Simple Temporal Networks with Partially Shrinkable Uncertainty (Extended Version)

Andreas Lanz, Roberto Posenato, Carlo Combi, Manfred Reichert
Abstract: The Simple Temporal Network with Uncertainty (STNU) model focuses on the representation and evaluation of temporal constraints on time-point variables (timepoints), of which some (i.e., contingent timepoints) cannot be assigned (i.e., executed by the system), but only be observed. Moreover, a temporal constraint is expressed as an admissible range of delays between two timepoints. Regarding the STNU model, it is interesting to determine whether it is possible to execute all the timepoints under the control of the system, while still satisfying all given constraints, no matter when the contingent timepoints happen within the given time ranges (controllability check). Existing approaches assume that the original contingent time range cannot be modified during execution. In real world, however, the allowed time range may change within certain boundaries, but cannot be completely shrunk. To represent such possibility more properly, we propose Simple Temporal Network with Partially Shrinkable Uncertainty (STNPSU) as an extension of STNU. In particular, STNPSUs allow representing a contingent range in a way that can be shrunk during run time as long as shrinking does not go beyond a given threshold. We further show that STNPSUs allow representing STNUs as a special case, while maintaining the same efficiency for both controllability checks and execution.

1 INTRODUCTION

For more than a decade, the temporal constraint community has focused on the concept of controllability (Morris et al., 2001). Given a set of temporal constraints, of which each is expressed as an admissible range of delays between two time-point variables (timepoints for short), we distinguish two types of constraints: contingent and requirement constraints. The latter represent the standard temporal constraints, where both timepoints are under control of the system that “executes” the timepoints according to the assigned constraints (i.e., the system fixes the timepoints on the time line). This means that, during execution, the range admissible for some timepoints could be restricted by the system as it depends on the execution of already executed timepoints. In turn, contingent constraints are related to pairs of timepoints of which one (i.e., the contingent timepoint) is not under control of the system. Contingent timepoints are either given by the environment (Morris et al., 2001), i.e., they are related to uncontrollable, but expected, events, or by an external agent (i.e., human or software) who may decide autonomously when to execute the contingent timepoint. Considering this scenario, the attention of the temporal constraint community has moved from the problem of consistency, which consists of determining whether there exists an execution of all timepoints satisfying all given constraints (Dechter et al., 1991), to the problem of controllability; i.e., to determine whether it is possible to execute all timepoints under the control of the system, while satisfying all given constraints, no matter when the contingent timepoints happen within their given time ranges (Morris et al., 2001).

Most contributions from literature assume that the original time range of a contingent constraint cannot be modified during execution. Thus there is no difference between contingent timepoints given by the environment and the ones executed by external agents. In the real world, however, it is quite common that during execution the allowed time range may change, although it cannot be completely shrunk. To represent the behavior of external agents more properly, we may assume that an agent accepts certain reductions (i.e., modifications) of the initial execution range, as long as these do not go beyond a given threshold. In other words, there is an unshrinkable range of execution time the agent can always use. Further, this range is included into a larger one, the system may shrink during execution. The basic idea of our approach is to represent the fact that both the agent and the system
Temporal constraints are represented through arcs to the first activity has an allowed duration range, while visualized as rounded boxes and subsequent activities still allowing the therapists to autonomously choose be executed in a way satisfying all constraints, while One can easily observe that in this case the network can depicted in Fig. 1 (b) where the range is represented as applied when executing Stretching the second therapist may accept that the allowed duration range may be shrunk during execution, while guar-anteeing that the “core” range of the first activity. This scenario is is a directed weighted graph where a node repre-sents a time-point variable (timepoint), usually corre-sponding to the start or end of activities, and an edge repre-sents a lower and an upper bound constraint on the distance between the two timepoints it connects. Each STN is associated with a distance graph, derived from the upper and lower bound constraints, where a constraint between a pair of timepoints X and Y is represented as two edges: X ≤ Y, representing the constraint Y ≤ X + v, and X ≥ Y, which stands for Y ≥ X + u. An STN is denoted as consistent if it is possible to execute each node, i.e., to assign a real value to each timepoint such that all temporal constraints are satisfied. The consistency property can be verified by searching for negative loops in the graph. It is well known that consistency checking as well as determining the earliest/latest value of each timepoint can be done in polynomial time (Dechter et al., 1991). To properly represent events that cannot be executed, but only observed, (Morris et al., 2001) introduced Simple Temporal Networks with Uncertainty (STNUs). STNUs augment STNs with contingent timepoints representing timepoints whose value is decided by the environment. Each contingent timepoint has one incoming edge, called contingent link, which is labeled by a time range. Therefore, any contingent timepoint may assume a value from a bounded range, but the exact value is decided by the environment at run time. (Morris et al., 2001) provided a formal seman-
tics for the dynamic controllability, which is discussed in detail in Sect. 2.1. Moreover, (Morris et al., 2001) presented a pseudo-polynomial-time algorithm, called DC-checking algorithm, that determines whether a given STNU is dynamically controllable (DC). Further, (Morris and Muscettola, 2005) proposed the first polynomial DC-checking algorithm, which operates in $O(n^3)$ time, where $n$ is the number of timepoints. In this paper, we denote this algorithm as MM5. In turn, (Morris, 2006) presented an $O(n^4)$-time and (Morris, 2014) an $O(n^5)$-time DC-checking algorithm for STNUs, i.e., there are two interesting optimizations of the MM5 algorithm not further discussed in this paper.

(Tsamardinos et al., 2003) introduced the Conditional Temporal Problem (CTP) that augments an STN with observation timepoints. Thereby, each observation timepoint is associated with a boolean proposition. When executing the observation timepoint, the truth value of its proposition becomes known. Moreover, each timepoint of a CTP has a propositional label, which corresponds to a conjunction of (positive or negative) literals. A timepoint may only be executed in scenarios for which its propositional label evaluates to true. (Tsamardinos et al., 2003) proposed a formal semantics for the dynamic consistency property. The latter holds if there exists a strategy for executing the timepoints of the network in a way such that all constraints will be satisfied no matter how the observations turn out.

(Molfitt and Pollack, 2007) proposed three extensions of STNUs. One of them is the partial shrinkage in which observation events may trigger the reduction of contingent time ranges. In more detail, at design time it is possible to assign different time ranges and a special timepoint, called observation, to each contingent link; then, at run time, the respective observation timepoint chooses the time range to be considered for the contingent link before enabling the contingent timepoint. This way it becomes possible to customize the temporal range in which a contingent timepoint may occur. Further, it is shown how the concept of dynamic controllability can be expanded to capture such extensions. An analysis regarding the costs of respective DC-checking algorithm was not provided. The STNPSU model presented in this paper is similar in the sense that it allows one to choose the time range to be considered for a contingent link during run time. However, its execution semantics is quite different.

(Hunsberger et al., 2012) combined the features of STNUs and CTPs in order to create a Conditional Simple Temporal Network with Uncertainty (CSTNU). For CSTNUs, dynamic controllability was defined in a way generalizing the respective notions for STNUs and CTPs. Since existing DC-checking algorithms for STNUs and CTPs work differently, they cannot simply be combined to yield a DC-checking algorithm for CSTNUs. Instead, (Combi et al., 2013) proposed a novel algorithm (even if not complete) that incorporates new rules taking into account the propositional truth values generated by the observation timepoints.

(Lanz et al., 2013) showed how CSTNUs can be applied in the context of time-aware business processes in order to verify their controllability at both design and run time. Concerning temporal aspects of a business process, it is emphasized that activity durations usually represent worst case estimates, which are either based on the experience of a domain expert or extracted from process logs; further, the execution times of most activities can be shortened if required. Accordingly, one may assume that an activity has a flexible maximum duration $\text{MaxD}$ that may be restricted up to a contingent minimum and maximum duration range $[\text{MinD}, \text{MaxD}]$. In other words, they proposed and analyzed a mapping of time-aware business processes to CSTNU in which activity durations are expressed in terms of shrinkable time intervals $[\text{MinD}, \text{MaxD}]$.

2.1 Dynamic Controllability of STNUs

As proposed by (Morris et al., 2001), an STNU is a set of time-point variables (timepoints) and temporal constraints together with a set of contingent links. Each contingent link has the form $(A, x, y, C)$, where $A$ and $C$ are timepoints and $0 < x < y < \infty$ holds. $A$ is called the activation timepoint and $C$ the contingent timepoint. Once $A$ is executed, $C$ is guaranteed to be executed such that $C - A \in [x, y]$ holds. However, the particular time at which $C$ is executed is uncontrollable since it is decided by the environment; i.e., it can be only observed when it happens.

Let $S = (T, C, L)$ be an STNU, with $T$ being a set of timepoints, $C$ a set of constraints, and $L$ a set of contingent links. The corresponding graph for $S$ has the form $(T, E, \mathcal{E}_{x}, \mathcal{E}_{0})$. Thereby, each timepoint in $T$ serves as a node in the graph; $E$ is a set of ordinary edges; $\mathcal{E}_{x}$ is a set of lower-case and $\mathcal{E}_{0}$ a set of upper-case edges (Morris and Muscettola, 2005).

- Each ordinary edge has the form $X \xrightarrow{Y} Y$, representing the constraint $Y - X \leq v$.
- Each lower-case edge has the form $A \xrightarrow{C} C$, representing the possibility that the contingent duration, $C - A$, might take on its minimum value $x$.
- Each upper-case edge $C \xrightarrow{A}$ represents the possibility that the contingent duration, $C - A$, might take on its maximum value $y$.

An STNU is dynamically controllable if there exists a strategy for executing its timepoints, in a way...
guaranteeing that all constraints in the network can be satisfied, no matter how the durations of the contingent links actually turn out. The strategy is dynamic since its execution decisions can react to observations of contingent links that have already been completed, while excluding those not completed yet.

This section presents preliminary notions and introduces the dynamic controllability of an STNU as defined in (Morris et al., 2001) and subsequently fixed in (Hunsberger, 2009).

For an STNU, a situation specifies fixed durations for all contingent links.

**Definition 1** (Situations). Let $S$ be an STNU comprising $k$ contingent links, $(A_1,x_1,y_1,C_1), \ldots, (A_k,x_k,y_k,C_k)$, with corresponding duration ranges $[x_1,y_1] \times \ldots \times [x_k,y_k]$. Then: $\Omega_S = [x_1,y_1] \times \ldots \times [x_k,y_k]$ is called the space of situations for $S$. Any $\omega = (d_1, \ldots, d_k) \in \Omega_S$ is called a situation. Where possible, we may write $\Omega$ instead of $\Omega_S$.

The concept of schedule formalizes the execution of timepoints.

**Definition 2** (Schedule). A schedule for an STNU is a mapping $\psi : T \rightarrow \mathbb{R}$ that assigns a real number to each timepoint in $T$.

Given a situation $\omega$ for an STNU, the replacement of its contingent links by the durations specified in $\omega$ determines a projection of the STNU onto situation $\omega$.

**Definition 3** (Situation Projection for an STNU). Suppose $S = (T,C,L)$ is an STNU and $\omega = (d_1, \ldots, d_k)$ a situation. The projection of $S$ onto $\omega$—denoted as $sitPrj(S,\omega)$—is the STN $(T,C')$ with:

$$C' = C \cup \{(d_i \leq C_i - A_i \leq d_i) \mid 1 \leq i \leq k\}$$

Given an STNU, multiple schedules may exist. We are interested in finding a strategy that determines schedules that satisfy all constraints in any situation.

**Definition 4** (Execution Strategy for an STNU). Let $S = (T,C,L)$ be an STNU. An execution strategy for $S$ is a mapping $\sigma : \Omega \rightarrow \psi$ such that for each situation $\omega \in \Omega$, $\sigma(\omega)$ is a (complete) schedule for the timepoints in $T$. Furthermore, if for each situation $\omega$ schedule $\sigma(\omega)$ is a solution for the situation projection $sitPrj(S,\omega)$, then $\sigma$ is called viable. In any case, the execution time of timepoint $x$ in schedule $\sigma(\omega)$ is denoted as $[\sigma(\omega)]_x$.

A situation history for an STNU specifies the durations of all contingent links that have finished their execution prior to a time $t$ in schedule $\sigma(\omega)$.

**Definition 5** (Situation History for an STNU). Let $S = (T,C,L)$ be any STNU, $\sigma$ any execution strategy for $S$, $\omega$ any situation, and $t$ any real number. The history of $t$ in situation $\omega$ for strategy $\sigma$—denoted as $sitHst(t,\omega,\sigma)$—is defined as follows:

$$sitHst(t,\omega,\sigma) = \{(A,C,[\sigma(\omega)]_C - [\sigma(\omega)]_A) \mid \exists x, y \text{ such that } (A,x,y,C) \in L \land [\sigma(\omega)]_C - [\sigma(\omega)]_A < t\}$$

**Definition 6** (Dynamic Execution Strategy for an STNU). An execution strategy $\sigma$ for an STNU is called **dynamic** if for any situations, $\omega_1$ and $\omega_2$, and any non-contingent timepoint $x$, it holds:

$$sitHst([\sigma(\omega_1)]_x,\omega_1,\sigma) = sitHst([\sigma(\omega_2)]_x,\omega_2,\sigma) \Rightarrow [\sigma(\omega_1)]_x = [\sigma(\omega_2)]_x$$

**Definition 7** (Dynamic Controllability of an STNU). An STNU $S$ is called dynamically controllable (DC) if there exists an execution strategy for $S$ that is both viable and dynamic.

Table 1: Edge-generation rules of the MM5 algorithm. Dashed edges are the generated ones.

<table>
<thead>
<tr>
<th>No Case:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$Q \quad \sigma$ $\quad S \quad v \quad \nu \quad T$</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Upper Case:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$Q \quad R \quad \sigma$ $\quad S \quad v \quad \nu \quad T$</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Lower Case:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$Q \quad \sigma$ $\quad S \quad v \quad \nu \quad T$</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cross Case:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$Q \quad R \quad \sigma$ $\quad S \quad v \quad \nu \quad T$</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Label Removal:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$Q \quad \sigma$ $\quad S \quad v \quad \nu \quad T$</td>
<td></td>
</tr>
</tbody>
</table>

In order to determine whether an STNU is dynamically controllable, Morris and Muscettola (2005) proposed a polynomial-time checking algorithm, MM5, which works by recursively generating new edges in the STNU graph according to the rules from Table 1 and checking whether newly added edges determine negative loops in the graph. For each rule, existing edges are represented as solid arrows and newly ones as dashed arrows. Each of the first four rules takes two existing edges as input and generates a single edge as output. Finally, notation $R \not\equiv S$ expresses that $R$ and $S$ must be distinct time-point variables, and does not represent a constraint on the values of those variables.

We observe that the edge-generation rules from Table 1 only generate ordinary or upper-case edges.
as well as non-contingent timepoints. As observed in (Hunsberger, 2009), the original defini-

Upper-case edges generated by respective rules represent conditional constraints, called waits (Morris et al., 2001). In particular, an upper-case edge \( B^C \rightarrow A \) represents the following constraint: as long as contingent timepoint \( C \) remains unexecuted, timepoint \( B \) must wait at least \( v \) units after the execution of \( A \), the activation timepoint for \( C \).

Procedure MM5-DC-Check shows the pseudocode of the MM5 DC-checking algorithm. Its time complexity is \( O(n^2) \) (Morris and Muscettola, 2005).

2.2 Alternative Characterization of an Execution Strategy

As observed in (Hunsberger, 2009), the original definition of dynamic execution strategy (DES) obscures the real-time features of typical execution scenarios and the kinds of execution decisions an execution system may make. Therefore, (Hunsberger, 2009) proposed an alternative characterization of a DES to not only represent the conditions under which a system must make real-time execution decisions, but also the outcomes of those decisions. Two kinds of real-time execution decisions (RTEDs) are defined: WAIT and \((\tau, \chi)\) decisions, which can be described as: "Wait until some contingent duration completes" or "If nothing happens before \( \tau \), then execute the (executable) timepoints in \( \chi \)." The outcome of a RTED depends on the situation and is represented by a partial schedule that specifies the execution of one or more additional timepoints. The outcome of a WAIT decision solely involves the execution of contingent timepoints, whereas the outcome of a \((\tau, \chi)\) decision may involve the execution of contingent as well as non-contingent timepoints. An RTED-based strategy is defined as a mapping from partial schedules to real-time execution decisions. (Hunsberger, 2009) proved that RTED-based strategies correspond one-to-one to DESs.

In more detail, given an STNU and a partial schedule \( \psi : T \rightarrow \mathbb{R} \) (i.e., the domain of \( \psi \) may be a subset of \( T \)), we denote by \( \mu(\psi) = \max \{ \psi(t) \mid t \in \text{Dom}(\psi) \} \) the maximum execution time of timepoints appearing in \( \psi \), by \( U(\psi) = \{ x \mid x \notin \text{Dom}(\psi) \} \) the set of unexecuted timepoints in \( \psi \), by \( U^0(\psi) \subseteq U(\psi) \) the set of non-contingent unexecuted timepoints, by \( U^c(\psi) \subseteq U(\psi) \) the set of contingent unexecuted timepoints, and by \( U^a(\psi) \subseteq U^c(\psi) \) the set of contingent activated unexecuted timepoints, respectively.

Let \( \psi \) be a partial schedule for an STNU \( S \) and \( \omega = (\omega_1, \ldots, \omega_n) \) a situation for \( S \). \( \psi \) respects \( \omega \) if for each contingent link \((A_i, x, y, C_i)\) one of the following conditions holds: (1) neither \( A_i \) nor \( C_i \) appear in \( \psi \); (2) only \( A_i \) appears in \( \psi \), and \( \psi(A_i) + \omega_i > \mu(\psi) \); or (3) both \( A_i \) and \( C_i \) appear in \( \psi \), and \( \psi(A_i) + \omega_i = \psi(C_i) \). \( \psi \) is called respectful if it respects at least one situation. If \( \psi \) is both respectful and partial, it is called a respectful, partial schedule (RPS). A strategy \( \sigma \) is respectful if for each \( \omega \), \( \sigma(\omega) \) respects \( \omega \).

Let us recall the definition of WAIT and \((\tau, \chi)\) decisions.

**WAIT Decision.** Let \( \psi \) be some RPS for \( S \) such that \( U^a(\psi) \) is non-empty. Then WAIT is an admissible RTED.

**Outcome of a WAIT Decision.** If \( U^a(\psi) \neq \emptyset \) and \( \omega \) is a situation respected by \( \psi \), then the time at which the next contingent timepoint will execute is defined as \( tnc(\psi, \omega) = \min \{ \psi(A_i) + \omega_i \mid C_i \in U^a(\psi) \} \). With \( \chi^\omega(\psi, \omega) = \{ C_i \in U^a(\psi) \mid \psi(A_i) + \omega_i = tnc(\psi, \omega) \} \), we denote the non-empty set of contingent timepoints that will be executed at time \( tnc(\psi, \omega) \). Then, the outcome of the WAIT decision for \( \psi \) in situation \( \omega \) is defined to be the execution of contingent timepoints at time \( tnc(\psi, \omega) \) if \( \tau > \mu(\psi) \) and \( \chi \) is a non-empty subset of \( U^a(\psi) \), then \((\tau, \chi)\) is an admissible RTED for \( \psi \).

**Outcome of a \((\tau, \chi)\) Decision.** Let \( \alpha \) be a situation respected by \( \psi \). The outcome of a \((\tau, \chi)\) decision depends on the relationship between \( tnc(\psi, \omega) \) and \( \tau \). For the sake of simplicity, let \( \tau = tnc(\psi, \omega) \) and \( \chi^\omega = \chi(\psi, \omega) \) if \( U^a(\psi) \neq \emptyset \); otherwise, let \( \tau = \infty \).

If \( \tau < tnc(\psi, \omega) \), the outcome solely involves the execution of the contingent timepoints in \( \chi^\omega \). In turn, if \( \tau < tnc(\psi, \omega) \), the outcome solely involves the execution of the non-contingent timepoints in \( \chi \). Finally, for \( \tau = tnc(\psi, \omega) \), the outcome involves the execution of the timepoints in both \( \chi^\omega \) and \( \chi \).

(Hunsberger, 2009) proved that the original dynamic execution strategy can be described in terms of RTEDs as shown in procedure RTEDExecutionStrategy. Thereby, function RTEDExecutionDecision is used to determine the next RTED. For the sake of brevity, the RTED WAIT is represented as \((\tau, \chi)\) decision with \( \tau := \infty \) and \( \chi := \emptyset \) in the given context.

Starting with a partial schedule \( \psi = \{(Z, 0)\} \), which only fixes the initial timepoint \( Z \), procedure RTEDExecutionStrategy iteratively determines an RTED \( \delta(\psi) \), considering two possibilities (cf. function RTEDExecutionDecision). If all executable time-
After determining the RTED with the values of $\tau$ (govant waits.

Regarding an STNU, the execution of a contingent of

dure floor showed that a conflict between

the constraint that $x$ floor pairs, shortest-path algorithm.

$\delta$ points have already been executed,


$\psi$RTEDExecutionStrategy($S$)

Function $\text{RTEDExecutionStrategy}(S, \psi)$

Input: $S$: STNU. $\psi$: partial schedule

Output: $\delta(\psi)$: real-time execution decision

if ($U^i(\psi) = \emptyset$) then

$$\delta(\psi) := (t^i, \chi), \text{ where } t^i := \infty \text{ and } \chi := \emptyset;$$ // WAIT RTED

else

$$\delta(\psi) := \max \{\delta(\psi), (t, \chi)\} \text{ RTED}$$

foreach ($x \in U^i(\psi)$) do

$$m(x), M(x) = \text{current-time window for } x;$$

$W(x) := -\infty;$

$W(x) := \max\{W(x), \psi(A_i) + w_i\};$

go(x) := $\min\{floor(x), M(x)\};$

$\delta(\psi) := (t^i, \chi), \text{ where } t^i := \max\{go(x) \mid x \in U^i(\psi)\}$

and $\chi := \{x \in U^i(\psi) \mid t^i = go(x)\};$

return $\delta(\psi)$;

Procedure $\text{RTEDExecutionStrategy}(S)$

Input: $S$: STNU.

$\psi = \{(Z, 0)\}; \text{ // initial partial schedule}$

while ($U(\psi) \neq \emptyset$) do

$$\delta(\psi) = \text{RTEDExecutionDecision}(S, \psi);$$

if (nothing happens before time $t^i$) then

Execute the timepoints in $t^i$;

else

Observe the contingent timepoints executed at some $t^i < t^i$;

Update $\psi$ to include the executed events;

Update $S$ to include the corresponding constraints;

RTEDExecutionStrategy

points have already been executed, $\delta(\psi) = (\infty, \emptyset)$ holds (i.e., RTED WAIT); otherwise, $\delta(\psi) = (t^i, \chi)$ with the values of $t^i$ and $\chi$ being computed by considering all unexecuted timepoints and using an allpairs, shortest-path algorithm. $floor(x)$ corresponds to the earliest timepoint, timepoint $x$ may be executed without violating its lower bound $m(x)$ or any of its relevant waits. $go(x)$ is the same, except that it enforces the constraint that $x$ does not violate its upper bound $M(x)$. It is noteworthy that Morris and Muscettola showed that a conflict between $floor(x)$ and $M(x)$ is not possible for an STNU accepted by their algorithm. After determining the RTED $\delta(\psi) = (t^i, \chi)$, procedure waits for the outcome of $\delta(\psi)$ and then updates $\psi$ and $S$ accordingly. If there are still unexecuted timepoints, the procedure iterates, otherwise it terminates.

3 GUARDED TEMPORAL CONSTRAINTS

Regarding an STNU, the execution of a contingent timepoint can be thought of as being completely out of the control of the system that executes the network. Typically, a system activates a contingent link $(A, x, y, C)$ by executing its activation timepoint $A$. Afterwards, the execution of $C$ is out of the system’s control. However, the contingent timepoint $C$ is guaranteed to execute such that $C = A \in [x, y]$ holds.

As motivated in Sect. 1, for real-world problems this is often too strict. In many cases, the system may exercise some control over the execution of the contingent timepoint. As example consider a case where, at an activation timepoint, the system transfers control to an external agent. The agent is then responsible for executing the corresponding contingent timepoint. In turn, the system waits for the agent to complete its task (i.e., to execute the contingent timepoint). When transferring the control to the agent, the system may inform the agent about the temporal constraints to be met. The agent then adapts its plan in order to comply with the additional constraints. At the same time, the system must guarantee that it is able to meet the commitment made, i.e., it needs to ensure that it can deal with any decision the agent makes for executing timepoint $C$ based on the given constraints.

In many cases, the agent responsible for executing timepoint $C$ cannot completely control the execution of $C$ either (e.g., in case the agent is executing a network itself). Particularly, he might only be able to provide a preferred duration range $[x, y]$ as well as bounds $x_{\text{max}}$ and $y_{\text{min}}$ to which $x$ may be increased or $y$ may be decreased (i.e., $x \leq x_{\text{max}}$ and $y \geq y_{\text{min}}$). In turn, the system executing the network must ensure that, when executing timepoint $A$ (i.e., when activating the constraint between $A$ and $C$), the agent responsible for executing timepoint $C$ has at least $y_{\text{min}}$ time units and is not required to take more than $x_{\text{max}}$ time units to execute $C$. We denote $x_{\text{max}}$ ($y_{\text{min}}$) as the guard of $x$ ($y$)

Note that this example addresses a common scenario, i.e., to transfer execution control at run time to another agent, which is responsible for executing a complex task (e.g., another network).

The need to model constraints of this type requires an extension of the STNU formalism, we denote as Simple Temporal Network with Partially Shrinkable Uncertainty (STNPSU). In particular, STNPSU extends contingent links of STNU to guarded links.

Definition 8 (STNPSU). A Simple Temporal Network with Partially Shrinkable Uncertainty (STNPSU) is a triple $(T, C, G)$ where:

- $T$ is a set of timepoints;
- $C$ is a set of requirement constraints $X \xrightarrow{[u,v]} Y$ (i.e., STN constraints); and
- $G$ is a set of guarded links each having the form $(A, [x, x_{\text{max}}], [y_{\text{min}}, y], C)$ where $A$ and $C$ are timepoints, and $0 \leq x \leq y < \infty, x \leq x_{\text{max}}, 0 \leq y_{\text{min}} \leq y$.
- If $(A_1, [x_1, x_{1}\text{max}]), [y_{1}\text{min}, y_1], C_1)$ as well as $(A_2,$
When executing the activation timepoint of the remaining network. In the given case, for example, the current outer bounds of the guarded link must always allow for at least one value within its guarded core \([x_{min}, x_{max}]\) (i.e., \([10, 20]\)).

During execution, when activating a guarded link of Type 1 or 2 (i.e., when executing its activation timepoint), the current outer bounds of the guarded link are fixed. This is to ensure that the outer bounds of the guarded link cannot be modified while it is active. Therefore, the current outer bounds of the guarded link are set to be strict. For example, when executing timepoint \(A\), the Type 2 guarded link \((A, [15, 20], [20, 20], C)\) is replaced by a strict guarded link \((A, [15, 15], [20, 20], C)\). The latter is equivalent to a contingent link \((A, 15, 20, C)\) of STNU and ensures that the agent responsible for executing timepoint \(C\) may now choose any time in range \([15, 20]\) to execute timepoint \(C\).

### 3.1 Dynamic Controllability of STNPSU

This section presents preliminary definitions of basic concepts required for the definition of dynamic controllability of a STNPSU.

The set of core situations specifies the contingent core of all guarded links of Type 1 (partially contingent guarded links), while the set of core settings specifies the guarded core of all guarded links of Type 2 (partially shrinkable guarded links).

**Definition 9 (Core Situations and Core Settings).** Suppose \(S = (T, C, G)\) is an STNPSU. Let \(G' = \{g \in G|g = (A, [x_{max}], [y_{min}, y_{max}], C) \cap x_{max} < y_{min}\}\) be the set of guarded links for which the guard \(x_{max}\) of the lower bound is lower than the guard \(y_{min}\) of the upper bound (i.e., Type 1). Further, let \(G'' = G \setminus G'\) be the set of guarded links for which \(y_{min} \leq x_{max}\) holds (i.e., Type 2).

If \(G''\) contains \(k\) guarded links, \((A_1, [x_{1_{min}}, y_{1}], C_1), \ldots, (A_k, [x_{k_{min}}, y_{k_{max}}, y_{k}], C_k)\), then \(\Omega_S = [x_{1_{max}}, y_{1_{max}}] \times \ldots \times [x_{k_{max}}, y_{k_{max}}]\) is called the space of core situations for \(S\). Any \(\omega S = (d_1, \ldots, d_k) \in \Omega_S\) is called a core situation.

Further, if \(G''\) contains \(m\) guarded links, \((A_1, [x_{1_{min}}, y_{1}, C_1], \ldots, (A_m, [x_{m_{min}}, y_{m_{max}}], y_{m_{min}}, y_{m}], C_m)\), then \(\Omega_S = [y_{1_{min}}, x_{1_{max}}] \times \ldots \times [y_{m_{min}}, x_{m_{max}}]\)
We denote the respective STNU as the space of core settings for $S$.

Given the space of core situations $\Omega'$ and the space of core settings $\Omega''$ of an STNPSU, a projection of the STNPSU onto an STNU can be obtained as follows: First, each guarded link in $G''$ is replaced by a contingent link for the range specified in $\Omega'$. Second, each guarded link in $G''$ is replaced by a requirement constraint for the range in $\Omega'$.

**Definition 10 (Core STNU of an STNPSU).** Let $S = (T, C, G)$ be an STNPSU.

Then: The projection of $S$ onto its space of core situations $\Omega'$ and its space of core settings $\Omega''$—denoted as $\text{stnuPrj}(S, \Omega', \Omega'')$—corresponds to an STNU $(T', C', L')$ with:

$$
C' = C \cup \{ (y_{\min}^i \leq G - A_i \leq y_{\max}^i) \mid 1 \leq i \leq m, \Omega' = [y_{\min}^i, y_{\max}^i] \times \ldots \times [y_{\min}^m, y_{\max}^m] \}
$$

$$
L'' = \{ (A_i, x_{\min}^i, y_{\min}^i, C_i) \mid 1 \leq i \leq k, \Omega'' = [x_{\min}, y_{\min}] \times \ldots \times [x_{\max}, y_{\max}] \}
$$

We denote the respective STNU as the core STNU of STNPSU $S$.

Finally, this leads us to the dynamic controllability of an STNPSU. We provide a formalization of the dynamic controllability of an STNPSU based on the dynamic controllability of an STNU. We choose this approach since the formalization of dynamic controllability of STNU is robust and verified in literature.

**Theorem 1 (Dynamic Controllability of STNPSU).** An STNPSU $S = (T, C, G)$ is dynamically controllable (DC), if the core STNU that results from the STNU Projection $\text{stnuPrj}(S, \Omega', \Omega'')$ of the STNPSU is dynamically controllable.

**Proof.** It is a matter of definitions to show that, if the core STNU is DC (cf. Sect. 2), the corresponding STNPSU is DC as well: each schedule being a solution of the core STNU is also a solution of the STNPSU. Indeed, it is always possible to restrict the STNPSU to its core situations. Thus, for each core situation of the STNPSU, a dynamic execution strategy (DES), which is a viable DES for the STNU, is also a viable DES for the STNPSU regarding its core situations. Hence, if the core STNU is DC, the STNPSU will be DC as well.

If the core STNU is not DC (i.e., no viable DES exists), at least one core situation $\alpha''$ of the STNPSU exists for which no DES exists within the core settings. Hence, for core situation $\alpha''$, one of the partially shrinkable guarded links must be restricted beyond its guards to find a DES which returns a solution. As this is not possible, the STNPSU is not DC either.

### 3.2 DC-Checking for Guarded Constraints

This section shows how the dynamic controllability of an STNPSU may be checked without need to restrict the respective STNPSU to its core STNU. First, we emphasize the close relationship between dynamic controllability of STNU and the one of STNPSU (cf. Theorem 1). In turn, this fosters the following graph-based representation of an STNPSU, which is similar to the one of an STNU.

**Definition 11 (Graph of a STNPSU).** The graph for an STNPSU $S$ has the form $(T', E', \varepsilon, E_u)$, where each timepoint in $T$ corresponds to a node in the graph; $\varepsilon$ is a set of ordinary edges, $E_u$ is a set of lower-case edges, and $E_u$ is a set of upper-case edges:

- Each requirement constraint $X \xrightarrow{[u, x]} Y$ is represented by two ordinary edges $X \xrightarrow{\alpha u} Y$ and $Y \xrightarrow{\alpha u} X$.
- Each guarded link $(A, [x, x_{\max}], [y_{\min}, y], C)$ is represented by:
  - two ordinary edges $A \xrightarrow{\alpha u} C$ and $C \xrightarrow{\alpha u} A$,
  - one lower-case edge $A \xrightarrow{\alpha_{\min}} C$, and
  - one upper-case edge $C \xrightarrow{\alpha_{\max}} A$.

As example of this graph-based representation of an STNPSU, consider the graph depicted in Fig. 2. It shows the STNPSU corresponding to the activity sequences from Fig. 1(b).

![Figure 2: STNPSU corresponding to the activity sequences from Fig. 1(b)](image)
to problem for checking dynamic controllability, but it is replacing it with a requirement edge. This poses no
be applied to the upper-case edge between
In case of a partially shrinkable guarded link (Type 2),
comprising two triangles (i.e., the No Case rule
First, consider the single guarded link depicted in
the value of a lower-case edge
always is lower than the negative value of the
upper-case edge
corresponds to the negative value of ordinary edge
pointing in the opposite direction (i.e.,
lower bound
Due to some edge-generation rule, a pair of require-
constraints need to be kept. The other ones are redun-
(a) Original values of a Guarded Link and a Requirement
change does not influence the applicability of the rule
to an STNU. Regarding an STNU, for R ⊢ S it holds
that v < −x (i.e., x = x_{\text{max}}; cf. Table 1), i.e., for an
STNU, the rule will never be applied if R ⊢ S holds.
Consider the guarded link depicted in Fig. 3(b).
Due to some edge-generation rule, a pair of require-
ments constraints Q →\nu → T and T →\nu Q is added next to the
guarded link. In this case only the most restrictive
constraints need to be kept. The other ones are redun-
and may be removed. If controllable, this scenario
results in one of the four cases depicted in Fig. 4(b–
(c) Note that if either v < u, u > x_{\text{max}} or v < y_{\text{min}}
holds, the scenario will be not controllable (i.e., there
exists a negative loop). If u < y_{\text{min}} and v > x_{\text{max}} holds,
the outer bounds x and y of the guarded link may have
to be restricted to u and v, respectively, as shown in
Fig. 4(b). Note that for partially contingent guarded
links (i.e., Type 1), this is the only controllable case.
In turn, if u ≥ y_{\text{min}} and v ≤ x_{\text{max}} holds (cf. Fig. 4(c) and (e)),
guard y_{\text{min}} of the upper bound y is no longer required as
can be already ensured by the new lower bound u
of the guarded link, i.e., the upper-case edge is
redundant and may be removed. Likewise, if v ≤ x_{\text{max}}
holds the lower-case edge (i.e., the guard x_{\text{max}} for the
lower bound x) is redundant and may be removed. To
preserve the difference between a guarded link and
a requirement/wait constraint in the STNPSU graph,
instead of removing the guards, we adapt their value
according to the restriction given by the requirement
defs (cf. Fig. 4(c)). Particularly, instead of
removing the lower-case or upper-case edge, we allow
their values to be decreased (i.e., the negative value
of the upper-case edge to be increased). Depending
Figure 5: Combining a Guarded Link with a Wait Constraint

(a) Guarded Link followed by Wait Constraint
(b) Guarded Link in parallel with Wait Constraint

Figure 6: Combining a Guarded Link with a derived Wait Constraint

(a) Original values of Guarded Link and Wait Constraint.
(b) If $x < v_w \leq x^{max} \land u < y^{min}$.

3.3 Executing STNPSUs

This section shows how an STNSU may be executed by means of an appropriate extension of procedure $RTED\text{ExecutionStrategy}$ (cf. Sect. 2.2).

Consider procedure $ExRTED\text{ExecutionStrategy}$. The first part of the procedure executes the same actions as procedure $RTED\text{ExecutionStrategy}$ (cf. Sect. 2.2). The second part activates all guarded links $(A_i, [x, x^{max}], [y^{min}, y], C_i)$ whose activation timepoint $A_i$ has just been executed. The guarded link semantics requires to allow each of them, once it is activated, to use any possible value in the range defined by the current outer bounds, i.e., $[x, y]$. By construction and due to the fact that the network is DC, for a Type 1 guarded link the possibility of using any possible value in the range is guaranteed only for the core range $[x^{max}, y^{min}]$.

One of the most interesting cases is depicted in Fig. 6(b). Due to an edge generation rule, a wait constraint $T \leftarrow v_w$ is added next to the guarded link. For the case $x < v_w \leq x^{max} \land u < y^{max}$ the result is depicted in Fig. 6(b). Consider the wait constraint added next to the upper-case edge. Note that this is not possible for an STNU. Regarding an STNU, either $v_w > x^{max} = x$ (i.e., the STNU is not controllable) or $v_w \leq x$ (i.e., the wait constraint is redundant) holds. For an STNSU, Fig. 6(b) can be interpreted as follows: For Type 1 constraints (i.e., $v_w < x^{max} < y^{min}$) $R \leftarrow v_w$ is not relevant as $T \leftarrow y^{min}$ is more restrictive, i.e., we are only interested in dynamic controllability of the core STNU. Otherwise, $R \leftarrow v_w$ possibly represents a standard wait constraint for the core settings in $[y^{min}, x^{max}]$. In turn, when executing the STNSU the wait constraint must be observed. Particularly, if the guarded link is activated prior to the execution of timepoint $R$, $v_w$ must be used as lower bound when executing the guarded link. Otherwise, $\min[-x,-u]$ is used as lower bound.

**Theorem 2.** Suppose $S$ is a dynamically controllable STNPSU, $\psi$ is a respectful, partial schedule, and $(A, [x, x^{max}], [y^{min}, y], C)$ is a guarded link of $S$. Let us assume that $A$ has just been executed and that the
Procedure ExRTEDEExecutionStrategy(\(\mathcal{S}\))
\[
\text{Input: } \mathcal{S} \text{: STNPSU.}
\]
\[
\psi = \{\{0, \mathcal{Z}\}\}; \quad \text{// initial partial schedule}
\]
\[
\text{while } (\mathcal{U}(\psi) \neq \emptyset) \text{ do}
\]
\[
(\psi', \chi) = \text{RTExecutionDecision}(\mathcal{S}, \psi);
\]
\[
\text{if (nothing happens before time } \chi') \text{ then}
\]
\[
\text{// execute the time-points in } \chi';
\]
\[
\text{else}
\]
\[
\text{observe the contingent timepoints executed at some } t_c < \chi';
\]
\[
\chi = \text{set of executed contingent timepoints};
\]
\[
\text{Update } \psi \text{ to include the execution events in } \chi;\]
\[
\text{Update } \mathcal{S} \text{ to include the corresponding constraints;}
\]
\[
\text{// Prepare the guarded link for execution}
\]
\[
\text{// Determine its maximum controllable range}
\]
\[
\text{range}(A, C_i) = \min\{v - u \mid a \in \mathcal{U}(\psi)! \land
\]}
\[
a -\rightarrow C_i \wedge v \geq 0 \land C_i \rightarrow_{a'} C_i'\};
\]
\[
\text{// Update its bounds to observe max. controll. range}
\]
\[
y' = \min\{y, \max\{y_{min}, x'+\text{range}(A, C_i)\}\}.
\]
\[
\text{if } x' \text{ is updated only if the update } y' \text{ is not sufficient}
\]
\[
x' = \max\{x', y' - \text{range}(A, C_i)\};
\]
\[
\text{// Update } \mathcal{S} \text{ to include the modified outer bounds of the guarded link;}
\]
\[
\text{until neither } x' \text{ nor } y' \text{ is modified;}
\]
\[
\text{// Consider the max possible wait constraint for } C_i
\]
\[
W(A, C_i) := -\infty;
\]
\[
\text{// Transform the guarded link to the strict guarded link}
\]
\[
\text{floor}(A, C_i) := \max\{x', W(A, C_i)\};
\]
\[
x' := \max\{\text{floor}(A, C_i), y'\};
\]
\[
\text{Update } \mathcal{S} \text{ to include the new constraint;}
\]
\]

outer bounds x and y of the guarded link AC have been updated as described in lines 13-25 of procedure ExRTEDEExecutionStrategy. Then: The new values x' and y' satisfy x' \leq x_{max} and y_{min} \leq y' and the STNPSU S' resulting after the transformation of the guarded link (A, x', y', C) into the strict one (A, [x', x_{max}], [y_{min}, y'], C) is dynamically controllable as well.

Sketch of the proof. Let us assume that, before the execution of lines 13–25, the guarded link AC is given as (A, [x, x_{max}], [y_{min}, y'], C). Instructions of lines 13–25 update the outer bounds [x, y] to [x', y']. Let us assume that the guarded link is of Type 1\(^\dagger\). Since the network is DC before line 13 and the core range [x_{max}, y_{min}] is a contingent range, the made update in lines 13–25 cannot reduce [x, y] to [x', y'] such that x' > x_{max} or y' < y_{min} holds. Hence, the updated range [x', y'] contains (possibly in a weak way) the core range of (A, [x_{max}, y_{min}], C). Thus, there are 4 possible cases:

1. x' = x_{max} \land y' = y_{min},
2. x' = x_{max} \land y' > y_{min},
3. x' < x_{max} \wedge y' = y_{min},
4. x' < x_{max} \wedge y' > y_{min}.

In case (1) the guarded link is already strict and, hence, it is not changed by the procedure. Case (4) is a combination of cases (2) and (3).

Hence, let us consider case (2) and (3). In case (2), by contradiction, suppose that the changing of the guarded link (A, [x_{max}, x_{max}], [y_{min}, y'], C) into the strict one (A, [x_{max}, x_{max}], [y_{min}, y'], C) makes the remaining network not dynamically controllable. This means that there exists at least one negative loop involving timepoints A, C, and some B, where B has not yet executed (i.e., it has to occur after A), but has to be executed before C. All other timepoints, i.e., unexecuted ones that have to be executed after C, cannot contribute to form a negative loop since each of them—by definition of controllability—must have at least one possible execution time for each possible execution time of C in the range \([\psi(A) + x_{max}, \psi(A) + y']\).

Now, instead of considering the distance graph and negative loops in it, let us reason in term of ranges and their spans\(^2\). Given the dynamic controllability of the network before the transformation, it is a fact that B has at least one possible execution time for each possible execution time of C in the range \([\psi(A) + x_{max}, \psi(A) + y_{min}]\), i.e., with such range there is no negative loop involving A, B, and C. Equivalently, the span of the constraint between B and C is greater or equal to the span \(y_{min} - x_{max} + 1\) of the contingent core of the guarded link between A and C. Therefore, a negative loop can only emerge when the new bound y' is considered, or, equivalent, when the span of the constraint between B and C is less than the span \(y' - x_{max} + 1\) of the outer bounds of the guarded link between A and C.

However, when preparing the guarded link for execution, y' is determined such that the span of range \([x_{max}, y']\) of the link between A and C is lower than or equal to the span of the constraint between B and C. Thus, there cannot exist any negative loop involving A, B, and C.

Case (3) can be shown in a similar way taking also into account any possible wait constraint that may increase the value of x'.

\(^\dagger\)The case of a guarded link of Type 2 can be discussed in a similar way.

\(^2\)The span of a range \([a, b]\) is \(b - a + 1\).
4 ON THE EXPRESSIVENESS OF GUARDED CONSTRAINTS

This section informally discusses the expressiveness of STNPSUs. Further, we show that most guarded links cannot be represented in STNUs by other solutions.

Consider again the physiotherapy session scenario from Fig. [3](b). Activity Stretching has a duration range of [10, 40], which may be shrunk to a core duration range of [15, 20] during run time according to the actual duration of activity Biking. Fig. [2](b) depicts the temporal aspects of the session in terms of an STNPSU: each of the two activities is represented through a pair of timepoints, of which one represents the starting instant of the activity and the other one the ending instant. The allowed duration of activity Stretching is represented as a simple form of a guarded one and that of activity Biking as a strict guarded one and that of activity Biking as a strict guarded one and that of activity Biking as a strict guarded one. The main contribution of this paper is to present an extension of STNU that allows for the definition and efficient management of a novel kind of constraints, i.e., guarded links. A guarded link represents an admissible range of delays between two timepoints, where each bound of the range can be shrunk during run time, but not beyond a given threshold. A guarded link constitutes a generalization of the two kinds of STNU constraints, i.e., requirement and contingent constraints, in the sense that a contingent link may be represented as a simple form of a guarded one and that a guarded link may represent a requirement constraint. An STNU where it is possible to define guarded links is denoted as Simple Temporal Network with Partially Shrinking Uncertainty (STNPSU). In particular, the dynamic-controllability check and the execution of a STNPSU can be done in polynomial time.

Networks such as STNU can be used as temporal foundation for a broad class of Process-Aware Information Systems currently being developed [Reichert and Weber, 2012]. In this context, the extension proposed in this paper may be used to better represent the temporal properties of sub processes (i.e., complex tasks). It is quite common to have sub processes whose allowed durations can be restricted in a limited way prior to their execution. In turn, once a sub process starts to execute, it is necessary to guarantee that the
allowed duration range can be used without any further interference.

There are different avenues for future work. First, we want to study the applicability of our approach to CSTNU, for which the presence of labeled constraints and links requires to consider further and different propagation rules that have to be extended to take account of guarded constraints. Second, the application of STNPSU as temporal foundation of Process-Aware Information Systems could be interesting. Particularly, STNPSU might serve as a tool for an appropriate and scalable analysis of the temporal properties of the business processes.

REFERENCES


<table>
<thead>
<tr>
<th>#</th>
<th>Authors</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>91-01</td>
<td>Ker-I Ko, P. Orponen, U. Schöning, O. Watanabe</td>
<td>Instance Complexity</td>
</tr>
<tr>
<td>91-02*</td>
<td>K. Gladitz, H. Fassbender, H. Vogler</td>
<td>Compiler-Based Implementation of Syntax-Directed Functional Programming</td>
</tr>
<tr>
<td>91-03*</td>
<td>Alfons Geser</td>
<td>Relative Termination</td>
</tr>
<tr>
<td>91-04*</td>
<td>J. Köbler, U. Schöning, J. Toran</td>
<td>Graph Isomorphism is low for PP</td>
</tr>
<tr>
<td>91-05</td>
<td>Johannes Köbler, Thomas Thierauf</td>
<td>Complexity Restricted Advice Functions</td>
</tr>
<tr>
<td>91-06*</td>
<td>Uwe Schöning</td>
<td>Recent Highlights in Structural Complexity Theory</td>
</tr>
<tr>
<td>91-07*</td>
<td>F. Green, J. Köbler, J. Toran</td>
<td>The Power of Middle Bit</td>
</tr>
<tr>
<td>91-08*</td>
<td>V. Arvind, Y. Han, L. Hamachandra, J. Köbler, A. Lozano, M. Mundhenk, A. Ogiwara, U. Schöning, R. Silvestri, T. Thierauf</td>
<td>Reductions for Sets of Low Information Content</td>
</tr>
<tr>
<td>92-01*</td>
<td>Vikraman Arvind, Johannes Köbler, Martin Mundhenk</td>
<td>On Bounded Truth-Table and Conjunctive Reductions to Sparse and Tally Sets</td>
</tr>
<tr>
<td>92-02*</td>
<td>Thomas Noll, Heiko Vogler</td>
<td>Top-down Parsing with Simulataneous Evaluation of Noncircular Attribute Grammars</td>
</tr>
<tr>
<td>92-03</td>
<td>Fakultät für Informatik</td>
<td>17. Workshop über Komplexitätstheorie, effiziente Algorithmen und Datenstrukturen</td>
</tr>
<tr>
<td>92-04*</td>
<td>V. Arvind, J. Köbler, M. Mundhenk</td>
<td>Lowness and the Complexity of Sparse and Tally Descriptions</td>
</tr>
<tr>
<td>92-05*</td>
<td>Johannes Köbler</td>
<td>Locating P/poly Optimally in the Extended Low Hierarchy</td>
</tr>
<tr>
<td>92-06*</td>
<td>Armin Kühnemann, Heiko Vogler</td>
<td>Synthesized and inherited functions -a new computational model for syntax-directed semantics</td>
</tr>
<tr>
<td>92-07*</td>
<td>Heinz Fassbender, Heiko Vogler</td>
<td>A Universal Unification Algorithm Based on Unification-Driven Leftmost Outermost Narrowing</td>
</tr>
</tbody>
</table>
92-08* Uwe Schöning
On Random Reductions from Sparse Sets to Tally Sets

92-09* Hermann von Hasseln, 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
94-05  V. Arvind, J. Köbler, R. Schuler  
On Helping and Interactive Proof Systems

94-06  Christian Kalus, Peter Dadam  
Incorporating record subtyping into a relational data model

94-07  Markus Tresch, Marc H. Scholl  
A Classification of Multi-Database Languages

94-08  Friedrich von Henke, Harald Rueß  
Arbeitstreffen Typtheorie: Zusammenfassung der Beiträge

Construction and Deduction Methods for the Formal Development of Software

94-10  Axel Dold  
Formalisierung schematischer Algorithmen

94-11  Johannes Köbler, Osamu Watanabe  
New Collapse Consequences of NP Having Small Circuits

94-12  Rainer Schuler  
On Average Polynomial Time

94-13  Rainer Schuler, Osamu Watanabe  
Towards Average-Case Complexity Analysis of NP Optimization Problems

94-14  Wolfram Schulte, Ton Vullinghs  
Linking Reactive Software to the X-Window System

94-15  Alfred Lupper  
Namensverwaltung und Adressierung in Distributed Shared Memory-Systemen

94-16  Robert Regn  
Verteilte Unix-Betriebssysteme

94-17  Helmuth Partsch  
Again on Recognition and Parsing of Context-Free Grammars:  
Two Exercises in Transformational Programming

94-18  Helmuth Partsch  
Transformational Development of Data-Parallel Algorithms: an Example

95-01  Oleg Verbitsky  
On the Largest Common Subgraph Problem

95-02  Uwe Schöning  
Complexity of Presburger Arithmetic with Fixed Quantifier Dimension

95-03  Harry Buhrman, Thomas Thierauf  
The Complexity of Generating and Checking Proofs of Membership

95-04  Rainer Schuler, Tomoyuki Yamakami  
Structural Average Case Complexity

95-05  Klaus Achatz, Wolfram Schulte  
Architecture Independent Massive Parallelization of Divide-And-Conquer Algorithms
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 Schultess
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, Helmuth 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 Kölber*
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\textsubscript{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 Number</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 Wennemers</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>97-18</td>
<td>Johannes Köbler, Rainer Schuler</td>
<td>Using Efficient Average-Case Algorithms to Collapse Worst-Case Complexity Classes</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 Wennemers</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>
</tbody>
</table>
98-11  Frank Houdek, Dietmar Ernst, Thilo Schwinn
Prüfen von C–Code und Statmate/Matlab–Spezifikationen: Ein Experiment

98-12  Gerhard Schellhorn
Proving Properties of Directed Graphs: A Problem Set for Automated Theorem Provers

98-13  Gerhard Schellhorn, Wolfgang Reif
Theorems from Compiler Verification: A Problem Set for Automated Theorem Provers

98-14  Mohammad Ali Livani
SHARE: A Transparent Mechanism for Reliable Broadcast Delivery in CAN

98-15  Mohammad Ali Livani, Jörg Kaiser
Predictable Atomic Multicast in the Controller Area Network (CAN)

99-01  Susanne Boll, Wolfgang Klas, Utz Westermann
A Comparison of Multimedia Document Models Concerning Advanced Requirements

99-02  Thomas Bauer, Peter Dadam
Verteilungsmodelle für Workflow-Management-Systeme - Klassifikation und Simulation

99-03  Uwe Schöning
On the Complexity of Constraint Satisfaction

99-04  Ercument Canver
Model-Checking zur Analyse von Message Sequence Charts über Statecharts

99-05  Johannes Köbler, Wolfgang Lindner, Rainer Schuler
Derandomizing RP if Boolean Circuits are not Learnable

99-06  Utz Westermann, Wolfgang Klas
Architecture of a DataBlade Module for the Integrated Management of Multimedia Assets

99-07  Peter Dadam, Manfred Reichert

99-08  Vikraman Arvind, Johannes Köbler
Graph Isomorphism is Low for ZPPNP and other Lowness results

99-09  Thomas Bauer, Peter Dadam
Efficient Distributed Workflow Management Based on Variable Server Assignments

2000-02  Thomas Bauer, Peter Dadam
Variable Serverzuordnungen und komplexe Bearbeiterzuordnungen im Workflow-Management-System ADEPT

2000-03  Gregory Baratoff, Christian Toepfer, Heiko Neumann
Combined space-variant maps for optical flow based navigation
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

2002-03  Axel Dold, Friedrich W. von Henke, Vincent Vialard, Wolfgang Goerigk  
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
2003-03  Christian Heinlein
Safely Extending Procedure Types to Allow Nested Procedures as Values

2003-04  Stefanie Rinderle, Manfred Reichert, Peter Dadam
On Dealing With Semantically Conflicting Business Process Changes.

2003-05  Christian Heinlein
Dynamic Class Methods in Java

2003-06  Christian Heinlein
Vertical, Horizontal, and Behavioural Extensibility of Software Systems

2003-07  Christian Heinlein
Safely Extending Procedure Types to Allow Nested Procedures as Values
(Corrected Version)

2003-08  Changling Liu, Jörg Kaiser
Survey of Mobile Ad Hoc Network Routing Protocols)

2004-01  Thom Frühwirth, Marc Meister (eds.)
First Workshop on Constraint Handling Rules

2004-02  Christian Heinlein
Concept and Implementation of C++, an Extension of C++ to Support User-Defined Operator Symbols and Control Structures

2004-03  Susanne Biundo, Thom Frühwirth, Günther Palm(eds.)
Poster Proceedings of the 27th Annual German Conference on Artificial Intelligence

2005-01  Armin Wolf, Thom Frühwirth, Marc Meister (eds.)
19th Workshop on (Constraint) Logic Programming

2005-02  Wolfgang Lindner (Hg.), Universität Ulm , Christopher Wolf (Hg.) KU Leuven
2. Krypto-Tag – Workshop über Kryptographie, Universität Ulm

2005-03  Walter Guttmann, Markus Maucher
Constrained Ordering

2006-01  Stefan Sarstedt
Model-Driven Development with ACTIVECHARTS, Tutorial

2006-02  Alexander Raschke, Ramin Tavakoli Kolagari
Ein experimenteller Vergleich zwischen einer plan-getriebenen und einer leichtgewichtigen Entwicklungsmethode zur Spezifikation von eingebetteten Systemen

2006-03  Jens Kohlmeyer, Alexander Raschke, Ramin Tavakoli Kolagari
Eine qualitative Untersuchung zur Produktlinien-Integration über Organisationsgrenzen hinweg

2006-04  Thorsten Liebig
Reasoning with OWL - System Support and Insights –

2008-01  H.A. Kestler, J. Messner, A. Müller, R. Schuler
On the complexity of intersecting multiple circles for graphical display
2008-02  Manfred Reichert, Peter Dadam, Martin Jurisch, Ulrich Kreher, Kevin Göser, Markus Lauer
Architectural Design of Flexible Process Management Technology

2008-03  Frank Raiser
Semi-Automatic Generation of CHR Solvers from Global Constraint Automata

2008-04  Ramin Tavakoli Kolagari, Alexander Raschke, Matthias Schneiderhan, Ian Alexander
Entscheidungsdokumentation bei der Entwicklung innovativer Systeme für produktlinien-basierte Entwicklungsprozesse

2008-05  Markus Kalb, Claudia Dittrich, Peter Dadam
Support of Relationships Among Moving Objects on Networks

2008-06  Matthias Frank, Frank Kargl, Burkhard Stiller (Hg.)
WMAN 2008 – KuVS Fachgespräch über Mobile Ad-hoc Netzwerke

2008-07  M. Maucher, U. Schöning, H.A. Kestler
An empirical assessment of local and population based search methods with different degrees of pseudorandomness

2008-08  Henning Wunderlich
Covers have structure

2008-09  Karl-Heinz Niggl, Henning Wunderlich
Implicit characterization of FPTIME and NC revisited

2008-10  Henning Wunderlich
On span-$P^e$ and related classes in structural communication complexity

2008-11  M. Maucher, U. Schöning, H.A. Kestler
On the different notions of pseudorandomness

2008-12  Henning Wunderlich
On Toda’s Theorem in structural communication complexity

2008-13  Manfred Reichert, Peter Dadam
Realizing Adaptive Process-aware Information Systems with ADEPT2

2009-01  Peter Dadam, Manfred Reichert
The ADEPT Project: A Decade of Research and Development for Robust and Flexible Process Support
Challenges and Achievements

2009-02  Peter Dadam, Manfred Reichert, Stefanie Rinderle-Ma, Kevin Göser, Ulrich Kreher, Martin Jurisch
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 Â-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
Analyzing 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
Ulmer Informatik-Berichte
ISSN 0939-5091

Herausgeber:
Universität Ulm
Fakultät für Ingenieurwissenschaften und Informatik
89069 Ulm