forall M_2 \in M \text{ with } M \leq M_2 : (\forall \lambda \in A^CO_\phi : m_2(\lambda) = \checkmark) \land \left( \forall \alpha \in IG^A_\phi : m_2(\alpha) \in \{\square, \triangle, \times\} \land \text{id}(\alpha) = \epsilon \right) \land \left( \forall M_3 \in M \text{ with } M \leq M_2 \leq M_3 : \forall \alpha \in IG^A_\phi : m_3(\alpha) \neq \checkmark \right) \land \left( \exists \alpha \in IG^A_\phi : m_2(\alpha) \in \{\square, \triangle\} \right) \right)
\]

- $M$ is **(permanently) satisfied** (written as $\text{Satisfied}(M)$), iff $M$ is temporally satisfied, but not violable; i.e.,

\[
\text{Satisfied}(M) :\Leftrightarrow \text{TempSatisfied}(M) \land \neg \text{Violable}(M)
\]

- $M$ is **(permanently) violated** (written as $\text{Violated}(M)$), iff each marking $M_2$ extending $M$ marks an element of the consequence occurrence pattern with $\text{Skipped}(\times)$ or there exists a marking $M_3$ extending $M_2$ and satisfying a condition of the consequence absence pattern.

\[
\text{Violated}(M) :\Leftrightarrow \left( \forall M_2 \in M \text{ with } M \leq M_2 : (\exists \lambda \in A^CO_\phi : m_2(\lambda) = \times) \lor \left( \exists M_3 \in M \text{ with } M \leq M_2 \leq M_3 : \exists \alpha \in IG^A_\phi : m_3(\alpha) = \checkmark \right) \right)
\]

- $M$ is **pending** (written as $\text{Pending}(M)$), iff $M$ is neither (temporally) satisfied nor violated yet.

\[
\text{Pending}(M) :\Leftrightarrow \neg \text{TempSatisfied}(M) \land \neg \text{Violated}(M)
\]

Table 4 applies Def. 14 to markings $A$ and $B$ from Table 3. Note that $A$ and $B$ are the only activations of $c_1$ in $M_1^{<99}$, since all other markings extend either marking $A$ or $B$ (except the initial marking 0). In particular, Table 4 shows that $c_1$ is activated twice; once satisfied and once violated. Further, Table 4 emphasises the events that complete the activations (39+58), the fulfillment (95), and the violation (99).
Table 4: Compliance assessments and metrics

<table>
<thead>
<tr>
<th>ID</th>
<th>Extensions</th>
<th>Activated</th>
<th>TempSatisfied</th>
<th>Violable</th>
<th>Satisfied</th>
<th>Violated</th>
<th>Pending</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>{A, C, F, G}</td>
<td>39...</td>
<td>95...</td>
<td>95...</td>
<td>39-95</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>{B, D, E, H}</td>
<td>58...</td>
<td>99...</td>
<td>58-99</td>
<td>00...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

6.2 Compliance Metrics and Measures

In the context of multiple instances and activations of compliance rules, providing compliance assessments for single activations is no longer sufficient. In turn, additional summaries on an higher level are required. For this purpose, compliance assessments (cf. Sec. 6.1) are combined to realize more sophisticated compliance metrics and measures. First of all, this section formally specifies the cardinality of properties in Def. 15 and, then, shows how the latter can be utilized for calculating compliance metrics and measures.

Definition 15 (Cardinality of a Property).
Let \( \Psi = (N, E, \circ, \text{type}, \text{src}, \text{tgt}, \text{at}, \text{pt}) \) be an eCRG and \( \mathcal{M}_\Psi \) be the current state of compliance of compliance rule \( \Psi \). Then:

\[
\#_{\Psi}^{\text{Prop}} := |\{M \in \mathcal{M}_\Psi | \text{Prop}(M)\}| \text{ is the cardinality of property Prop; i.e., the number of markings with property Prop} \in \{ \text{Activated, TempSatisfied, Violable, Satisfied, Violated, Pending} \}.
\]

Note that it is easy to specify metrics based on Def. 15. As example consider Table 5, which refers to 3 metrics. In particular, the compliance rate \( \mu_1 \), the critical rate \( \mu_2 \), and the violation rate \( \mu_3 \) are defined and calculated with respect to the motivating example from Section 3. (cf. Fig. 2 and Table 4).

Table 5: Compliance assessments and metrics

<table>
<thead>
<tr>
<th>#</th>
<th>Date</th>
<th>Time</th>
<th>( \mu_1(c_1) )</th>
<th>( \mu_2(c_1) )</th>
<th>( \mu_3(c_1) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1/7/2013</td>
<td>15:00</td>
<td>n.d.</td>
<td>n.d.</td>
<td>n.d.</td>
</tr>
<tr>
<td>1</td>
<td>1/7/2013</td>
<td>17:00</td>
<td>( \frac{2}{4} \times 0.00 )</td>
<td>( \frac{2+1}{4} = 1.00 )</td>
<td>( \frac{2}{4} = 0.00 )</td>
</tr>
<tr>
<td>2</td>
<td>1/7/2013</td>
<td>19:00</td>
<td>( \frac{2}{4} = 0.00 )</td>
<td>( \frac{2+1}{4} = 1.00 )</td>
<td>( \frac{2}{4} = 0.00 )</td>
</tr>
<tr>
<td>3</td>
<td>2/7/2013</td>
<td>17:00</td>
<td>( \frac{2}{4} = 0.00 )</td>
<td>( \frac{2+2}{4} = 1.00 )</td>
<td>( \frac{4}{2} = 0.00 )</td>
</tr>
<tr>
<td>4</td>
<td>2/7/2013</td>
<td>18:18</td>
<td>( \frac{2}{4} = 0.50 )</td>
<td>( \frac{2+1}{4} = 0.50 )</td>
<td>( \frac{4}{4} = 0.00 )</td>
</tr>
<tr>
<td>5</td>
<td>2/7/2013</td>
<td>19:00</td>
<td>( \frac{2}{4} = 0.50 )</td>
<td>( \frac{2+2}{4} = 1.00 )</td>
<td>( \frac{4}{4} = 0.00 )</td>
</tr>
</tbody>
</table>

Table 5 shows the progress of the compliance rate \( \mu_1(c_1) \), critical rate \( \mu_2(c_1) \) and violation rate \( \mu_3(c_1) \) for compliance rule \( c_1 \) over time and along the event log from Fig. 2. As long as no request is received, there is no activation of \( c_1 \) and hence all rates are not yet defined (cf. line 0). Due the request from Mr. Smith at 1/7/2013 – 15:27 compliance rule \( c_1 \).
is activated once in state Pending. Therefore, the critical rate $\mu_2(c_1)$ becomes 1.00, but the compliance rate $\mu_1(c_1)$ as well as violation rate $\mu_3(c_1)$ become 0.00 (cf. line 1). In turn, the values of $\mu_1(c_1)$, $\mu_2(c_1)$ and $\mu_3(c_1)$ are neither changed by the request from Mrs. John at 1/7/2013 – 18:03, which activates $c_2$ the second time, nor by the solvency check at 2/7/2013 – 15:43 (cf. line 2+3). This is due to the fact that the second activation of $c_1$ starts also in state pending as well as the first activation remains pending even after the solvency check. However, when the request of Mr. Smith is approved at 2/7/2013 – 18:13, the corresponding activation is Satisfied. Hence, the compliance rate $\mu_1(c_1)$ increases to 0.50 and critical rate $\mu_2(c_1)$ decreases to 0.50 as well, whereas the violation rate $\mu_3(c_1)$ stagnates at 0.00 (cf. line 4). Finally, the second activation of $c_1$ becomes Violated by the approval at 2/7/2013 – 18:19. Accordingly, no activation remains Pending and the critical rate $\mu_2(c_1)$ drops to 0.00. In turn, the violation rate $\mu_3(c_1)$ increases to 0.50 as the compliance rate $\mu_1(c_1)$ did before.

### 6.3 Recommendations

Section 5.4 argues that steps ensuring the compliant continuation of business process instances can be recommend based on markings. However, this requires the selection of appropriate markings. Note that the simple approach taking the marking that provides the most extensive extension of an activation does not always fit. Consider Table 6, which provides the set of markings $\mathcal{M}_s^{\psi}$ after processing the event log from Fig. 2 up to event 95. For both activations $A$ and $B$ there is an extending marking ($\mu \leq C$ and $B \leq D$). However, in the first case, $C$ provides the adequate recommendation, whereas in the second case the recommendation is not $D$, but $B$.

<table>
<thead>
<tr>
<th>#</th>
<th>Request</th>
<th>Approved (CA)</th>
<th>Solvency Check</th>
<th>Approved (CO)</th>
<th>Denied</th>
<th>Amount</th>
<th>Rating</th>
<th>Activated</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Mrs. John</td>
<td>592</td>
<td>smith</td>
<td>592</td>
<td>Smith</td>
<td>27,000</td>
<td>Satisfied</td>
<td>–</td>
</tr>
<tr>
<td>2</td>
<td>Mr. Smith</td>
<td>592</td>
<td>smith</td>
<td>592</td>
<td>Smith</td>
<td>27,000</td>
<td>Satisfied</td>
<td>–</td>
</tr>
<tr>
<td>3</td>
<td>Mr. Smith</td>
<td>592</td>
<td>smith</td>
<td>592</td>
<td>Smith</td>
<td>27,000</td>
<td>Satisfied</td>
<td>–</td>
</tr>
</tbody>
</table>

Def. 16 provides a more advanced and proper selection of markings that may then be used as recommendation.

**Definition 16 (Recommendations).**

Let $\Psi = (N, E, \phi, type, src, tgt, at, pt)$ be an eCRG and $\mathcal{M}_\Psi$ be the current state of compliance rule $\Psi$. Further, let $M \in \mathcal{M}_\Psi$ be an activated and not violated marking; i.e., it holds: $\text{Activated}(M) \land \neg \text{Violated}(M)$. Then: We say $M_2 \in \mathcal{M}_\Psi$ is a simple recommendation for activation $M$ of compliance rule $\Psi$ (written as $M \preceq M_2$), if $M_2$ extends $M$ and $M_2$ neither skips any part of the consequence occurrence pattern nor satisfies any part of the consequence absence pattern nor if there is another marking $M_3 \in \mathcal{M}_\Psi$ further extending $M_2$, i.e.,

$$M \preceq M_2 \iff M \preceq M_2$$

$$\land \left( \exists \alpha \in \Gamma_\Psi^{CO} : m_2(\alpha) = \top \right)$$

$$\land \left( \exists M_3 \in \mathcal{M} \text{ with } M \preceq M_2 \preceq M_3 : \exists \alpha \in \Gamma_\Psi^{CA} : m_3(\phi(\alpha)) = \top \right)$$
- $M_2 \in \mathcal{M}_\Psi$ is a (real) recommendation for the activation of $\Psi$ through marking $M$ (written as $M \rightarrow_\mathcal{R} M_2$), iff $M_2$ is a simple recommendation for $M$ and there is no other recommendation $M_3 \in \mathcal{M}_\Psi$ for $M$ extending $M_2$, i.e.,

$$M \rightarrow_\mathcal{R} M_2 \iff M \subseteq M_2 \land \exists M_3 \in \mathcal{M} \text{ with } M \subseteq M_3 : M_2 < M_3 \text{ (i.e., } M_2 \leq M_3 \land M_2 \neq M_3 \).$$

- $\mathcal{R} : \mathcal{M}_\Psi \rightarrow 2^{\mathcal{M}_\Psi} : \mathcal{R}(M) := \{ M' \in \mathcal{M}_\Psi | M \rightarrow_\mathcal{R} M' \}$ is the set of all recommendations for $M$.

Note that each marking $M$, which is activated and not violated, must have at least one simple and, hence, also one real recommendation. Otherwise, it would match the definition of violated (cf. Def. 14).

The application of Def. 16 on our example results in $\mathcal{R}(A) = \{ C \}$ and $\mathcal{R}(B) = \{ B \}$. Note that the application of $\mathcal{R}$ to an activation might still result in more than one recommendation. In this case, each recommendation corresponds to another, already started solution approach ensuring compliance with the respective rule. However, if only a single recommendation is desired, the most advanced recommendation (i.e., satisfying most consequence occurrence nodes) or the latest one can easily be selected.
7 Summary and Outlook

This report introduces an operational semantics for the extended compliance rule graph (eCRG) language [17, 18]. Beyond the control flow perspective, the latter supports the monitoring of the data, time, and resource perspectives as well as the monitoring of the interactions with partners. In particular, eCRGs are annotated with text, colors and symbols in order to visually highlight the current state of compliance. Furthermore, formal definitions specify how observed events continuously change and evolve these annotations. Furthermore, formal criteria for assessing compliance are provided, which, in turn, constitute the basis for compliance metrics introduced as examples. Finally, the recommendations are provided, which support users in selecting the next process steps, whose execution will ensure compliance.

As opposed to other approaches enabling the monitoring of business process compliance at run-time, the operational semantics of the eCRG language supports all 10 compliance monitoring functionalities (CMF) that have been proposed in [5] (cf. Table 7). In particular, full support of the control flow, data, time, and resource perspectives (CMF 1-3) is provided as well as interactions with partners are considered. The proposed approach assumes activities to be non-atomic, but stateful (CMF 4+5). Different instantiations (i.e., activations) of compliance rules are explicitly identified and are further extended to highlight the causes of compliance violations (CMF 6+9). Beyond detecting the latter (i.e. reactive monitoring), recommendations (i.e., proactive monitoring) are provided as well (CMF 7+8). Finally, the operational semantics of the eCRG language builds a suitable basis for the specification of compliance metrics in order to measure different degrees of compliance (CMF 10).

Next steps will be the implementation of a proof-of-concept prototype in order to evaluate the operational semantics for the eCRG language.

Note that this work was done within the research project C³Pro that deals with change and compliance in cross-organizational business processes [21]. Accordingly, our overall aim is to ensure multi-perspective compliance for all phases of the process life cycle. Hence, we will investigate a priori compliance checking with the eCRG at design time as well as we plan to consider compliance checking in the context of cross-organizational process changes and change propagation [44].
References

Liste der bisher erschienenen Ulmer Informatik-Berichte
Einige davon sind per FTP von ftp.informatik.uni-ulm.de erhältlich
Die mit * markierten Berichte sind vergriffen

List of technical reports published by the University of Ulm
Some of them are available by FTP from ftp.informatik.uni-ulm.de
Reports marked with * are out of print

91-01  Ker-I Ko, P. Orponen, U. Schöning, O. Watanabe
       Instance Complexity

91-02* K. Gladitz, H. Fassbender, H. Vogler
       Compiler-Based Implementation of Syntax-Directed Functional Programming

91-03* Alfons Geser
       Relative Termination

91-04* J. Köbler, U. Schöning, J. Toran
       Graph Isomorphism is low for PP

91-05 Johannes Köbler, Thomas Thierauf
       Complexity Restricted Advice Functions

91-06* Uwe Schöning
       Recent Highlights in Structural Complexity Theory

91-07* F. Green, J. Köbler, J. Toran
       The Power of Middle Bit

91-08* V. Arvind, Y. Han, L. Hamachandra, J. Köbler, A. Lozano, M. Mundhenk, A. Ogiwara,
       U. Schöning, R. Silvestri, T. Thierauf
       Reductions for Sets of Low Information Content

92-01* Vikraman Arvind, Johannes Köbler, Martin Mundhenk
       On Bounded Truth-Table and Conjunctive Reductions to Sparse and Tally Sets

92-02* Thomas Noll, Heiko Vogler
       Top-down Parsing with Simulataneous Evaluation of Noncircular Attribute Grammars

92-03  Fakultät für Informatik
       17. Workshop über Komplexitätstheorie, effiziente Algorithmen und Datenstrukturen

92-04* V. Arvind, J. Köbler, M. Mundhenk
       Lowness and the Complexity of Sparse and Tally Descriptions

92-05* Johannes Köbler
       Locating $\text{P/poly}$ Optimally in the Extended Low Hierarchy

92-06* Armin Kühnemann, Heiko Vogler
       Synthesized and inherited functions - a new computational model for syntax-directed semantics

92-07* Heinz Fassbender, Heiko Vogler
       A Universal Unification Algorithm Based on Unification-Driven Leftmost Outermost Narrowing
92-08* Uwe Schöning  
On Random Reductions from Sparse Sets to Tally Sets

92-09* Hermann von Hassel, Laura Martignon  
Consistency in Stochastic Network

92-10 Michael Schmitt  
A Slightly Improved Upper Bound on the Size of Weights Sufficient to Represent Any Linearly Separable Boolean Function

92-11 Johannes Köbler, Seinosuke Toda  
On the Power of Generalized MOD-Classes

92-12 V. Arvind, J. Köbler, M. Mundhenk  
Reliable Reductions, High Sets and Low Sets

92-13 Alfons Geser  
On a monotonic semantic path ordering

92-14* Joost Engelfriet, Heiko Vogler  
The Translation Power of Top-Down Tree-To-Graph Transducers

93-01 Alfred Lupper, Konrad Froitzheim  
AppleTalk Link Access Protocol basierend auf dem Abstract Personal Communications Manager

The COCOON Object Model

93-03 Thomas Thierauf, Seinosuke Toda, Osamu Watanabe  
On Sets Bounded Truth-Table Reducible to P-selective Sets

93-04 Jin-Yi Cai, Frederic Green, Thomas Thierauf  
On the Correlation of Symmetric Functions

93-05 K.Kuhn, M.Reichert, M. Nathe, T. Beuter, C. Heinlein, P. Dadam  
A Conceptual Approach to an Open Hospital Information System

93-06 Klaus Gaßner  
Rechnerunterstützung für die konzeptuelle Modellierung

93-07 Ullrich Keßler, Peter Dadam  
Towards Customizable, Flexible Storage Structures for Complex Objects

94-01 Michael Schmitt  
On the Complexity of Consistency Problems for Neurons with Binary Weights

94-02 Armin Kühnemann, Heiko Vogler  
A Pumping Lemma for Output Languages of Attributed Tree Transducers

94-03 Harry Buhrman, Jim Kadin, Thomas Thierauf  
On Functions Computable with Nonadaptive Queries to NP

94-04 Heinz Faßbender, Heiko Vogler, Andrea Wedel  
Implementation of a Deterministic Partial E-Unification Algorithm for Macro Tree Transducers
<table>
<thead>
<tr>
<th>Year</th>
<th>Authors</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>94-05</td>
<td>V. Arvind, J. Köbler, R. Schuler</td>
<td>On Helping and Interactive Proof Systems</td>
</tr>
<tr>
<td>94-06</td>
<td>Christian Kalus, Peter Dadam</td>
<td>Incorporating record subtyping into a relational data model</td>
</tr>
<tr>
<td>94-07</td>
<td>Markus Tresch, Marc H. Scholl</td>
<td>A Classification of Multi-Database Languages</td>
</tr>
<tr>
<td>94-08</td>
<td>Friedrich von Henke, Harald Rueß</td>
<td>Arbeitstreffen Typtheorie: Zusammenfassung der Beiträge</td>
</tr>
<tr>
<td>94-10</td>
<td>Axel Dold</td>
<td>Formalisierung schematischer Algorithmen</td>
</tr>
<tr>
<td>94-11</td>
<td>Johannes Köbler, Osamu Watanabe</td>
<td>New Collapse Consequences of NP Having Small Circuits</td>
</tr>
<tr>
<td>94-12</td>
<td>Rainer Schuler</td>
<td>On Average Polynomial Time</td>
</tr>
<tr>
<td>94-13</td>
<td>Rainer Schuler, Osamu Watanabe</td>
<td>Towards Average-Case Complexity Analysis of NP Optimization Problems</td>
</tr>
<tr>
<td>94-14</td>
<td>Wolfram Schulte, Ton Vullinghs</td>
<td>Linking Reactive Software to the X-Window System</td>
</tr>
<tr>
<td>94-15</td>
<td>Alfred Lupper</td>
<td>Namensverwaltung und Adressierung in Distributed Shared Memory-Systemen</td>
</tr>
<tr>
<td>94-16</td>
<td>Robert Regn</td>
<td>Verteilte Unix-Betriebssysteme</td>
</tr>
<tr>
<td>94-17</td>
<td>Helmuth Partsch</td>
<td>Again on Recognition and Parsing of Context-Free Grammars: Two Exercises in Transformational Programming</td>
</tr>
<tr>
<td>94-18</td>
<td>Helmuth Partsch</td>
<td>Transformational Development of Data-Parallel Algorithms: an Example</td>
</tr>
<tr>
<td>95-01</td>
<td>Oleg Verbitsky</td>
<td>On the Largest Common Subgraph Problem</td>
</tr>
<tr>
<td>95-02</td>
<td>Uwe Schöning</td>
<td>Complexity of Presburger Arithmetic with Fixed Quantifier Dimension</td>
</tr>
<tr>
<td>95-03</td>
<td>Harry Buhrman, Thomas Thierauf</td>
<td>The Complexity of Generating and Checking Proofs of Membership</td>
</tr>
<tr>
<td>95-04</td>
<td>Rainer Schuler, Tomoyuki Yamakami</td>
<td>Structural Average Case Complexity</td>
</tr>
<tr>
<td>95-05</td>
<td>Klaus Achatz, Wolfram Schulte</td>
<td>Architecture Independent Massive Parallelization of Divide-And-Conquer Algorithms</td>
</tr>
</tbody>
</table>
95-06 Christoph Karg, Rainer Schuler
Structure in Average Case Complexity

95-07 P. Dadam, K. Kuhn, M. Reichert, T. Beuter, M. Nathe
ADEPT: Ein integrierender Ansatz zur Entwicklung flexibler, zuverlässiger
kooperierender Assistenzsysteme in klinischen Anwendungsumgebungen

95-08 Jürgen Kehrer, Peter Schulthes
Aufbereitung von gescannten Röntgenbildern zur filmlosen Diagnostik

95-09 Hans-Jörg Burtschick, Wolfgang Lindner
On Sets Turing Reducible to P-Selective Sets

95-10 Boris Hartmann
Berücksichtigung lokaler Randbedingung bei globaler Zieloptimierung mit neuronalen
Netzen am Beispiel Truck Backer-Upper

95-11 Thomas Beuter, Peter Dadam:
Prinzipien der Replikationskontrolle in verteilten Systemen

95-12 Klaus Achatz, Wolfram Schulte
Massive Parallelization of Divide-and-Conquer Algorithms over Powerlists

95-13 Andrea Mößle, Heiko Vogler
Efficient Call-by-value Evaluation Strategy of Primitive Recursive Program Schemes

95-14 Axel Dold, Friedrich W. von Henke, Holger Pfeifer, Harald Rueß
A Generic Specification for Verifying Peephole Optimizations

96-01 Ercüment Canver, Jan-Tecker Gayen, Adam Moik
Formale Entwicklung der Steuerungssoftware für eine elektrisch ortsbediente Weiche
mit VSE

96-02 Bernhard Nebel
Solving Hard Qualitative Temporal Reasoning Problems: Evaluating the Efficiency of
Using the ORD-Horn Class

96-03 Ton Vullinghs, Wolfram Schulte, Thilo Schwinn
An Introduction to TkGofer

96-04 Thomas Beuter, Peter Dadam
Anwendungsspezifische Anforderungen an Workflow-Mangement-Systeme am
Beispiel der Domäne Concurrent-Engineering

96-05 Gerhard Schellhorn, Wolfgang Ahrendt
Verification of a Prolog Compiler - First Steps with KIV

96-06 Manindra Agrawal, Thomas Thierauf
Satisfiability Problems

96-07 Vikraman Arvind, Jacobo Torán
A nonadaptive NC Checker for Permutation Group Intersection

96-08 David Cyrluk, Oliver Möller, Harald Rueß
An Efficient Decision Procedure for a Theory of Fix-Sized Bitvectors with
Composition and Extraction
96-09  Bernd Biechele, Dietmar Ernst, Frank Houdek, Joachim Schmid, Wolfram Schulte
Erfahrungen bei der Modellierung eingebetteter Systeme mit verschiedenen SA/RT–
Ansätzen

96-10  Falk Bartels, Axel Dold, Friedrich W. von Henke, Holger Pfeifer, Harald Rueß
Formalizing Fixed-Point Theory in PVS

96-11  Axel Dold, Friedrich W. von Henke, Holger Pfeifer, Harald Rueß
Mechanized Semantics of Simple Imperative Programming Constructs

96-12  Axel Dold, Friedrich W. von Henke, Holger Pfeifer, Harald Rueß
Generic Compilation Schemes for Simple Programming Constructs

96-13  Klaus Achatz, Helmut Partsch
From Descriptive Specifications to Operational ones: A Powerful Transformation
Rule, its Applications and Variants

97-01  Jochen Messner
Pattern Matching in Trace Monoids

97-02  Wolfgang Lindner, Rainer Schuler
A Small Span Theorem within P

97-03  Thomas Bauer, Peter Dadam
A Distributed Execution Environment for Large-Scale Workflow Management
Systems with Subnets and Server Migration

97-04  Christian Heinlein, Peter Dadam
Interaction Expressions - A Powerful Formalism for Describing Inter-Workflow
Dependencies

97-05  Vikraman Arvind, Johannes Köbler
On Pseudorandomness and Resource-Bounded Measure

97-06  Gerhard Partsch
Punkt-zu-Punkt- und Mehrpunkt-basierende LAN-Integrationsstrategien für den
digitalen Mobilfunkstandard DECT

97-07  Manfred Reichert, Peter Dadam
ADEPT_{flex} - Supporting Dynamic Changes of Workflows Without Loosing Control

97-08  Hans Braxmeier, Dietmar Ernst, Andrea Mößle, Heiko Vogler
The Project NoName - A functional programming language with its development
environment

97-09  Christian Heinlein
Grundlagen von Interaktionsausdrücken

97-10  Christian Heinlein
Graphische Repräsentation von Interaktionsausdrücken

97-11  Christian Heinlein
Sprachtheoretische Semantik von Interaktionsausdrücken
<table>
<thead>
<tr>
<th>Paper ID</th>
<th>Authors</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>97-12</td>
<td>Gerhard Schellhorn, Wolfgang Reif</td>
<td>Proving Properties of Finite Enumerations: A Problem Set for Automated Theorem Provers</td>
</tr>
<tr>
<td>97-13</td>
<td>Dietmar Ernst, Frank Houdek, Wolfram Schulte, Thilo Schwinn</td>
<td>Experimenteller Vergleich statischer und dynamischer Softwareprüfung für eingebettete Systeme</td>
</tr>
<tr>
<td>97-14</td>
<td>Wolfgang Reif, Gerhard Schellhorn</td>
<td>Theorem Proving in Large Theories</td>
</tr>
<tr>
<td>97-15</td>
<td>Thomas Wennenkers</td>
<td>Asymptotik rekurrenter neuronaler Netze mit zufälligen Kopplungen</td>
</tr>
<tr>
<td>97-16</td>
<td>Peter Dadam, Klaus Kuhn, Manfred Reichert</td>
<td>Clinical Workflows - The Killer Application for Process-oriented Information Systems?</td>
</tr>
<tr>
<td>97-17</td>
<td>Mohammad Ali Livani, Jörg Kaiser</td>
<td>EDF Consensus on CAN Bus Access in Dynamic Real-Time Applications</td>
</tr>
<tr>
<td>98-01</td>
<td>Daniela Damm, Lutz Claes, Friedrich W. von Henke, Alexander Seitz, Adelinde Uhrmacher, Steffen Wolf</td>
<td>Ein fallbasiertes System für die Interpretation von Literatur zur Knochenheilung</td>
</tr>
<tr>
<td>98-02</td>
<td>Thomas Bauer, Peter Dadam</td>
<td>Architekturen für skalierbare Workflow-Management-Systeme - Klassifikation und Analyse</td>
</tr>
<tr>
<td>98-03</td>
<td>Marko Luther, Martin Strecker</td>
<td>A guided tour through Typelab</td>
</tr>
<tr>
<td>98-04</td>
<td>Heiko Neumann, Luiz Pessoa</td>
<td>Visual Filling-in and Surface Property Reconstruction</td>
</tr>
<tr>
<td>98-05</td>
<td>Ercüment Canver</td>
<td>Formal Verification of a Coordinated Atomic Action Based Design</td>
</tr>
<tr>
<td>98-06</td>
<td>Andreas Küchler</td>
<td>On the Correspondence between Neural Folding Architectures and Tree Automata</td>
</tr>
<tr>
<td>98-07</td>
<td>Heiko Neumann, Thorsten Hansen, Luiz Pessoa</td>
<td>Interaction of ON and OFF Pathways for Visual Contrast Measurement</td>
</tr>
<tr>
<td>98-08</td>
<td>Thomas Wennenkers</td>
<td>Synfire Graphs: From Spike Patterns to Automata of Spiking Neurons</td>
</tr>
<tr>
<td>98-09</td>
<td>Thomas Bauer, Peter Dadam</td>
<td>Variable Migration von Workflows in ADEPT</td>
</tr>
<tr>
<td>98-10</td>
<td>Heiko Neumann, Wolfgang Sepp</td>
<td>Recurrent V1 – V2 Interaction in Early Visual Boundary Processing</td>
</tr>
<tr>
<td></td>
<td>Author(s)</td>
<td>Title</td>
</tr>
<tr>
<td>---</td>
<td>------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>98-11</td>
<td>Frank Houdek, Dietmar Ernst, Thilo Schwinn</td>
<td>Prüfen von C–Code und Statmate/Matlab–Spezifikationen: Ein Experiment</td>
</tr>
<tr>
<td>98-12</td>
<td>Gerhard Schellhorn</td>
<td>Proving Properties of Directed Graphs: A Problem Set for Automated Theorem Provers</td>
</tr>
<tr>
<td>98-13</td>
<td>Gerhard Schellhorn, Wolfgang Reif</td>
<td>Theorems from Compiler Verification: A Problem Set for Automated Theorem Provers</td>
</tr>
<tr>
<td>98-14</td>
<td>Mohammad Ali Livani</td>
<td>SHARE: A Transparent Mechanism for Reliable Broadcast Delivery in CAN</td>
</tr>
<tr>
<td>98-15</td>
<td>Mohammad Ali Livani, Jörg Kaiser</td>
<td>Predictable Atomic Multicast in the Controller Area Network (CAN)</td>
</tr>
<tr>
<td>99-01</td>
<td>Susanne Boll, Wolfgang Klas, Utz Westermann</td>
<td>A Comparison of Multimedia Document Models Concerning Advanced Requirements</td>
</tr>
<tr>
<td>99-02</td>
<td>Thomas Bauer, Peter Dadam</td>
<td>Verteilungsmodelle für Workflow-Management-Systeme - Klassifikation und Simulation</td>
</tr>
<tr>
<td>99-03</td>
<td>Uwe Schöning</td>
<td>On the Complexity of Constraint Satisfaction</td>
</tr>
<tr>
<td>99-04</td>
<td>Ercument Canver</td>
<td>Model-Checking zur Analyse von Message Sequence Charts über Statecharts</td>
</tr>
<tr>
<td>99-05</td>
<td>Johannes Köbler, Wolfgang Lindner, Rainer Schuler</td>
<td>Derandomizing RP if Boolean Circuits are not Learnable</td>
</tr>
<tr>
<td>99-06</td>
<td>Utz Westermann, Wolfgang Klas</td>
<td>Architecture of a DataBlade Module for the Integrated Management of Multimedia Assets</td>
</tr>
<tr>
<td>99-08</td>
<td>Vikraman Arvind, Johannes Köbler</td>
<td>Graph Isomorphism is Low for ZPP^NP and other Lowness results</td>
</tr>
<tr>
<td>99-09</td>
<td>Thomas Bauer, Peter Dadam</td>
<td>Efficient Distributed Workflow Management Based on Variable Server Assignments</td>
</tr>
<tr>
<td>2000-02</td>
<td>Thomas Bauer, Peter Dadam</td>
<td>Variable Serverzuordnungen und komplexe Bearbeiterzuordnungen im Workflow-Management-System ADEPT</td>
</tr>
<tr>
<td>2000-03</td>
<td>Gregory Baratoff, Christian Toepfer, Heiko Neumann</td>
<td>Combined space-variant maps for optical flow based navigation</td>
</tr>
</tbody>
</table>
2000-04, Wolfgang Gehring
Ein Rahmenwerk zur Einführung von Leistungspunktsystemen

2000-05, Susanne Boll, Christian Heinlein, Wolfgang Klas, Jochen Wandel
Intelligent Prefetching and Buffering for Interactive Streaming of MPEG Videos

2000-06, Wolfgang Reif, Gerhard Schellhorn, Andreas Thums
Fehlersuche in Formalen Spezifikationen

2000-07, Gerhard Schellhorn, Wolfgang Reif (eds.)

2000-08, Thomas Bauer, Manfred Reichert, Peter Dadam
Effiziente Durchführung von Prozessmigrationen in verteilten Workflow-Management-Systemen

2000-09, Thomas Bauer, Peter Dadam
Vermeidung von Überlastsituationen durch Replikation von Workflow-Servern in ADEPT

2000-10, Thomas Bauer, Manfred Reichert, Peter Dadam
Adaptives und verteiltes Workflow-Management

2000-11, Christian Heinlein
Workflow and Process Synchronization with Interaction Expressions and Graphs

2001-01, Hubert Hug, Rainer Schuler
DNA-based parallel computation of simple arithmetic

2001-02, Friedhelm Schwenker, Hans A. Kestler, Günther Palm
3-D Visual Object Classification with Hierarchical Radial Basis Function Networks

2001-03, Hans A. Kestler, Friedhelm Schwenker, Günther Palm
RBF network classification of ECGs as a potential marker for sudden cardiac death

2001-04, Christian Dietrich, Friedhelm Schwenker, Klaus Riede, Günther Palm
Classification of Bioacoustic Time Series Utilizing Pulse Detection, Time and Frequency Features and Data Fusion

2002-01, Stefanie Rinderle, Manfred Reichert, Peter Dadam
Effiziente Verträglichkeitsprüfung und automatische Migration von Workflow-Instanzen bei der Evolution von Workflow-Schemata

2002-02, Walter Guttmann
Deriving an Applicative Heapsort Algorithm

A Mechanically Verified Compiling Specification for a Realistic Compiler

2003-01, Manfred Reichert, Stefanie Rinderle, Peter Dadam
A Formal Framework for Workflow Type and Instance Changes Under Correctness Checks

2003-02, Stefanie Rinderle, Manfred Reichert, Peter Dadam
Supporting Workflow Schema Evolution By Efficient Compliance Checks
<table>
<thead>
<tr>
<th>Year</th>
<th>Author(s)</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>2003-03</td>
<td>Christian Heinlein</td>
<td>Safely Extending Procedure Types to Allow Nested Procedures as Values</td>
</tr>
<tr>
<td>2003-05</td>
<td>Christian Heinlein</td>
<td>Dynamic Class Methods in Java</td>
</tr>
<tr>
<td>2003-06</td>
<td>Christian Heinlein</td>
<td>Vertical, Horizontal, and Behavioural Extensibility of Software Systems</td>
</tr>
<tr>
<td>2003-07</td>
<td>Christian Heinlein</td>
<td>Safely Extending Procedure Types to Allow Nested Procedures as Values  (Corrected Version)</td>
</tr>
<tr>
<td>2003-08</td>
<td>Changling Liu, Jörg Kaiser</td>
<td>Survey of Mobile Ad Hoc Network Routing Protocols)</td>
</tr>
<tr>
<td>2004-01</td>
<td>Thom Frühwirth, Marc Meister (eds.)</td>
<td>First Workshop on Constraint Handling Rules</td>
</tr>
<tr>
<td>2004-03</td>
<td>Susanne Biundo, Thom Frühwirth, Günther Palm (eds.)</td>
<td>Poster Proceedings of the 27th Annual German Conference on Artificial Intelligence</td>
</tr>
<tr>
<td>2005-01</td>
<td>Armin Wolf, Thom Frühwirth, Marc Meister (eds.)</td>
<td>19th Workshop on (Constraint) Logic Programming</td>
</tr>
<tr>
<td>2005-02</td>
<td>Wolfgang Lindner (Hg.), Universität Ulm, Christopher Wolf (Hg.) KU Leuven</td>
<td>2. Krypto-Tag – Workshop über Kryptographie, Universität Ulm</td>
</tr>
<tr>
<td>2005-03</td>
<td>Walter Guttmann, Markus Maucher</td>
<td>Constrained Ordering</td>
</tr>
<tr>
<td>2006-01</td>
<td>Stefan Sarstedt</td>
<td>Model-Driven Development with ACTIVECHARTS, Tutorial</td>
</tr>
<tr>
<td>2006-03</td>
<td>Jens Kohlmeyer, Alexander Raschke, Ramin Tavakoli Kolagari</td>
<td>Eine qualitative Untersuchung zur Produktlinien-Integration über Organisationsgrenzen hinweg</td>
</tr>
<tr>
<td>2006-04</td>
<td>Thorsten Liebig</td>
<td>Reasoning with OWL - System Support and Insights –</td>
</tr>
<tr>
<td>2008-01</td>
<td>H.A. Kestler, J. Messner, A. Müller, R. Schuler</td>
<td>On the complexity of intersecting multiple circles for graphical display</td>
</tr>
<tr>
<td>Year</td>
<td>Authors</td>
<td>Title</td>
</tr>
<tr>
<td>-------</td>
<td>------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>2008-03</td>
<td>Frank Raiser</td>
<td>Semi-Automatic Generation of CHR Solvers from Global Constraint Automata</td>
</tr>
<tr>
<td>2008-04</td>
<td>Ramin Tavakoli Kolagari, Alexander Raschke, Matthias Schneiderhan, Ian Alexander</td>
<td>Entscheidungsdocumentation bei der Entwicklung innovativer Systeme für produktlinien-basierte Entwicklungsprozesse</td>
</tr>
<tr>
<td>2008-05</td>
<td>Markus Kalb, Claudia Dittrich, Peter Dadam</td>
<td>Support of Relationships Among Moving Objects on Networks</td>
</tr>
<tr>
<td>2008-06</td>
<td>Matthias Frank, Frank Kargl, Burkhard Stiller (Hg.)</td>
<td>WMAN 2008 – KuVS Fachgespräch über Mobile Ad-hoc Netzwerke</td>
</tr>
<tr>
<td>2008-07</td>
<td>M. Maucher, U. Schöning, H.A. Kestler</td>
<td>An empirical assessment of local and population based search methods with different degrees of pseudorandomness</td>
</tr>
<tr>
<td>2008-08</td>
<td>Henning Wunderlich</td>
<td>Covers have structure</td>
</tr>
<tr>
<td>2008-09</td>
<td>Karl-Heinz Niggl, Henning Wunderlich</td>
<td>Implicit characterization of FPTIME and NC revisited</td>
</tr>
<tr>
<td>2008-10</td>
<td>Henning Wunderlich</td>
<td>On span-P^c and related classes in structural communication complexity</td>
</tr>
<tr>
<td>2008-11</td>
<td>M. Maucher, U. Schöning, H.A. Kestler</td>
<td>On the different notions of pseudorandomness</td>
</tr>
<tr>
<td>2008-12</td>
<td>Henning Wunderlich</td>
<td>On Toda’s Theorem in structural communication complexity</td>
</tr>
<tr>
<td>2008-13</td>
<td>Manfred Reichert, Peter Dadam</td>
<td>Realizing Adaptive Process-aware Information Systems with ADEPT2</td>
</tr>
<tr>
<td>2009-01</td>
<td>Peter Dadam, Manfred Reichert</td>
<td>The ADEPT Project: A Decade of Research and Development for Robust and Flexible Process Support Challenges and Achievements</td>
</tr>
</tbody>
</table>
2009-03 Alena Hallerbach, Thomas Bauer, Manfred Reichert
Correct Configuration of Process Variants in Provop

2009-04 Martin Bader
On Reversal and Transposition Medians

2009-05 Barbara Weber, Andreas Lanz, Manfred Reichert
Time Patterns for Process-aware Information Systems: A Pattern-based Analysis

2009-06 Stefanie Rinderle-Ma, Manfred Reichert
Adjustment Strategies for Non-Compliant Process Instances

Statistical Computing 2009 – Abstracts der 41. Arbeitstagung

2009-08 Ulrich Kreher, Manfred Reichert, Stefanie Rinderle-Ma, Peter Dadam
Effiziente Repräsentation von Vorlagen- und Instanzdaten in Prozess-Management-Systemen

2009-09 Dammertz, Holger, Alexander Keller, Hendrik P.A. Lensch
Progressive Point-Light-Based Global Illumination

2009-10 Dao Zhou, Christoph Müssel, Ludwig Lausser, Martin Hopfensitz, Michael Kühl, Hans A. Kestler
Boolean networks for modeling and analysis of gene regulation

2009-11 J. Hanika, H.P.A. Lensch, A. Keller
Two-Level Ray Tracing with Recordering for Highly Complex Scenes

2009-12 Stephan Buchwald, Thomas Bauer, Manfred Reichert
Durchgängige Modellierung von Geschäftsprozessen durch Einführung eines Abbildungsmodells: Ansätze, Konzepte, Notationen

2010-01 Hariolf Betz, Frank Raiser, Thom Frühwirth
A Complete and Terminating Execution Model for Constraint Handling Rules

2010-02 Ulrich Kreher, Manfred Reichert
Speichereffiziente Repräsentation instanzspezifischer Änderungen in Prozess-Management-Systemen

2010-03 Patrick Frey
Case Study: Engine Control Application

2010-04 Matthias Lohrmann und Manfred Reichert
Basic Considerations on Business Process Quality

2010-05 HA Kestler, H Binder, B Lausen, H-P Klenk, M Schmid, F Leisch (eds):
Statistical Computing 2010 - Abstracts der 42. Arbeitstagung

2010-06 Vera Künzle, Barbara Weber, Manfred Reichert
Object-aware Business Processes: Properties, Requirements, Existing Approaches
2011-01  Stephan Buchwald, Thomas Bauer, Manfred Reichert
Flexibilisierung Service-orientierter Architekturen

2011-02  Johannes Hanika, Holger Dammertz, Hendrik Lensch
Edge-Optimized À-Trous Wavelets for Local Contrast Enhancement with Robust Denoising

2011-03  Stefanie Kaiser, Manfred Reichert
Datenflussvarianten in Prozessmodellen: Szenarien, Herausforderungen, Ansätze

2011-04  Hans A. Kestler, Harald Binder, Matthias Schmid, Friedrich Leisch, Johann M. Kraus (eds):

2011-05  Vera Künzle, Manfred Reichert
PHILharmonicFlows: Research and Design Methodology

2011-06  David Knuplesch, Manfred Reichert
Ensuring Business Process Compliance Along the Process Life Cycle

2011-07  Marcel Dausend
Towards a UML Profile on Formal Semantics for Modeling Multimodal Interactive Systems

2011-08  Dominik Gessenharter
Model-Driven Software Development with ACTIVECHARTS - A Case Study

2012-01  Andreas Steigmiller, Thorsten Liebig, Birte Glimm
Extended Caching, Backjumping and Merging for Expressive Description Logics

2012-02  Hans A. Kestler, Harald Binder, Matthias Schmid, Johann M. Kraus (eds):
Statistical Computing 2012 - Abstracts der 44. Arbeitstagung

2012-03  Felix Schüssel, Frank Honold, Michael Weber
Influencing Factors on Multimodal Interaction at Selection Tasks

2012-04  Jens Kolb, Paul Hübner, Manfred Reichert
Model-Driven User Interface Generation and Adaption in Process-Aware Information Systems

2012-05  Matthias Lohrmann, Manfred Reichert
Formalizing Concepts for Efficacy-aware Business Process Modeling

2012-06  David Knuplesch, Rüdiger Pryss, Manfred Reichert
A Formal Framework for Data-Aware Process Interaction Models

2012-07  Clara Ayora, Victoria Torres, Barbara Weber, Manfred Reichert, Vicente Pelechano
Dealing with Variability in Process-Aware Information Systems: Language Requirements, Features, and Existing Proposals

2013-01  Frank Kargl
Abstract Proceedings of the 7th Workshop on Wireless and Mobile Ad-Hoc Networks (WMAN 2013)
2013-02  *Andreas Lanz, Manfred Reichert, Barbara Weber*
A Formal Semantics of Time Patterns for Process-aware Information Systems

2013-03  *Matthias Lohrmann, Manfred Reichert*
Demonstrating the Effectiveness of Process Improvement Patterns with Mining Results

2013-04  *Semra Catalkaya, David Knuplesch, Manfred Reichert*
Bringing More Semantics to XOR-Split Gateways in Business Process Models Based on Decision Rules

2013-05  *David Knuplesch, Manfred Reichert, Linh Thao Ly, Akhil Kumar, Stefanie Rinderle-Ma*
On the Formal Semantics of the Extended Compliance Rule Graph

2013-06  *Andreas Steigmiller, Birte Glimm, Thorsten Liebig*
Nominal Schema Absorption

2013-07  *Hans A. Kestler, Matthias Schmid, Florian Schmid, Dr. Markus Maucher, Johann M. Kraus (eds)*

2013-08  *Daniel Ott, Dr. Alexander Raschke*
Evaluating Benefits of Requirement Categorization in Natural Language Specifications for Review Improvements

2013-09  *Philip Geiger, Rüdiger Pryss, Marc Schickler, Manfred Reichert*
Engineering an Advanced Location-Based Augmented Reality Engine for Smart Mobile Devices

2014-01  *Andreas Lanz, Manfred Reichert*
Analysing the Impact of Process Change Operations on Time-Aware Processes

2014-02  *Andreas Steigmiller, Birte Glimm, and Thorsten Liebig*
Coupling Tableau Algorithms for the DL SROIQ with Completion-based Saturation Procedures

2014-03  *Thomas Geier, Felix Richter, Susanne Biundo*
Conditioned Belief Propagation Revisited: Extended Version

2014-04  *Hans A. Kestler, Matthias Schmid, Ludwig Lausser, Johann M. Kraus (eds)*

2014-05  *Andreas Lanz, Roberto Posenato, Carlo Combi, Manfred Reichert*
Simple Temporal Networks with Partially Shrinkable Uncertainty (Extended Version)

2014-06  *David Knuplesch, Manfred Reichert*
An Operational Semantics for the Extended Compliance Rule Graph Language