RWTH Aachen

Department of Computer Science
Technical Report

Counterexamples in Probabilistic Model Checking

Tingting Han and Joost-Pieter Katoen

ISSN 0935–3232 · Aachener Informatik Berichte · AIB-2006-09

RWTH Aachen · Department of Computer Science · August 2006

Counterexamples in Probabilistic Model Checking

Tingting Han and Joost-Pieter katoen

Lehrstuhl für Informatik 2
RWTH Aachen, Germany
Email: {tingting.han, katoen}@informatik.rwth-aachen.de

Abstract. This paper considers algorithms and complexity results for the generation of counterexamples in model checking of probabilistic until-formulae in discrete-time Markov chains (DTMCs). It is shown that finding the strongest evidence (i.e, the most probable path) that violates a (bounded) until-formula can be found in polynomial time using single-source (hop-constrained) shortest path algorithms. We also show that computing the smallest counterexample that is mostly deviating from the required probability bound can be found in a pseudo-polynomial time complexity by adopting a certain class of algorithms for the (hop-constrained) k shortest paths problem.

1 Introduction

A major strength of model checking is the possibility to generate counterexamples in case a property is violated. The shape of a counterexample depends on the temporal logic used. For LTL and the universal fragment of CTL, a single path through the system model suffices to indicate the refutation of the property. For existentially quantified path-formulae in logics such as CTL, either witnesses are provided (to indicate why the property holds), or more advanced structures such as trees of paths [10] or annotated paths [28] are provided as counterexample. Counterexamples are of utmost importance in model checking: they provide diagnostic feedback (also in cases where only a fragment of the entire model can be searched), they constitute the key to successful abstraction-refinement techniques [9], and are at the core of obtaining (optimal) schedules in e.g. timed model checking [7]. As a result, advanced counterexample generation and analysis techniques have intensively been investigated, see e.g., [23, 6, 12].

In probabilistic model checking, however, counterexample generation is almost not developed [2, 3]. Probabilistic model checking is a technique to verify system models in which transitions are equipped with randomness. Popular models are discrete- and continuous-time Markov chains (DTMCs and CTMCs), and variants thereof which exhibit nondeterminism. Efficient model-checking algorithms have been developed for these models, and have been applied to case studies from various application areas. The crux of probabilistic model checking is to combine techniques from numerical mathematics with standard reachability analysis. In this way, properties such as "the probability to reach a set of goal states is at most 0.6" can be automatically checked (up to a certain precision).

In the probabilistic setting, typically there is no single trace (but rather a set of them) that indicates why a given property is refuted. In case of a property refutation, current probabilistic model checkers produce a log file that shows the computed probability for all states. This information is too detailed to aid as a useful support in finding the cause(s). This paper considers algorithms and complexity results for the generation of counterexamples in model checking of (a safe fragment of) probabilistic CTL [20] on DTMCs. We concentrate on properties of the form $\mathcal{P}_{\leqslant p}(\Phi \mathcal{U}^{\leqslant h} \Psi)$. In case s refutes this formula, the probability of all paths in s satisfying $\Phi \mathcal{U}^{\leqslant h} \Psi$ exceeds p. We first consider the generation of *strongest evidences* for violation, i.e., paths satisfying $\Phi \mathcal{U}^{\leqslant h} \Psi$ that have the largest probability mass. Strongest evidences "contribute" mostly to the property refutation. For unbounded until (i.e., $h{=}\infty$), determining strongest evidences is equivalent to a standard shortest path (SP) problem; in case h is bounded, we obtain a special case of the (resource) constrained shortest path (CSP) problem [1] that can be solved in $\mathcal{O}(hm)$ where m is the number of transitions in the DTMC.

As most probable paths may have a very small probability mass, their information may be limited. As a next step, therefore, we consider the problem of determining most probable subtrees. Whereas in traditional model checking one is interested in the shortest counterexample, we consider trees of *smallest size* that exceed the probability bound *maximally.* The problem of generating such smallest, most indicative counterexamples can be casted as a k shortest paths problem. For unbounded-until formulae (i.e., $h{=}\infty$), it is shown that the generation of such smallest counterexamples can be found in pseudo-polynomial time by adopting k shortest path algorithms [14, 17, 26] that can compute k on the fly. For bounded until-formulae, we propose a variant of the recursive enumeration algorithm of Jiménez and Marzal [22]. The time complexity of this adapted algorithm is $\mathcal{O}(hm{+}hk\log(\frac{m}{n}))$, where n is the number of states in the DTMC.

2 Preliminaries

This section introduces DTMCs and the logic PCTL-safety.

2.1 DTMCs

Definition 1 (DTMCs). *A (labelled)* discrete-time Markov chain *(DTMC) is a tuple $\mathcal{D} = (S, \boldsymbol{P}, L)$ where:*

- *S is a finite set of states;*
- *$\boldsymbol{P} : S \times S \to [0,1]$ is a probability matrix satisfying $\sum_{s' \in S} \boldsymbol{P}(s,s') = 1$ for all $s \in S$;*
- *$L : S \to 2^{AP}$ is a labelling function which assigns to each state $s \in S$ the set $L(s)$ of atomic propositions that are valid in s.*

A state s in $\mathcal{D}$ is called absorbing if $\mathbf{P}(s,s) = 1$. W.l.o.g. we assume a DTMC to have a unique initial state.

Definition 2 (Paths). *Let $\mathcal{D} = (S, \boldsymbol{P}, L)$ be a DTMC.*

- *An* infinite path *σ in $\mathcal{D}$ is an infinite sequence $s_0{\cdot}s_1{\cdot}s_2{\cdot}...$ of states such that $\boldsymbol{P}(s_i, s_{i+1}) > 0$ for all $i \geqslant 0$.*
- *A* finite path *in $\mathcal{D}$ is a finite prefix of an infinite path.*

For state s and path σ, $\sigma{\cdot}s$ denotes the path obtained by extending σ by s. Let $|\sigma|$ denote the length of the path σ, i.e., $|s_0{\cdot}s_1{\cdot}...{\cdot}s_n| = n$, $|s_0| = 0$ and $|\sigma| = \infty$ for infinite σ. For $0 \leqslant i \leqslant |\sigma|$, $\sigma[i] = s_i$ denotes the i-th state in σ. $Path(s)$ denotes the set of infinite paths that start in state s, formally, $Path(s) = \{\sigma \mid \sigma[0] = s\}$. Let $Path_{fin}(s)$ denotes the set of finite paths that start in state s.

A DTMC $\mathcal{D}$ enriched with an initial state s_0 induces a probability space. The underlying σ-algebra from the basic cylinders is induced by the finite paths starting in s_0. The probability measure $\Pr^{\mathcal{D}}_{s_0}$ (briefly Pr) induced by $(\mathcal{D}, s_0)$ is the unique measure on this σ-algebra where:

$$\Pr\underbrace{\{\sigma \in Path(s_0) \mid s_0{\cdot}s_1{\cdot}\ldots{\cdot}s_n \text{ is a prefix of } \sigma\}}_{\text{basic cylinder of the finite path } s_0\cdot s_1\cdot\ldots\cdot s_n} = \prod_{0 \leqslant i < n} \mathbf{P}(s_i, s_{i+1}).$$

Example 1. Fig. 1 illustrates a simple DTMC consisting of 10 states. s is the initial state, $AP = \{a, b\}$ and L is given through the subsets of AP labelling the states as $L(s) = L(s_i) = \{a\}$, for $1 \leqslant i \leqslant 4$; $L(t_1) = L(t_2) = L(t_3) = \{b\}$ and $L(u_1) = L(u_2) = \emptyset$. The DTMC contains no absorbing states. $\sigma_1 = s \cdot u_1 \cdot u_2 \cdot s_1 \cdot t_1 {\cdot} s_3$ is a finite path in this DTMC with $\Pr\{\sigma_1\} = 0.1 \times 0.4 \times 0.8 \times 0.1 \times 1 = 0.0032$ and $|\sigma_1| = 5$, $\sigma_1[3] = s_1$. $\sigma_2 = s{\cdot}s_3{\cdot}(s_4{\cdot}t_3)^\omega$ is an infinite path, $|\sigma_2| = \infty$.

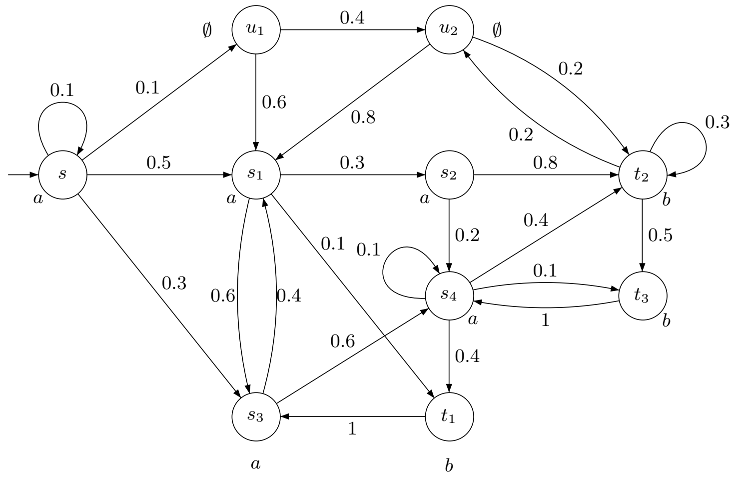


Fig. 1. An example DTMC

2.2 PCTL-safety

In this paper, counterexamples are explored when properties specified in the safety fragment of PCTL [5] are violated. The syntax and semantics of PCTL-safety are given and its expressiveness is illustrated by some examples.

Syntax. Let $p \in [0, 1]$ and let AP denote a fixed, finite set of atomic propositions ranged over by $a, b, c, \ldots$. The syntax of PCTL-safety state formulae (in positive normal form, i.e., negations can only occur adjacent to atomic propositions.) is defined as follows:

$$\varPhi ::= \mathrm{tt} \mid \mathrm{ff} \mid a \mid \neg a \mid \varPhi \wedge \varPhi \mid \varPhi \vee \varPhi \mid \mathcal{P}_{\leqslant p}(\phi)$$

where ϕ is path-formula defined by

$$\phi ::= \Phi \mathcal{U}^{\leqslant h} \Phi,$$

where $h \in \mathbb{N} \cup \{\infty\}$. We call the operator $\mathcal{U}^{\leqslant h}$ *unbounded until* if $h = \infty$ and abbreviate it as $\mathcal{U}$; and call it *bounded until* otherwise. For the sake of simplicity, we do not consider the next-operator. Note that the main distinction with PCTL [20] is that the probability bounds are upper bounds, and formulae are required to be in positive normal form.

As for CTL, the temporal operator $\lozenge^{\leqslant h}$ (eventually) can be derived as

$$\mathcal{P}_{\leqslant p}(\lozenge^{\leqslant h}\Phi) = \mathcal{P}_{\leqslant p}(\mathrm{tt}\ \mathcal{U}^{\leqslant h}\ \Phi)$$

The dual form is defined as $\mathcal{P}_{>p}(\Box^{\leqslant h}\Phi) = \mathcal{P}_{\leqslant 1-p}(\lozenge^{\leqslant h}\neg\Phi)$. Note that the negation can be "pushed" inside Φ until finally it is adjacent to an atomic proposition (see [5])[1]. For the more general case, $\mathcal{P}_{>p}(\Phi\mathcal{W}^{\leqslant h}\Psi) = \mathcal{P}_{\leqslant 1-p}(\neg\Psi\mathcal{U}^{\leqslant h}\neg(\Phi \vee \Psi))$. Thus in the PCTL-safety fragment, four forms $\mathcal{P}_{\leqslant p}(\Phi\mathcal{U}^{\leqslant h}\Psi)$, $\mathcal{P}_{>p}(\Phi\mathcal{W}^{\leqslant h}\Psi)$, $\mathcal{P}_{\leqslant p}(\lozenge^{\leqslant h}\Phi)$ and $\mathcal{P}_{>p}(\Box^{\leqslant h}\Phi)$ are allowed.

Semantics. Let DTMC $\mathcal{D} = (S, \mathbf{P}, L)$. The semantics of PCTL-safety is defined by a satisfaction relation, denoted $\models$, which is characterized as the least relation over the states in S (infinite paths in $\mathcal{D}$, respectively) and the state formulae (path formulae). The semantics of the propositional fragment is identical to that for CTL. The meaning of the probabilistic operator is formalized as follows [20]. The semantics of PCTL-safety state formulae is defined for path formula ϕ as:

$s \models \mathrm{tt}$	iff	*true*	$s \models \Phi \vee \Psi$	iff	$s \models \Phi$ or $s \models \Psi$
$s \models \mathrm{ff}$	iff	*false*	$s \models \Phi \wedge \Psi$	iff	$s \models \Phi$ and $s \models \Psi$
$s \models a$	iff	$a \in L(s)$	$s \models \mathcal{P}_{\leqslant p}(\phi)$	iff	$\mathit{Prob}(s, \phi) \leqslant p$
$s \models \neg a$	iff	$a \notin L(s)$			

Let $\mathit{Path}(s, \phi)$ denote the set of infinite paths that start in state s and satisfy ϕ. Formally, $\mathit{Path}(s, \phi) = \{\sigma \in \mathit{Path}(s) \mid \sigma \models \phi\}$. Here, $\mathit{Prob}(s, \phi) = \Pr\{\sigma \in \mathit{Path}(s, \phi)\}$ denotes the probability of $\mathit{Path}(s, \phi)$. Let σ be an infinite path in $\mathcal{D}$. The semantics of PCTL-safety path formulae is defined as:

$$\sigma \models \Phi\mathcal{U}^{\leqslant h}\Psi \quad \text{iff} \quad \exists i \leqslant h \text{ such that } \sigma[i] \models \Psi \text{ and } \forall j : 0 \leqslant j < i.(\sigma[j] \models \Phi).$$

For finite path σ, $\models$ is defined in a similar way by changing the range of i to $i \leqslant \max\{h, |\sigma|\}$. Let $\mathit{Path}_{\mathit{fin}}(s, \phi)$ denote the set of finite paths starting in s that fulfill ϕ.

Example 2. We give some examples to illustrate how the PCTL-safety formulae are utilized to specify system properties.

- In Fig. 1, $\mathcal{P}_{\leqslant 0.27}(a\mathcal{U}b)$ asserts that the probability of reaching a b-state via an a-path is at most 0.27.

[1] During the transformation, arbitrary PCTL-formulae can occur, but the result is again a PCTL-safety formula.

- Let *error* be an atomic proposition that characterizes any state in which a system error has occurred. Then $\mathcal{P}_{\leqslant 0.001}(\Diamond^{\leqslant 50} error)$ asserts that the probability for a system error to occur within 50 steps is at most 0.001. Dually, $\mathcal{P}_{>0.999}(\Box^{\leqslant 50} \neg error)$ states that the probability for not having a system error (running successfully) within 50 steps exceeds 0.999.
- Let *red* and *green* be two atomic propositions. $\mathcal{P}_{>0.8}(green\ \mathcal{W}\ red)$ asserts that the probability of either being green forever, or reaching a red state via a green path, is greater than 0.8. Stated differently, with probability at most 0.2, a state is reached that is neither red nor green via a path that does not contain a red state, which can be specified by the dual formula $\mathcal{P}_{\leqslant 0.2}((\neg red)\mathcal{U}(\neg green \wedge \neg red))$.

3 Counterexamples in a probabilistic setting

Let us first consider what a counterexample actually is. To that end, consider the formula $\mathcal{P}_{\leqslant p}(\phi)$, where ϕ is a path-formula. If state s refutes $\mathcal{P}_{\leqslant p}(\phi)$:

$$\begin{array}{ll} & s \not\models \mathcal{P}_{\leqslant p}(\phi) \\ \text{iff} & \text{not } (Prob(s, \phi) \leqslant p) \\ \text{iff} & Prob(s, \phi) > p \\ \text{iff} & \Pr\{\sigma \mid \sigma \in Path(s, \phi)\} > p \end{array}$$

So, $\mathcal{P}_{\leqslant p}(\phi)$ is refuted by state s whenever the total probability mass of all ϕ-paths that start in s exceeds p. This indicates that a counterexample for $\mathcal{P}_{\leqslant p}(\phi)$ is in general a *set* of paths satisfying ϕ. As ϕ is an until-formula and can be witnessed by finite state sequences, finite paths do suffice in counterexamples. As a counterexample should exceed p, a maximally probable ϕ-path is a strong evidence for the violation of $\mathcal{P}_{\leqslant p}(\phi)$. For counterexamples that are as small as possible, i.e., that contain the smallest possible set of paths indicating the refutation, such maximally probable paths are essential.

Definition 3 (Strongest evidence). *A* strongest evidence *for violating $\mathcal{P}_{\leqslant p}(\phi)$ in state s is a finite path $\sigma \in Path_{fin}(s, \phi)$ such that $Pr\{\sigma\} \geqslant Pr\{\sigma'\}$ for any $\sigma' \in Path_{fin}(s, \phi)$.*

Dually, a strongest evidence for violating $\mathcal{P}_{\leqslant p}(\phi)$ is a strongest witness for $\mathcal{P}_{>p}(\phi)$ [2]. Note that a strongest evidence does not need to be a counterexample as its probability mass may be (far) below p. A counterexample is defined as follows:

Definition 4 (Counterexample). *A* counterexample *for $\mathcal{P}_{\leqslant p}(\phi)$ in state s is a set C of paths such that $C \subseteq Path_{fin}(s, \phi)$ and $Pr(C) > p$.*

Note that counterexamples are always finite as we consider non-strict upper-bounds in the probability operator[3]. Let $CX_p(s, \phi)$ be the set of counterexamples for $\mathcal{P}_{\leqslant p}(\phi)$ in state s. For $C \in CX_p(s, \phi)$ and C's superset C': $C \subseteq C' \subseteq$

[2] $\mathcal{P}_{>p}(\phi)$ is a PCTL-formula, not necessarily a safe one.

[3] For strict upper bounds in the probability operator, i.e., $\mathcal{P}_{<p}(\phi)$, a counterexample C may contain infinite paths, since $\Pr(C) = p$ is a counterexample. The limit of the sum of path probabilities obeying a geometric distribution may equal p, but the limit requires infinite paths.

$Path_{fin}(s, \phi)$, it follows that $C' \in CX_p(s, \phi)$, since $\Pr(C') \geqslant \Pr(C) > p$. A counterexample is called *minimal* if it is minimal w.r.t. $\subseteq$. Note that a counterexample for state s is a set of finite paths that all start in s, and thus can be considered as a finite tree rooted at s.

As in conventional model checking, we are not interested in generating arbitrary counterexamples, but those that are easy to comprehend, and provide a clear evidence of the refutation of the formula. So, akin to shortest counterexamples we define the notion of a smallest, most indicative counterexample. Such smallest counterexamples should contain as few paths as possible—allowing easier analysis of the cause of refutation—but whose probability is clearly exceeding p.

Definition 5 (Smallest counterexample). *$C \in CX_p(s, \phi)$ is a* smallest (most indicative) counterexample *if:*

1. *$|C| \leqslant |C'|$, for any $C' \in CX_p(s, \phi)$ and*
2. *$Pr(C) \geqslant Pr(C'')$, for any $C'' \in CX_p(s, \phi)$ and $|C| = |C''|$.*

Intuitively, a smallest counterexample is mostly deviating from the required probability bound given that it has the smallest number of paths. Any smallest counterexample is minimal, but not necessarily the reverse. Note that the strongest evidence, minimal counterexample or smallest counterexample may not be unique, as paths may have equal probability. As a result, not every strongest evidence is contained in a minimal (or smallest) counterexample. However, any smallest counterexample contains at least one strongest evidence.

Example 3. In the DTMC in Fig. 1, we have $s \not\models \mathcal{P}_{\leqslant 0.27}(a\mathcal{U}b)$. Let $\sigma_1 = s{\cdot}s_1{\cdot}s_2{\cdot}t_2$, $\sigma_2 = s{\cdot}s_1{\cdot}s_3{\cdot}s_4{\cdot}t_1$, $\sigma_3 = s{\cdot}s_1{\cdot}s_3{\cdot}s_4{\cdot}t_2$, $\sigma_4 = s{\cdot}s_3{\cdot}s_4{\cdot}t_1$, $\sigma_5 = s{\cdot}s_3{\cdot}s_4{\cdot}t_2$, $\sigma_6 = s{\cdot}s_3{\cdot}s_4{\cdot}t_3$, where:

$$\Pr\{\sigma_1\} = 0.12,\ \Pr\{\sigma_2\} = \Pr\{\sigma_3\} = \Pr\{\sigma_4\} = \Pr\{\sigma_5\} = 0.072,\ \Pr\{\sigma_6\} = 0.018.$$

Path σ_1 is a strongest evidence, as it is the maximally probable path from s to $\{t_1, t_2, t_3\}$, i.e., b-states. The set $C_1 = \{\sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_6\}$ with $\Pr(C_1) = 0.306$ is a counterexample, but neither a minimal counterexample, nor a smallest counterexample, as $C_2 = \{\sigma_2, \sigma_3, \sigma_4, \sigma_5\} \subset C_1$ with $\Pr(C_2) = 0.288$ is also a counterexample. C_2 is a minimal counterexample, since the probability of any proper subset of C_2 is less than 0.27. But C_2 is not a smallest counterexample, as $C_3 = \{\sigma_1, \sigma_2, \sigma_3, \sigma_4\}$ with $\Pr(C_3) = 0.336$ is a counterexample too and $|C_2| = |C_3|$ but $\Pr(C_3) > \Pr(C_2)$. In fact, any set containing the strongest evidence and any three paths in C_2 is a smallest counterexample.

In the remainder of the paper, we first consider the computation of strongest evidences. Formally, we consider the strongest evidence problem (SE), that for a given state s with $s \not\models \mathcal{P}_{\leqslant p}(\phi)$, determines the strongest evidence for this violation. Subsequently, we consider the corresponding smallest counterexample problem (SC). For both cases, we distinguish between until-formulae for which $h{=}\infty$ (unbounded until) and $h \in \mathbb{N}$ (bounded until) as distinctive algorithms are used for these cases.

4 From DTMC to a weighted digraph

Prior to finding strongest evidences or shortest counterexamples, we first modify the DTMC and turn it into a weighted directed graph. Let $\phi = \Phi \mathcal{U}^{\leqslant h} \Psi$, $h = \infty$ or $h \in \mathbb{N}$ and $Sat(\Phi) = \{s \in S \mid s \models \Phi\}$ for any state-formula Φ. Due to the bottom-up traversal of the model-checking algorithm over the formula ϕ, we may assume that $Sat(\Phi)$ and $Sat(\Psi)$ are known.

Step 1: Adapting the DTMC. First, we make all states in the DTMC $\mathcal{D} = (S, \mathbf{P}, L)$ that neither satisfy Φ nor Ψ absorbing. Then we add an extra state t so that all the Ψ-states are equipped with a transition to t with probability 1 (all other outgoing transitions of Ψ-states are omitted). State t can thus only be reached via a Ψ-state. The thus obtained DTMC $\mathcal{D}' = (S', \mathbf{P}', L')$ has state space $S \cup \{t\}$ for $t \notin S$. The transition probabilities in $\mathcal{D}'$ are defined as follows:

$$\begin{cases} \mathbf{P}'(s',s') = 1 \text{ and } \mathbf{P}'(s',s'') = 0 \text{ for } s'' \neq s' & \text{if } s' \notin Sat(\Phi) \cup Sat(\Psi) \text{ or } s' = t \\ \mathbf{P}'(s',t) = 1 \text{ and } \mathbf{P}'(s',s'') = 0 \text{ for } s'' \neq t & \text{if } s' \in Sat(\Psi) \\ \mathbf{P}'(s',s'') = \mathbf{P}(s',s'') \text{ for } s'' \in S \text{ and } \mathbf{P}'(s',t) = 0 & \text{otherwise} \end{cases}$$

$L'(s') = L(s')$ for $s' \in S$ and $L'(t) = \{at_t\}$, where $at_t \notin L(s')$ for any $s' \in S$, i.e., at_t uniquely identifies being at state t. Remark that all the $(\neg\Phi \wedge \neg\Psi)$-states could be collapsed into a single state, but this is not further explored here. The time complexity of this transformation is $\mathcal{O}(n)$ where $n = |S|$.

It is evident that the validity of $\Phi \mathcal{U}^{\leqslant h} \Psi$ is not affected by this amendment of the DTMC. All paths in $\mathcal{D}'$ of length at most $h+1$ that end in t satisfy $\Phi \mathcal{U}^{\leqslant h} \Psi$ in $\mathcal{D}$. More precisely, any finite path satisfying $(\Phi \vee \Psi) \mathcal{U}^{\leqslant h+1} at_t$ in $\mathcal{D}'$ has a finite path in $\mathcal{D}$ satisfying $\Phi \mathcal{U}^{\leqslant h} \Psi$. The following lemma guarantees that the later results in $\mathcal{D}'$ also hold in $\mathcal{D}$.

Lemma 1. *Let $\sigma' = \sigma \cdot t$ be a path in $Path_{fin}(s)$ in $\mathcal{D}'$. Then:*

1. $\sigma' \models (\Phi \vee \Psi) \mathcal{U}^{\leqslant h+1} at_t$ *in DTMC* $\mathcal{D}'$;
2. $\sigma \models \Phi \mathcal{U}^{\leqslant h} \Psi$ *in DTMC* $\mathcal{D}$;
3. $\Pr\{\sigma\} = \Pr\{\sigma'\}$.

For the rest of the technical report, we suppose that all the DTMCs are the results of Step 1, and the logic formula considered is $(\Phi \vee \Psi) \mathcal{U}^{\leqslant h+1} at_t$.

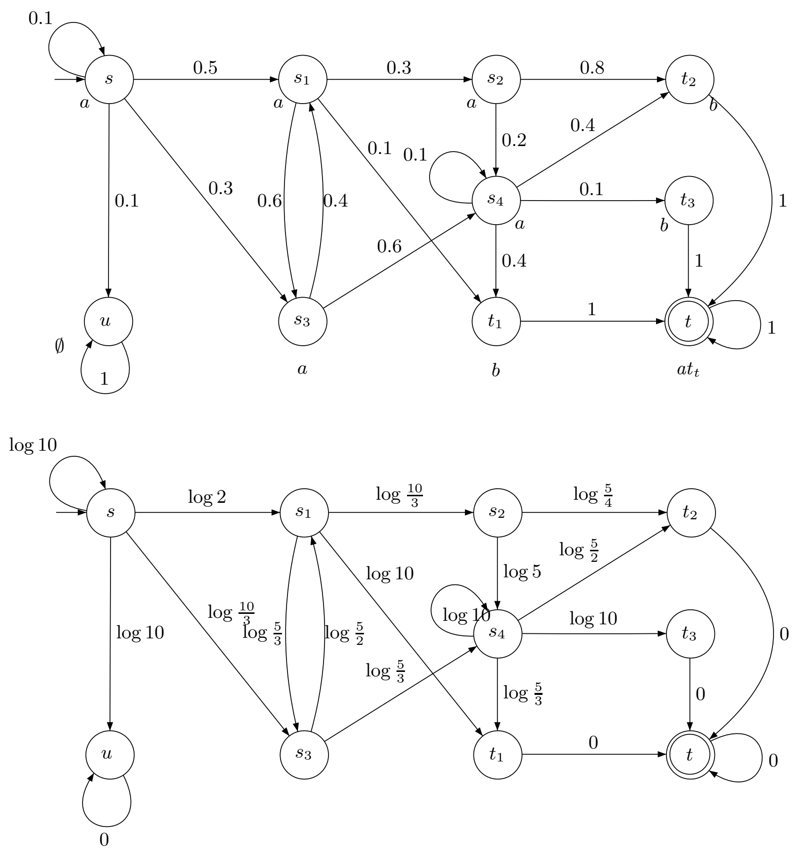
Step 2: Conversion into a weighted digraph. As a second preprocessing step, the DTMC obtained in the first phase is transformed into a weighted digraph. Recall that a weighted digraph is a tuple $\mathcal{G} = (V, E, w)$ where V is a finite set of vertices, $E : V \times V$ is a set of edges, and $w : E \to \mathbb{R}_{\geqslant 0} \cup \{\infty\}$ is a function assigning non-negative weights to edges.

Definition 6 (Weighted digraph of a DTMC). *For DTMC $\mathcal{D} = (S, \boldsymbol{P}, L)$, the weighted digraph $\mathcal{G}_D = (V, E, w)$ where:*

- $V = S$ *and* $(v, v') \in E$ *iff* $\boldsymbol{P}(v, v') > 0$, *and*
- $w(v,v') = \begin{cases} \log(\boldsymbol{P}(v,v')^{-1}) & \text{if } \boldsymbol{P}(v,v') > 0, \\ \infty & \text{otherwise.} \end{cases}$

Note that in any DTMC, $\mathbf{P}(s, s') \in [0, 1]$, thus $\mathbf{P}(s, s')^{-1} \in [1, +\infty)$, and consequently, $\log(\mathbf{P}(s, s')^{-1}) \in [0, +\infty)$. Thus, we indeed obtain a non-negatively weighted digraph. Note that this transformation can be done in $\mathcal{O}(m)$, where $m = |\mathbf{P}|$.

Example 4. The transformation from DTMC in Fig. 1 to a weighted digraph is illustrated in Fig. 2. For the path formula $\phi = a\mathcal{U}b$, in Fig. 2(a) all the b-states (i.e., t_1, t_2, t_3) are made absorbing and take a transition with probability 1 into the new state t (indicated by a double circle). All the $(\neg a \wedge \neg b)$-states (i.e., u_1, u_2) are made absorbing and then (to simplify the figure) collapsed into one state u. In Fig. 2(b), the resulting weighted digraph is depicted where all the states remain the same, however the edge weights are obtained by taking the logarithm of the reciprocal of the corresponding transition probabilities.



(a) result of Step 1

(b) result of Step 2

Fig. 2. Transformation from DTMC to weighted digraph

A path σ from s to t in $\mathcal{G}$ is a sequence $\sigma = v_0 \cdot v_1 \cdot \ldots \cdot v_j \in V^+$, where $v_0 = s, v_j = t$ and $(v_i, v_{i+1}) \in E$, for $0 \leqslant i < |\sigma|$. As for paths in DTMCs, $|\sigma|$ denotes the length of σ. The *distance* of finite path $\sigma = v_0 \cdot v_1 \cdot \ldots \cdot v_j$ in graph $\mathcal{G}$ is $d(\sigma) = \sum_{i=0}^{j-1} w(v_i, v_{i+1})$. Due to the fact that multiplication of probabilities in $\mathcal{D}$ corresponds to addition of weights in $\mathcal{G}_D$, and that weights are based on taking the logarithm of the reciprocal of the transition probabilities in $\mathcal{D}$, distances in $\mathcal{G}$ and path-probabilities in DTMC $\mathcal{D}$ are related as follows.

Lemma 2. *Let σ and σ' be finite paths in DTMC $\mathcal{D}$ and its graph $\mathcal{G}_D$. Then:*

$$\Pr\{\sigma'\} \geqslant \Pr\{\sigma\} \quad \textit{iff} \quad d(\sigma') \leqslant d(\sigma).$$

Proof. Consider the following finite path σ in $\mathcal{D}$:

$$\sigma = s_0 \xrightarrow{\mathbf{P}(s_0,s_1)} s_1 \xrightarrow{\mathbf{P}(s_1,s_2)} s_2 \cdots s_{n-1} \xrightarrow{\mathbf{P}(s_{n-1},s_n)} s_n$$

By definition, its probability is:

$$\Pr\{\sigma\} = \mathbf{P}(s_0, s_1) \cdot \mathbf{P}(s_1, s_2) \cdots \mathbf{P}(s_{n-1}, s_n) = \prod_{i=0}^{n-1} \mathbf{P}(s_i, s_{i+1})$$

Consider an alternative path σ':

$$\sigma' = s'_0 \xrightarrow{\mathbf{P}(s_0,s'_1)} s'_1 \xrightarrow{\mathbf{P}(s'_1,s'_2)} s'_2 \cdots s'_{m-1} \xrightarrow{\mathbf{P}(s'_{m-1},s'_m)} s'_m$$

$$\Pr\{\sigma'\} = \mathbf{P}(s'_0, s'_1) \cdot \mathbf{P}(s'_1, s'_2) \cdots \mathbf{P}(s'_{m-1}, s'_m) = \prod_{i=0}^{m-1} \mathbf{P}(s'_i, s'_{i+1})$$

If $\Pr\{\sigma'\} \geqslant \Pr\{\sigma\}$, we have:

$$\begin{array}{cc}
 & \prod_{i=0}^{m-1} \mathbf{P}(s'_i, s'_{i+1}) \geqslant \prod_{i=0}^{n-1} \mathbf{P}(s_i, s_{i+1}) \\
\Longleftrightarrow & \dfrac{1}{\prod_{i=0}^{m-1} \mathbf{P}(s'_i, s'_{i+1})} \leqslant \dfrac{1}{\prod_{i=0}^{n-1} \mathbf{P}(s_i, s_{i+1})} \\
\Longleftrightarrow & \log(\dfrac{1}{\prod_{i=0}^{m-1} \mathbf{P}(s'_i, s'_{i+1})}) \leqslant \log(\dfrac{1}{\prod_{i=0}^{n-1} \mathbf{P}(s_i, s_{i+1})}) \\
\Longleftrightarrow & \sum_{i=0}^{m-1} \log(\dfrac{1}{\mathbf{P}(s'_i, s'_{i+1})}) \leqslant \sum_{i=0}^{n-1} \log(\dfrac{1}{\mathbf{P}(s_i, s_{i+1})}) \\
\Longleftrightarrow & \sum_{i=0}^{m-1} w(s'_i, s'_{i+1}) \leqslant \sum_{i=0}^{n-1} w(s_i, s_{i+1}) \\
\Longleftrightarrow & d(\sigma^*) \leqslant d(\sigma)
\end{array}$$

□

Note that Lemma 2 also holds for infinite paths. The following lemma specifies the correspondence between paths in DTMC D and its weighted digraph.

Lemma 3. *For any path σ from s to t in DTMC $\mathcal{D}$ and $k > 0$:*

1. *σ is a k-th most probable path in $\mathcal{D}$ iff σ is a k-th shortest path in $\mathcal{G}_D$;*
2. *σ is a k-th most probable path of at most h hops in $\mathcal{D}$ iff σ is a k-th shortest path of at most h hops in $\mathcal{G}_D$;*

The correspondence between path probabilities in the DTMC and distances in its weighted digraph constitutes the basis for the remaining algorithms.

5 Finding strongest evidences

This section considers algorithms for determining strongest evidences, i.e., maximally probable paths.

5.1 Unbounded until

Based on the results of Lemma 3.1 where $k = 1$, we consider the shortest path problem.

Definition 7 (SP problem). *Given a weighted digraph $\mathcal{G} = (V, E, w)$ and $s, t \in V$, the* shortest path *(SP) problem is to determine a path σ from s to t such that $d(\sigma) \leqslant d(\sigma')$ for any path σ' from s to t in $\mathcal{G}$.*

From Lemma 3.1 together with the transformation of a DTMC into a weighted digraph, it follows that there is a polynomial reduction from the SE problem for unbounded until to the SP problem. As the SP problem is known to be in PTIME, it follows:

Theorem 1. *The SE problem for unbounded until is in PTIME.*

Various efficient algorithms [13, 8, 15, 11] exist for the SP problem, e.g., when using Dijkstra's algorithm, the SE problem for unbounded until can be solved in time $\mathcal{O}(m + n \log n)$ when using appropriate data structures such as Fibonacci heaps.

5.2 Bounded until

Lemma 3.2 when $k = 1$ applies when considering maximally probable paths of a certain maximal hop count. This suggests to consider the hop-constrained shortest path problem.

Definition 8 (HSP problem). *Given a weighted digraph $\mathcal{G} = (V, E, w)$, $s, t \in V$ and $h \in \mathbb{N}$, the* hop-constrained SP *(HSP) problem is to determine a path σ in $\mathcal{G}$ from s to t with $|\sigma| \leqslant h$ such that $d(\sigma) \leqslant d(\sigma')$ for any path σ' from s to t with $|\sigma'| \leqslant h$.*

The HSP problem is a special case of the constrained shortest path (CSP) problem [27, 1], where the only constraint is the hop count.

Definition 9 (CSP problem). *Given a weighted digraph $\mathcal{G} = (V, E, w)$, $s, t \in V$ and resource constraints λ^i, for $1 \leqslant i \leqslant c$. Edge $e \in E$ uses $r^i(e) \geqslant 0$ units of resource i. The* (resource) constrained shortest path *problem (CSP) is to determine a shortest path σ in $\mathcal{G}$ from s to t such that $\sum_{e \in \sigma} r^i(e) \leqslant \lambda^i$ for $1 \leqslant i \leqslant c$.*

The CSP problem is NP-complete, even for a single resource constraint [1]. However, if each edge uses a constant unit of that resource (such as the hop count), the CSP problem can be solved in polynomial time, cf. [18], problem ND30. Thus:

Theorem 2. *The SE problem for bounded until is in PTIME.*

For $h \geqslant n{-}1$, it is possible to use Dijkstra's SP algorithm (as for unbounded until), as a shortest path does not contain cycles. If $h < n{-}1$, however, Dijkstra's algorithm does not guarantee to obtain a shortest path of at most h hops. We, therefore, adopt the Bellman-Ford (BF) algorithm [8, 15, 11] which fits well to our problem as it proceeds by increasing hop count. It can be readily modified to generate a shortest path within a given hop count. In the sequel of the paper, this algorithm is generalised for computing shortest counterexamples. The BF-algorithm is based on a set of recursive equations; we extend these with the hop count h. For $v \in V$, let $\pi(v, h)$ denote the shortest path from s to v of at most h hops (if it exists). Then:

$$\pi(v,h) = \begin{cases} s & \text{if } v = s \text{ and } h \geqslant 0; \quad (1a) \\ \bot & \text{if } v \neq s \text{ and } h = 0; \quad (1b) \\ \arg\min_u \{ d(\pi(u, h{-}1) \cdot v) \mid (u,v) \in E \} & \text{if } v \neq s \text{ and } h > 0. \quad (1c) \end{cases}$$

where $\bot$ denotes undefined. The last clause states that $\pi(v, h)$ consists of the shortest path to v's predecessor u, i.e., $\pi(u, h{-}1)$, extended with edge (u, v). Note that $\min_u\{d(\pi(u, h{-}1) \cdot v) \mid (u, v) \in E\}$ is the distance of the shortest path; by means of arg, the path is obtained. It follows (cf. [24]) that equation (1) characterizes the shortest path from s to v in at most h hops, and can be solved in time $\mathcal{O}(hm)$. As $h < n{-}1$, this is indeed in PTIME. Recall that for $h \geqslant n{-}1$, Dijkstra's algorithm has a favorable time complexity.

Remark 1. Note that the self-loop of vertex t is neglected when computing the hop-constrained shortest path using BF algorithm. This is because the logic operator is the bounded until $\mathcal{U}^{\leqslant h}$ instead of point interval until $\mathcal{U}^{=h}$, so that once it reaches t within the hop bound, the path formula holds and the self-loop does not change the path probability.

Example 5. To illustrate the BF algorithm, we compute the shortest path in at most 4 hops from s to t in our example in Fig.2(b), i.e., $\pi(t, 4)$. In order to compute $\pi(t, 4)$, three predecessors of t are considered so that $\pi(t_1, 3)$, $\pi(t_2, 3)$, $\pi(t_3, 3)$ are invoked. Again, to compute $\pi(t_1, 3)$, two predecessors of t_1 are considered so that $\pi(s_1, 2), \pi(s_4, 2)$ are invoked. In sequel, $\pi(s, 1)$ and $\pi(s_3, 1)$ are invoked for $\pi(s_1, 2)$, where $\pi(s, 1)$ is s, defined by equation (1a). $\pi(s_3, 1)$ is derived by invoking its two predecessors $\pi(s, 0)$, which is s (by (1a)) and $\pi(s_1, 0)$ which is $\bot$ (by (1b)). The computation is given in Fig. 3. Note that $\pi(s_4, 2), \pi(s_3, 1), \pi(s, 0), \pi(s_1, 0), \pi(s_3, 0)$ are indicated more than once (with more than one incoming edge), but are (like in dynamic programming) computed only once.

Remark 2. Alternatively, the *Viterbi algorithm* [16, 29] for probabilistic automata can be applied to our problem. The Viterbi algorithm determines the most probable path that generates a given trace. Let $\mathcal{D}'$ be a DTMC that is obtained after the first step described in Section 4, and suppose that $L'(s)$ contains the set of atomic propositions that are valid in s and all subformulae of the formula under consideration. (Note that these labels are known due to the recursive descent nature of the PCTL model checking algorithm.) Let $tr(\sigma)$ denote the projection of a path $\sigma = s_0{\cdot}s_1{\cdot}\ldots{\cdot}s_h$ on its trace γ, i.e., $\gamma = tr(\sigma) = L'(s_0)L'(s_1)\cdots L'(s_h)$. $\sigma{\downarrow_i}$ denotes the prefix of path σ truncated at length i (thus ending in s_i), formally, $\sigma{\downarrow_i} = \sigma[0]{\cdot}\sigma[1]{\cdot}\ldots{\cdot}\sigma[i]$. Thus, $tr(\sigma{\downarrow_i}) = L'(s_0)L'(s_1)\cdots L'(s_i)$. $\gamma{\downarrow_i}$ denotes

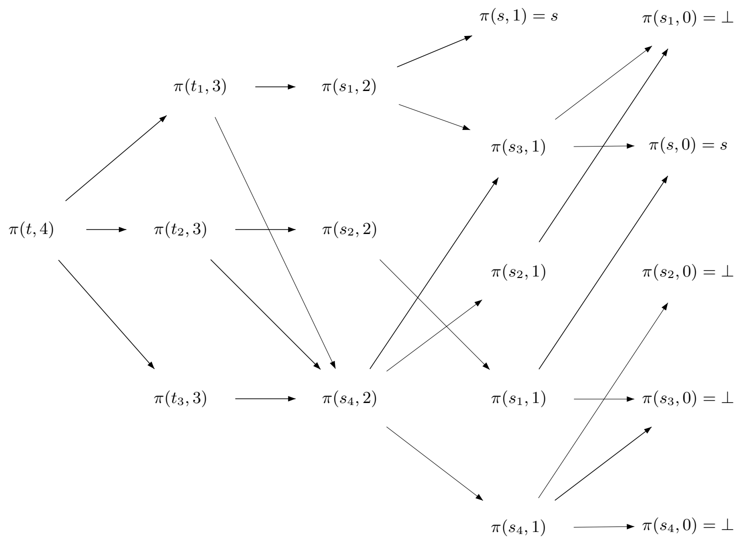


Fig. 3. An example run of the Bellman-Ford algorithm

the prefix of trace γ with length i. Let $\rho(\gamma, i, v)$ denote the probability of the most probable path $\sigma{\downarrow_i}$ whose trace equals $\gamma{\downarrow_i}$ and reaches state v. $\rho(\gamma, i, v)$ can be formally defined as follows:

$$\rho(\gamma, i, v) = \max_{tr(\sigma\downarrow_i)=\gamma_i} \prod_{j=0}^{i-1} \mathbf{P}(s_j, s_{j+1}) \cdot \mathbf{1}_v(s_i),$$

where $\mathbf{1}_v(s_i)$ is the characteristic function of v, i.e., $\mathbf{1}_v(s_i)$ returns 1, if $s_i = v$; 0, else.

The Viterbi algorithm provides an algorithmic solution to compute $\rho(\gamma, i, v)$:

$$\rho(\gamma, i, v) = \begin{cases} 1 & \text{if } s = v \text{ and } i = 0; \\ 0 & \text{if } s \neq v \text{ and } i = 0; \\ \max_{u \in S} \rho(\gamma, i-1, u) \cdot \mathbf{P}(u, v) & \text{otherwise.} \end{cases}$$

By computing $\rho(\Phi^h\Psi, h, s_h)$, the Viterbi algorithm determines the most probable h-hop path $\sigma = s_0{\cdot}s_1{\cdot}...{\cdot}s_h$ that generates the trace $\gamma = L'(s_0)L'(s_1)...L'(s_h)$ $= \Phi^h\Psi$ with length $(h{+}1)$. For our SE problem for bounded until, the trace of the most probable hop-constrained path from s to t is among $\{\Psi\, at_t, \Phi\Psi\, at_t, ..., \Phi^h\Psi\, at_t\}$. The self-loop at vertex t with probability 1 can make sure that all these paths have length $h{+}1$ but not change their probabilities, e.g., the path with trace $\Psi\, at_t$ can be extended so that the trace becomes $\Psi\, {at_t}^{h+1}$. We obtain the most probable path for $\Phi\mathcal{U}^{\leqslant h}\Psi$ by computing $\rho((\Phi{\vee}\Psi{\vee} at_t)^{h+1} at_t, h{+}1, t)$ using the Viterbi algorithm. The time complexity of the Viterbi algorithm is $\mathcal{O}(hm)$, as for the BF algorithm.

6 Finding smallest counterexamples

Recall that a smallest (most indicative) counterexample is a counterexample of minimal cardinality, whose probability deviates maximally from the required probability bound. In this section, we investigate algorithms for computing such smallest counterexamples. First observe that any smallest counterexample that contains, say k paths, contains the k most probable paths. This follows from the fact that any non-k most probable path can be exchanged with a more probable path, without changing the size of the counterexample, but by increasing its probability.

6.1 Unbounded until

Lemma 3.1 is applicable here. This suggests to consider the k shortest paths problem.

Definition 10 (KSP problem). *Given a weighted digraph $\mathcal{G} = (V, E, w)$, $s, t \in V$, and $k \in \mathbb{N}$, the k* shortest paths *(KSP) problem is to find k distinct shortest paths between s and t in G, if such paths exist.*

Theorem 3. *The SC problem for unbounded until is a KSP problem.*

Proof. We prove that a smallest counterexample of size k, contains k most probable paths. It is proven by contradiction. Let C be a smallest counterexample for ϕ with $|C| = k$, and assume C does not contain the k most probable paths satisfying ϕ. Then there is a path $\sigma \notin C$ satisfying ϕ such that $\Pr\{\sigma\} > \Pr\{\sigma'\}$ for some $\sigma' \in C$. Let $C' = C \setminus \{\sigma'\} \cup \{\sigma\}$. Then C' is a counterexample for ϕ, $|C| = |C'|$ and $\Pr(C) > \Pr(C')$. This contradicts C being a smallest counterexample. □

The question remains how to obtain k. Various algorithms for the KSP problem require k to be known a priori. This is inapplicable in our setting, as the number of paths in a shortest counterexample is implicitly provided by the probability bound in the PCTL-formula. We therefore consider algorithms that allow to determine k on the fly, i.e., that can halt at any k and resume if necessary. A good candidate is Eppstein's algorithm [14]. Although this algorithm has the best known asymptotic time complexity, viz. $\mathcal{O}(m+n \log n+k)$, in practice the recursive enumeration algorithm (REA) by Jiménez and Marzal [22] prevails. This algorithm has a time complexity in $\mathcal{O}(m+kn \log \frac{m}{n})$ and is based on a generalisation of the recursive equations for the BF-algorithm, and is readily adaptable to the case for bounded h, as we demonstrate below. Note that the time complexity of KSP algorithms depends on k, and as k may be exponential, their complexity is *pseudo*-polynomial.

6.2 Bounded until

Similar to the bounded until case for strongest evidences, we now consider the KSP problem where the length of paths is constrained, as Lemma 3.2 is applicable here.

Definition 11 (HKSP problem). *Given a weighted digraph $\mathcal{G} = (V, E, w)$, $s, t \in V$ and $h, k \in \mathbb{N}$, the* hop-constrained *KSP (HKSP) problem is to determine k shortest paths each of length at most h between s and t.*

Similar to Theorem 3 we obtain:

Theorem 4. *The SC problem for bounded until is a HKSP problem.*

To our knowledge, algorithms for the HKSP problem do not exist. In order to solve the HKSP problem, we propose a new algorithm that is strongly based on Jiménez and Marzal's REA algorithm [22]. The advantage of adapting this algorithm is that k can be determined on the fly, an essential characteristic for our setting. The algorithm is a conservative extension of the REA algorithm.

For $v \in V$, let $\pi^k(v,h)$ denote the k-th shortest path from s to v of length at most h (if it exists). As before, we use $\perp$ to denote the non-existence of a path. We establish the following equations:

$$\pi^k(v,h) = \begin{cases} s & \text{if } k=1, v=s \text{ and } h \geqslant 0 & (2a) \\ \perp & \text{if } (k>1, v=s, h=0) \text{ or } (v \neq s, h=0) & (2b) \\ \arg\min_\sigma \{d(\sigma) \mid \sigma \in Q^k(v,h)\} & \text{otherwise} & (2c) \end{cases}$$

where $Q^k(v,h)$ is a set of candidate paths among which $\pi^k(v,h)$ is chosen. The candidate sets are defined by:

$$Q^k(v,h) = \begin{cases} \{\pi^1(u,h{-}1)\cdot v \mid (u,v) \in E\} \\ \qquad\qquad \text{if } k=1, v \neq s \text{ or } k=2, v=s \\ (Q^{k-1}(v,h) - \{\pi^{k'}(u,h{-}1)\cdot v\}) \cup \{\pi^{k'+1}(u,h{-}1)\cdot v\} \\ \qquad\qquad \text{if } k>1 \text{ and } u,k' \text{ are the node and index,} \\ \qquad\qquad \text{such that } \pi^{k-1}(v,h) = \pi^{k'}(u,h{-}1)\cdot v \end{cases} \quad (3)$$

Assume that the path $\pi^{k'+1}(u,h{-}1)\cdot v{=}\perp$ if it does not exist, which happens when $Q^{k'+1}(u,h{-}1){=}\emptyset$. Note that $\perp\cdot v{=}\perp$ for any $v \in V$. $Q^k(v,h){=}\emptyset$ if it only contains $\perp$.

If $k{=}1$, the shortest path to v's predecessor u is extended with the edge to v. In the latter clause, $\pi^{k'}(u,h{-}1)$ denotes the selected $(k{-}1)$-st shortest path from s to u, where u is the direct predecessor of v. Paths in $Q^k(v,h)$ for $k>1$ are thus either candidate paths for $k{-}1$ where the selected path is eliminated (first summand) or the $(k'{+}1)$-st shortest path from s to u extended with edge (u,v) (second summand). Note that for the source state s, there is no need to define $Q^k(s,h)$ as $\pi^k(s,h)$ is defined by equations $(2a)$ and $(2b)$, which act as termination conditions.

Proposition 1. *The equations* $(2a)$-$(2c)$ *and* (3) *characterize the hop-constrained k shortest paths from s to v in at most h hops.*

Proof. Let $\mathcal{R}^k(v,h)$ denote the set of the k shortest paths from s to v in at most h hops. Each path in $\mathcal{R}^k(v,h)$ reaches v from some vertex $u \in \mathit{Pred}(v){=}\{w \in V \mid (w,v) \in E\}$. In order to compute $\pi^k(v,h)$, we should consider for every $u \in \mathit{Pred}(v)$, *all paths* from s to u that do not yield a path in $\mathcal{R}^{k-1}(v,h)$. However, since $k_1{<}k_2$ implies that $d(\pi^{k_1}(u,h{-}1)){+}w(u,v) \leqslant d(\pi^{k_2}(u,h{-}1)){+}w(u,v)$, only *the shortest* of these paths needs to be taken into account when computing $\pi^k(v,h)$. Thus we can associate to (v,h) a set of candidate paths $Q^k(v,h)$ among which $\pi^k(v,h)$ can be chosen, that contains at most one path for each predecessor $u \in \mathit{Pred}(v)$. This set Q^k is recursively defined by equation (3). $\square$

The adapted REA. The adapted REA for computing the k shortest paths from s to t which each consist of at most h hops is sketched as follows. The algorithm is based on the recursive equations given just above.

i Compute $\pi^1(t,h)$ by the BF algorithm and set $k := 1$.

ii Repeat until $\pi^k(t,h)$ does not exist or $\sum_{i=1}^{k} \Pr\{\pi^i(t,h)\} > p$:

(a) Set $k := k{+}1$ and compute $\pi^k(t,h)$ by invoking *NextPath*(v,h,k).

For $k{>}1$, and once $\pi^1(v,h), \ldots, \pi^{k-1}(v,h)$ are available, *NextPath*(t,h,k) computes $\pi^k(v,h)$ as follows:

1. If $h{\leqslant}0$, goto step 4.
2. If $k{=}2$, then set $Q[v,h] := \{\pi^1(u,h{-}1){\cdot}v \mid (u,v) \in E \text{ and } \pi^1(v,h) \neq \pi^1(u,h{-}1){\cdot}v\}$.
3. Let u and k' be the node and index such that $\pi^{k-1}(v,h) = \pi^{k'}(u,h{-}1){\cdot}v$.
   (a) If $\pi^{k'+1}(u,h{-}1)$ has not yet been computed, invoke *NextPath*$(u,h{-}1,k'{+}1)$.
   (b) If $\pi^{k'+1}(u,h{-}1)$ exists, then insert $\pi^{k'+1}(u,h{-}1){\cdot}v$ in $Q[v,h]$.
4. If $Q[v,h] \neq \emptyset$, then select and delete a path with minimum weight from $Q[v,h]$ and assign it to $\pi^k(v,h)$, else $\pi^k(v,h)$ does not exist.

In the main program, first the shortest path from s to t is determined using, e.g., the BF-algorithm. The intermediate results are recorded, e.g., all the paths in Fig. 3. Then, the k shortest paths are determined iteratively using the subroutine *NextPath*. The computation terminates when the k-th shortest path does not exist, or the total probability mass of the k shortest paths so far exceeds the bound p. Recall that p is the lower bound of the PCTL formula to be checked. Note that $Q[v,h]$ in the algorithm corresponds to $Q^k(v,h)$, where k is the parameter of the program. In steps 2 through 3, the set $Q^k(v,h)$ is determined from $Q^{k-1}(v,h)$ according to equation (3). In the final step, $\pi^k(v,h)$ is selected from $Q^k(v,h)$ according to equation (2c).

Example 6. We illustrate how the algorithm progresses by computing 3 shortest paths in Fig. 2 (b) with hop count $h = 4$, i.e., $\pi^1(t,4)$, $\pi^2(t,4)$ and $\pi^3(t,4)$.

By BF algorithm, we compute $\pi^1(t,4) = s{\cdot}s_1{\cdot}s_2{\cdot}t_2{\cdot}t$, several shortest paths to different destinations with different hop constraints are derived as a by-product as we showed in Example 5, where $\pi(v,h)$ in the previous example is $\pi^1(v,h)$ here. We summarize the paths below:

$h = 3$	$h = 2$	$h = 1$	$h = 0$
		$\pi^1(s,1) = s{\cdot}s$	$\pi^1(s,0) = s$
$\pi^1(t_1,3) = s{\cdot}s_1{\cdot}t_1$	$\pi^1(s_1,2) = s{\cdot}s_1$	$\pi^1(s_1,1) = s{\cdot}s_1$	$\pi^1(s_1,0) = \perp$
$\pi^1(t_2,3) = s{\cdot}s_1{\cdot}s_2{\cdot}t_2$	$\pi^1(s_2,2) = s{\cdot}s_1{\cdot}s_2$	$\pi^1(s_2,1) = \perp$	$\pi^1(s_2,0) = \perp$
$\pi^1(t_3,3) = s{\cdot}s_3{\cdot}s_4{\cdot}t_3$	$\pi^1(s_4,2) = s{\cdot}s_3{\cdot}s_4$	$\pi^1(s_3,1) = s{\cdot}s_3$	$\pi^1(s_3,0) = \perp$
		$\pi^1(s_4,1) = \perp$	$\pi^1(s_4,0) = \perp$

The algorithm for $\pi^2(t,4)$ is running in Fig. 4.

Note that in order to compute $\pi^2(t,4)$, the vertices along $\pi^1(t,4) = s{\cdot}s_1{\cdot}s_2{\cdot}t_2{\cdot}t$ are computed successively, i.e., $\pi^2(t_2,3)$, $\pi^2(s_2,2)$, $\pi^2(s_1,1)$ and $\pi^2(s,0)$, by recursively invoking *NextPath*. The hop count of each invocation is decreased by 1, so that *NextPath* is invoked 5 times when $h = 4$ and $k = 2$. Similarly, to compute $\pi^3(t,4)$, the vertices along $\pi^2(t,4) = s{\cdot}s_3{\cdot}s_4{\cdot}t_1{\cdot}t$ ($\pi^2(t_1,3)$, $\pi^2(s_4,2)$, $\pi^2(s_3,1)$ and $\pi^2(s,0)$) are needed. Actually, in this particular example, $\pi^3(t,4)$ can be easily derived, since in Step 3$(b)^*$, $d(\pi^1(t_1,3){\cdot}t) = d(\pi^2(t_3,3){\cdot}t)$, then $\pi^3(t,4)$ definitely equals $\pi^2(t_3,3){\cdot}t$.

$k = 2$, invoke $\mathit{NextPath}(t, 4, 2)$:
  2. set $Q[t, 4] = \{\pi^1(t_1, 3)\cdot t,\ \pi^1(t_3, 3)\cdot t\}$. /*$t_1, t_2, t_3$ are predecessors of t*/
  3. $u = t_2$, $k' = 1$, $h = 3$. /* $\pi^1(t, 4) = \pi^1(t_2, 3)\cdot t = s\cdot s_1\cdot s_2\cdot t_2\cdot t$*/
  3(a). compute $\pi^2(t_2, 3)$ by invoking $\mathit{NextPath}(t_2, 3, 2)$:
    2. ($k = 2$) set $Q[t_2, 3] = \{\pi^1(s_4, 2)\cdot t_2\}$.
    3. $u = s_2$, $k' = 1$, $h = 2$. /* $\pi^1(t_2, 3) = \pi^1(s_2, 2)\cdot t_2 = s\cdot s_1\cdot s_2\cdot t_2$ */
    3(a). compute $\pi^2(s_2, 2)$ by invoking $\mathit{NextPath}(s_2, 2, 2)$:
      2. ($k = 2$) set $Q[s_2, 2] = \emptyset$.
      3. $u = s_1$, $k' = 1$, $h = 1$. /* $\pi^1(s_2, 2) = \pi^1(s_1, 1)\cdot s_2 = s\cdot s_1\cdot s_2$ */
      3(a). compute $\pi^2(s_1, 1)$ by invoking $\mathit{NextPath}(s_1, 1, 2)$:
        2 ($k = 2$) set $Q[s_1, 1] = \emptyset$.
        3. $u = s$, $k' = 1$, $h = 0$. /* $\pi^1(s_1, 1) = \pi^1(s, 0)\cdot s_1 = s\cdot s_1$ */
        3(a). compute $\pi^2(s, 0)$ by invoking $\mathit{NextPath}(s, 0, 2)$:
          1. Goto 4.
          4. $\pi^2(s, 0) = \perp$.
        3(b). $Q[s_1, 1] = \emptyset$.
        4. $\pi^2(s_1, 1) = \perp$.
      3(b). $Q[s_2, 2] = \emptyset$.
      4. $\pi^2(s_2, 2) = \perp$.
    3(b). $Q[t_3, 3] = \{\pi^1(s_4, 2)\cdot t_2\}$.
    4. $\pi^2(t_2, 3) = s\cdot s_3\cdot s_4\cdot t_2$.
  3(b)*. $Q[t, 4] = \{\pi^1(t_1, 3)\cdot t,\ \pi^1(t_3, 3)\cdot t,\ \pi^2(t_3, 3)\cdot t\}$. /*$d(\pi^1(t_1, 3)\cdot t) = d(\pi^2(t_3, 3)\cdot t)$*/
  4. $\pi^2(t, 4) = \pi^1(t_1, 3)\cdot t = s\cdot s_3\cdot s_4\cdot t_1\cdot t$. /* $\pi^1(t_1, 3)\cdot t$ is the shortest in $Q[t, 4]$ */

Fig. 4. An example of the adapted REA

Time complexity. Before we analyze the algorithm time complexity, we first prove that the recursive calls to *NextPath* to compute $\pi^k(t, h)$ visit, in the worst case, all the vertices in $\pi^{k-1}(t, h)$, which is at most h.

Lemma 4. *For $k{>}1$ and for all $v \in V$, the computation of $\pi^k(v, h)$ by means of $\mathit{NextPath}(v, h, k)$ may recursively generate calls to $\mathit{NextPath}(u, h{-}1, j)$ only for vertices u in $\pi^{k-1}(v, h)$.*

Proof. Suppose $\pi^{k-1}(v, h) = u_1\cdot u_2\cdot\ldots\cdot u_p$, where $u_1{=}s$ and $u_p{=}t$. For every $i = 1, ..., p$, let k_i be the index such that $\pi^{k_i}(u_i) = u_1\cdot u_2\cdot\ldots\cdot u_i$. Since $\pi^{k-1}(v, h) = \pi^{k_{p-1}}(u_{p-1}, h{-}1)\cdot v$, $\mathit{NextPath}(v, h, k)$ may require a recursive call to $\mathit{NextPath}(u_{p-1}, h{-}1, k_{p-1}{+}1)$ in case $\pi^{k_{p-1}+1}(u_{p-1}, h{-}1)$ has not been already computed; since $\pi^{k_{p-1}}(u_{p-1}, h{-}1) = \pi^{k_{p-2}}(u_{p-2}, h{-}2)\cdot u_{p-1}$, $\mathit{NextPath}(u_{p-1}, h{-}1, k_{p-1} + 1)$ may require a recursive call to $\mathit{NextPath}(u_{p-2}, h{-}2, k_{p-2} + 1)$; and so on. In the worst case, the recursive calls extend through the nodes $u_p, u_{p-1}, ..., u_1$. If the recursion reaches $\pi^1(s, h')(0{\leqslant}h'{\leqslant}h)$ or $\pi^{k'}(v, 0)(k'{>}0, v{\neq}s)$ or $\pi^{k''}(s, 0)(k'' > 1)$ so that the termination conditions in equation ($2a$) and ($2b$) or Algorithm Step 1 hold, then no more recursive calls are performed. □

To determine the computational complexity of the algorithm, we assume the candidate sets to be implemented by heaps (as in [22]). The k shortest paths to a vertex v can be stored in a linked list, where each path $\pi^k(v, h) = \pi^{k'}(u)\cdot v$ is compactly represented by its length and a back pointer to $\pi^{k'}(u)$. Using these data structures, we obtain:

Proposition 2. *The time complexity of the adapted REA is $\mathcal{O}(hm{+}hk\log(\frac{m}{n}))$.*

Proof. The computation of the first step takes $\mathcal{O}(hm)$ using the BF-algorithm. Due to Lemma 4, the number of recursive invocations to *NextPath* is bounded by

h, the maximum length of $\pi^{k-1}(t,h)$. At any given time, the set $Q^k(v,h)$ contains at most $|Pred(v)|$ paths where $Pred(v) = \{u \mid (u,v) \in E\}$, i.e., one path for each predecessor vertex of v. By using heaps to store the candidate sets, a minimal element can be determined and deleted (cf. Step4) in $\mathcal{O}(\log |Pred(v)|)$ time. Insertion of a path (as in Steps 2 and 3(b)) takes the same time complexity. Since $\sum_{v \in V} |Pred(v)| = m$, $\sum_{v \in V} \log |Pred(v)|$ is maximized when all vertices have an equal number of predecessors, i.e., $|Pred(v)| = \frac{m}{n}$. Hence, it takes $\mathcal{O}(h \log(\frac{m}{n}))$ to compute $\pi^k(v,h)$. We have k such paths to compute, yielding $\mathcal{O}(hm + hk \log(\frac{m}{n}))$.

Note that the time complexity is pseudo-polynomial due to the dependence on k which may be exponential in n. As in our setting, k is not known in advance, this can not be reduced to a polynomial time complexity.

7 Conclusion

Summary of results. We have investigated the computation of strongest evidences (maximally probable paths) and smallest counterexamples for PCTL model checking of DTMCs. Relationships to various kinds of shortest path problems have been established. Summarizing we have obtained:

counterexample problem	shortest path problem	time complexity
SE (until)	SP	$\mathcal{O}(m + n \log n)$
SE (bounded until)	HSP	$\mathcal{O}(hm)$
SC (until)	KSP	$\mathcal{O}(m + n \log n + k)$
SC (bounded until)	HKSP	$\mathcal{O}(hm + hk \log(\frac{m}{n}))$

where n and m are the number of states and transitions, h is the hop bound, and k is the number of shortest paths.

For DTMCs with rewards, we can establish along the same lines as in this paper that determining strongest evidences for violating reward- and hop-bounded until-formulae boils down to solving a non-trivial instance of the CSP problem. As this problem is NP complete, efficient algorithms for finding counterexamples for PRCTL [4] will be hard to obtain.

Further research. Topics for further research are: experimental research of the proposed algorithms in probabilistic model checking, considering loopless paths (see e.g., [21, 25, 19]), and extension towards continuous-time models.

Related work. With the notable exception of [2, 3], counterexample generation for probabilistic model checking has not been addressed before. Aljazzar *et al.* [2] consider the generation of a most probable path for timed reachability in CTMCs. They map this onto a bounded-until problem on DTMCs, and use heuristics (Z*) for determining the most probable path. Unbounded until is not considered, and neither a correctness proof nor complexity results are provided. Recently, [3] generalises this heuristic-based approach for CTMCs to obtain failure subgraphs, i.e., counterexamples. To our knowledge, smallest counterexamples have not been considered yet.

Acknowledgement. This research has been performed as part of the QUPES project that is financed by the Netherlands Organization for Scientific Research (NWO). David N. Jansen is kindly acknowledged for remarks on an earlier version of this paper.

References

1. R.K. Ahuja, T.L. Magnanti and J.B. Orlin. *Network Flows: Theory, Algorithms and Applications*, Prentice Hall, Inc., 1993.
2. H. Aljazzar, H. Hermanns and S. Leue. Counterexamples for timed probabilistic reachability. FORMATS 2005, LNCS 3829: 177-195, 2005.
3. H. Aljazzar and S. Leue. Extended directed search for probabilistic timed reachability. FORMATS 2006. (to appear)
4. S. Andova, H. Hermanns and J.-P. Katoen. Discrete-time rewards model-checked. FORMATS 2003, LNCS 2791: 88-104, 2003.
5. C. Baier, J.-P. Katoen, H. Hermanns and V. Wolf. Comparative branching-time semantics for Markov chains. *Inf. Comput.* 200(2): 149-214 (2005).
6. T. Ball, M. Naik and S. K. Rajamani. From symptom to cause: localizing errors in counterexample traces. POPL: 97-105, 2003.
7. G. Behrmann, K. G. Larsen and J. I. Rasmussen. Optimal scheduling using priced timed automata. *ACM SIGMETRICS Perf. Ev. Review* 32(4): 34-40 (2005).
8. R. Bellman. On a routing problem. *Quarterly of Appl. Math.*, 16(1): 87-90 (1958).
9. E.M. Clarke, O. Grumberg, S. Jha, Y. Lu and H. Veith: Counterexample-guided abstraction refinement. CAV, LNCS 1855: 154-169, 2000.
10. E.M. Clarke, S. Jha, Y. Lu and H. Veith. Tree-like counterexamples in model checking. LICS: 19-29 (2002).
11. T.H. Cormen, C.E. Leiserson, R.L. Rivest and C. Stein. *Introduction to Algorithms*, 2001. Section 24.1: The Bellman-Ford algorithm, pp.588-592.
12. L. de Alfaro, T.A. Henzinger and F. Mang. Detecting errors before reaching them. CAV, LNCS 2725: 186-201, 2000.
13. E.W. Dijkstra. A note on two problems in connection with graphs. *Num. Math.*, 1:395-412 (1959).
14. D. Eppstein. Finding the k shortest paths. *SIAM J. Comput.* 28(2): 652-673 (1998).
15. L.R. Ford jr. and D.R. Fulkerson. *Flows in Networks*, Princeton Univ. Press, 1962.
16. G.D. Forney. The Viterbi algorithm. *Proc. of the IEEE* 61(3): 268-278 (1973).
17. B. L. Fox. k-th shortest paths and applications to the probabilistic networks. In *ORSA/TIMS National Mtg*, volume 23, page B263. *Bull. Operations Research Soc. of America*, 1975.
18. M.R. Garey and D.S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness*, Freeman, San Francisco, 1979.
19. E. Hadjiconstantinou and N. Christofides. An efficient implementation of an algorithm for finding K shortest simple paths. *Networks* 34(2): 88-101 (1999).
20. H. Hansson and B. Jonsson. A logic for reasoning about time and reliability. *Formal Asp. Comput.* 6(5): 512-535 (1994).
21. J. Hershberger, M. Maxel and S. Suri. Finding the k shortest simple paths: A new algorithm and its implementation. ALENEX 2003, in *Proc. of the Fifth Workshop on Algorithm Engineering and Experiments*, Baltimore, USA, pp. 26-36, 2003.
22. V.M. Jiménez and A. Marzal. Computing the K shortest paths: A new algorithm and an experimental comparison. WAE 1999, LNCS 1668: 15-29, 1999.
23. H. Jin, K. Ravi and F. Somenzi. Fate and free will in error traces. *STTT* 6(2): 102-116 (2004).
24. E.L. Lawler. *Combinatorial Optimization: Networks and Matroids.* Holt, Reinhart, and Winston, 1976.
25. E.Q.V. Martins and M.M.B. Pascoal. A new implementation of Yen's ranking loopless paths algorithm. *4OR* 1(2): 121-133 (2003).
26. E.Q.V. Martins, M.M.B. Pascoal and J.L.E. Dos Santos. Deviation algorithms for ranking shortest paths. *Int. J. Found. Comput. Sci.* 10(3): 247-262 (1999).
27. K. Mehlhorn and M. Ziegelmann. Resource constrained shortest paths. ESA 2000, LNCS 1879: 326-337, 2000.
28. S. Shoham and O. Grumberg. A game-based framework for CTL counterexamples and 3-valued abstraction-refinement. CAV, LNCS 2725: 275-287, 2003.
29. E. Vidal, F. Thollard, C. de la Higuera, F. Casacuberta and R.C. Carrasco. Probabilistic finite-state machines-Part I. *IEEE Trans. Pattern Anal. Mach. Intell.* 27(7): 1013-1025 (2005).

Aachener Informatik-Berichte

This is a list of recent technical reports. To obtain copies of technical reports please consult http://aib.informatik.rwth-aachen.de/ or send your request to: Informatik-Bibliothek, RWTH Aachen, Ahornstr. 55, 52056 Aachen, Email: `biblio@informatik.rwth-aachen.de`

1987-01 *	Fachgruppe Informatik: Jahresbericht 1986
1987-02 *	David de Frutos Escrig, Klaus Indermark: Equivalence Relations of Non-Deterministic Ianov-Schemes
1987-03 *	Manfred Nagl: A Software Development Environment based on Graph Technology
1987-04 *	Claus Lewerentz, Manfred Nagl, Bernhard Westfechtel: On Integration Mechanisms within a Graph-Based Software Development Environment
1987-05 *	Reinhard Rinn: Über Eingabeanomalien bei verschiedenen Inferenzmodellen
1987-06 *	Werner Damm, Gert Döhmen: Specifying Distributed Computer Architectures in AADL*
1987-07 *	Gregor Engels, Claus Lewerentz, Wilhelm Schäfer: Graph Grammar Engineering: A Software Specification Method
1987-08 *	Manfred Nagl: Set Theoretic Approaches to Graph Grammars
1987-09 *	Claus Lewerentz, Andreas Schürr: Experiences with a Database System for Software Documents
1987-10 *	Herbert Klaeren, Klaus Indermark: A New Implementation Technique for Recursive Function Definitions
1987-11 *	Rita Loogen: Design of a Parallel Programmable Graph Reduction Machine with Distributed Memory
1987-12	J. Börstler, U. Möncke, R. Wilhelm: Table compression for tree automata
1988-01 *	Gabriele Esser, Johannes Rückert, Frank Wagner Gesellschaftliche Aspekte der Informatik
1988-02 *	Peter Martini, Otto Spaniol: Token-Passing in High-Speed Backbone Networks for Campus-Wide Environments
1988-03 *	Thomas Welzel: Simulation of a Multiple Token Ring Backbone
1988-04 *	Peter Martini: Performance Comparison for HSLAN Media Access Protocols
1988-05 *	Peter Martini: Performance Analysis of Multiple Token Rings
1988-06 *	Andreas Mann, Johannes Rückert, Otto Spaniol: Datenfunknetze
1988-07 *	Andreas Mann, Johannes Rückert: Packet Radio Networks for Data Exchange
1988-08 *	Andreas Mann, Johannes Rückert: Concurrent Slot Assignment Protocol for Packet Radio Networks
1988-09 *	W. Kremer, F. Reichert, J. Rückert, A. Mann: Entwurf einer Netzwerktopologie für ein Mobilfunknetz zur Unterstützung des öffentlichen Straßenverkehrs
1988-10 *	Kai Jakobs: Towards User-Friendly Networking
1988-11 *	Kai Jakobs: The Directory - Evolution of a Standard
1988-12 *	Kai Jakobs: Directory Services in Distributed Systems - A Survey
1988-13 *	Martine Schümmer: RS-511, a Protocol for the Plant Floor

1988-14 *	U. Quernheim: Satellite Communication Protocols - A Performance Comparison Considering On-Board Processing
1988-15 *	Peter Martini, Otto Spaniol, Thomas Welzel: File Transfer in High Speed Token Ring Networks: Performance Evaluation by Approximate Analysis and Simulation
1988-16 *	Fachgruppe Informatik: Jahresbericht 1987
1988-17 *	Wolfgang Thomas: Automata on Infinite Objects
1988-18 *	Michael Sonnenschein: On Petri Nets and Data Flow Graphs
1988-19 *	Heiko Vogler: Functional Distribution of the Contextual Analysis in Block-Structured Programming Languages: A Case Study of Tree Transducers
1988-20 *	Thomas Welzel: Einsatz des Simulationswerkzeuges QNAP2 zur Leistungsbewertung von Kommunikationsprotokollen
1988-21 *	Th. Janning, C. Lewerentz: Integrated Project Team Management in a Software Development Environment
1988-22 *	Joost Engelfriet, Heiko Vogler: Modular Tree Transducers
1988-23 *	Wolfgang Thomas: Automata and Quantifier Hierarchies
1988-24 *	Uschi Heuter: Generalized Definite Tree Languages
1989-01 *	Fachgruppe Informatik: Jahresbericht 1988
1989-02 *	G. Esser, J. Rückert, F. Wagner (Hrsg.): Gesellschaftliche Aspekte der Informatik
1989-03 *	Heiko Vogler: Bottom-Up Computation of Primitive Recursive Tree Functions
1989-04 *	Andy Schürr: Introduction to PROGRESS, an Attribute Graph Grammar Based Specification Language
1989-05	J. Börstler: Reuse and Software Development - Problems, Solutions, and Bibliography (in German)
1989-06 *	Kai Jakobs: OSI - An Appropriate Basis for Group Communication?
1989-07 *	Kai Jakobs: ISO's Directory Proposal - Evolution, Current Status and Future Problems
1989-08 *	Bernhard Westfechtel: Extension of a Graph Storage for Software Documents with Primitives for Undo/Redo and Revision Control
1989-09 *	Peter Martini: High Speed Local Area Networks - A Tutorial
1989-10 *	P. Davids, Th. Welzel: Performance Analysis of DQDB Based on Simulation
1989-11 *	Manfred Nagl (Ed.): Abstracts of Talks presented at the WG '89 15th International Workshop on Graphtheoretic Concepts in Computer Science
1989-12 *	Peter Martini: The DQDB Protocol - Is it Playing the Game?
1989-13 *	Martine Schümmer: CNC/DNC Communication with MAP
1989-14 *	Martine Schümmer: Local Area Networks for Manufactoring Environments with hard Real-Time Requirements
1989-15 *	M. Schümmer, Th. Welzel, P. Martini: Integration of Field Bus and MAP Networks - Hierarchical Communication Systems in Production Environments
1989-16 *	G. Vossen, K.-U. Witt: SUXESS: Towards a Sound Unification of Extensions of the Relational Data Model

1989-17 * J. Derissen, P. Hruschka, M.v.d. Beeck, Th. Janning, M. Nagl: Integrating Structured Analysis and Information Modelling
1989-18 A. Maassen: Programming with Higher Order Functions
1989-19 * Mario Rodriguez-Artalejo, Heiko Vogler: A Narrowing Machine for Syntax Directed BABEL
1989-20 H. Kuchen, R. Loogen, J.J. Moreno Navarro, M. Rodriguez Artalejo: Graph-based Implementation of a Functional Logic Language
1990-01 * Fachgruppe Informatik: Jahresbericht 1989
1990-02 * Vera Jansen, Andreas Potthoff, Wolfgang Thomas, Udo Wermuth: A Short Guide to the AMORE System (Computing Automata, MOnoids and Regular Expressions)
1990-03 * Jerzy Skurczynski: On Three Hierarchies of Weak SkS Formulas
1990-04 R. Loogen: Stack-based Implementation of Narrowing
1990-05 H. Kuchen, A. Wagener: Comparison of Dynamic Load Balancing Strategies
1990-06 * Kai Jakobs, Frank Reichert: Directory Services for Mobile Communication
1990-07 * Kai Jakobs: What's Beyond the Interface - OSI Networks to Support Cooperative Work
1990-08 * Kai Jakobs: Directory Names and Schema - An Evaluation
1990-09 * Ulrich Quernheim, Dieter Kreuer: Das CCITT - Signalisierungssystem Nr. 7 auf Satellitenstrecken; Simulation der Zeichengabestrecke
1990-11 H. Kuchen, R. Loogen, J.J. Moreno Navarro, M. Rodriguez Artalejo: Lazy Narrowing in a Graph Machine
1990-12 * Kai Jakobs, Josef Kaltwasser, Frank Reichert, Otto Spaniol: Der Computer fährt mit
1990-13 * Rudolf Mathar, Andreas Mann: Analyzing a Distributed Slot Assignment Protocol by Markov Chains
1990-14 A. Maassen: Compilerentwicklung in Miranda - ein Praktikum in funktionaler Programmierung (written in german)
1990-15 * Manfred Nagl, Andreas Schürr: A Specification Environment for Graph Grammars
1990-16 A. Schürr: PROGRESS: A VHL-Language Based on Graph Grammars
1990-17 * Marita Möller: Ein Ebenenmodell wissensbasierter Konsultationen - Unterstützung für Wissensakquisition und Erklärungsfähigkeit
1990-18 * Eric Kowalewski: Entwurf und Interpretation einer Sprache zur Beschreibung von Konsultationsphasen in Expertensystemen
1990-20 Y. Ortega Mallen, D. de Frutos Escrig: A Complete Proof System for Timed Observations
1990-21 * Manfred Nagl: Modelling of Software Architectures: Importance, Notions, Experiences
1990-22 H. Fassbender, H. Vogler: A Call-by-need Implementation of Syntax Directed Functional Programming
1991-01 Guenther Geiler (ed.), Fachgruppe Informatik: Jahresbericht 1990
1991-03 B. Steffen, A. Ingolfsdottir: Characteristic Formulae for Processes with Divergence
1991-04 M. Portz: A new class of cryptosystems based on interconnection networks

1991-05	H. Kuchen, G. Geiler: Distributed Applicative Arrays
1991-06 *	Ludwig Staiger: Kolmogorov Complexity and Hausdorff Dimension
1991-07 *	Ludwig Staiger: Syntactic Congruences for w-languages
1991-09 *	Eila Kuikka: A Proposal for a Syntax-Directed Text Processing System
1991-10	K. Gladitz, H. Fassbender, H. Vogler: Compiler-based Implementation of Syntax-Directed Functional Programming
1991-11	R. Loogen, St. Winkler: Dynamic Detection of Determinism in Functional Logic Languages
1991-12 *	K. Indermark, M. Rodriguez Artalejo (Eds.): Granada Workshop on the Integration of Functional and Logic Programming
1991-13 *	Rolf Hager, Wolfgang Kremer: The Adaptive Priority Scheduler: A More Fair Priority Service Discipline
1991-14 *	Andreas Fasbender, Wolfgang Kremer: A New Approximation Algorithm for Tandem Networks with Priority Nodes
1991-15	J. Börstler, A. Zündorf: Revisiting extensions to Modula-2 to support reusability
1991-16	J. Börstler, Th. Janning: Bridging the gap between Requirements Analysis and Design
1991-17	A. Zündorf, A. Schürr: Nondeterministic Control Structures for Graph Rewriting Systems
1991-18 *	Matthias Jarke, John Mylopoulos, Joachim W. Schmidt, Yannis Vassiliou: DAIDA: An Environment for Evolving Information Systems
1991-19	M. Jeusfeld, M. Jarke: From Relational to Object-Oriented Integrity Simplification
1991-20	G. Hogen, A. Kindler, R. Loogen: Automatic Parallelization of Lazy Functional Programs
1991-21 *	Prof. Dr. rer. nat. Otto Spaniol: ODP (Open Distributed Processing): Yet another Viewpoint
1991-22	H. Kuchen, F. Lücking, H. Stoltze: The Topology Description Language TDL
1991-23	S. Graf, B. Steffen: Compositional Minimization of Finite State Systems
1991-24	R. Cleaveland, J. Parrow, B. Steffen: The Concurrency Workbench: A Semantics Based Tool for the Verification of Concurrent Systems
1991-25 *	Rudolf Mathar, Jürgen Mattfeldt: Optimal Transmission Ranges for Mobile Communication in Linear Multihop Packet Radio Networks
1991-26	M. Jeusfeld, M. Staudt: Query Optimization in Deductive Object Bases
1991-27	J. Knoop, B. Steffen: The Interprocedural Coincidence Theorem
1991-28	J. Knoop, B. Steffen: Unifying Strength Reduction and Semantic Code Motion
1991-30	T. Margaria: First-Order theories for the verification of complex FSMs
1991-31	B. Steffen: Generating Data Flow Analysis Algorithms from Modal Specifications
1992-01	Stefan Eherer (ed.), Fachgruppe Informatik: Jahresbericht 1991
1992-02 *	Bernhard Westfechtel: Basismechanismen zur Datenverwaltung in strukturbezogenen Hypertextsystemen
1992-04	S. A. Smolka, B. Steffen: Priority as Extremal Probability
1992-05 *	Matthias Jarke, Carlos Maltzahn, Thomas Rose: Sharing Processes: Team Coordination in Design Repositories

1992-06	O. Burkart, B. Steffen: Model Checking for Context-Free Processes
1992-07 *	Matthias Jarke, Klaus Pohl: Information Systems Quality and Quality Information Systems
1992-08 *	Rudolf Mathar, Jürgen Mattfeldt: Analyzing Routing Strategy NFP in Multihop Packet Radio Networks on a Line
1992-09 *	Alfons Kemper, Guido Moerkotte: Grundlagen objektorientierter Datenbanksysteme
1992-10	Matthias Jarke, Manfred Jeusfeld, Andreas Miethsam, Michael Gocek: Towards a logic-based reconstruction of software configuration management
1992-11	Werner Hans: A Complete Indexing Scheme for WAM-based Abstract Machines
1992-12	W. Hans, R. Loogen, St. Winkler: On the Interaction of Lazy Evaluation and Backtracking
1992-13 *	Matthias Jarke, Thomas Rose: Specification Management with CAD
1992-14	Th. Noll, H. Vogler: Top-down Parsing with Simultaneous Evaluation on Noncircular Attribute Grammars
1992-15	A. Schuerr, B. Westfechtel: Graphgrammatiken und Graphersetzungssysteme(written in german)
1992-16 *	Graduiertenkolleg Informatik und Technik (Hrsg.): Forschungsprojekte des Graduiertenkollegs Informatik und Technik
1992-17	M. Jarke (ed.): ConceptBase V3.1 User Manual
1992-18 *	Clarence A. Ellis, Matthias Jarke (Eds.): Distributed Cooperation in Integrated Information Systems - Proceedings of the Third International Workshop on Intelligent and Cooperative Information Systems
1992-19-00	H. Kuchen, R. Loogen (eds.): Proceedings of the 4th Int. Workshop on the Parallel Implementation of Functional Languages
1992-19-01	G. Hogen, R. Loogen: PASTEL - A Parallel Stack-Based Implementation of Eager Functional Programs with Lazy Data Structures (Extended Abstract)
1992-19-02	H. Kuchen, K. Gladitz: Implementing Bags on a Shared Memory MIMD-Machine
1992-19-03	C. Rathsack, S.B. Scholz: LISA - A Lazy Interpreter for a Full-Fledged Lambda-Calculus
1992-19-04	T.A. Bratvold: Determining Useful Parallelism in Higher Order Functions
1992-19-05	S. Kahrs: Polymorphic Type Checking by Interpretation of Code
1992-19-06	M. Chakravarty, M. Köhler: Equational Constraints, Residuation, and the Parallel JUMP-Machine
1992-19-07	J. Seward: Polymorphic Strictness Analysis using Frontiers (Draft Version)
1992-19-08	D. Gärtner, A. Kimms, W. Kluge: pi-Redˆ+ - A Compiling Graph-Reduction System for a Full Fledged Lambda-Calculus
1992-19-09	D. Howe, G. Burn: Experiments with strict STG code
1992-19-10	J. Glauert: Parallel Implementation of Functional Languages Using Small Processes
1992-19-11	M. Joy, T. Axford: A Parallel Graph Reduction Machine
1992-19-12	A. Bennett, P. Kelly: Simulation of Multicache Parallel Reduction

1992-19-13	K. Langendoen, D.J. Agterkamp: Cache Behaviour of Lazy Functional Programs (Working Paper)
1992-19-14	K. Hammond, S. Peyton Jones: Profiling scheduling strategies on the GRIP parallel reducer
1992-19-15	S. Mintchev: Using Strictness Information in the STG-machine
1992-19-16	D. Rushall: An Attribute Grammar Evaluator in Haskell
1992-19-17	J. Wild, H. Glaser, P. Hartel: Statistics on storage management in a lazy functional language implementation
1992-19-18	W.S. Martins: Parallel Implementations of Functional Languages
1992-19-19	D. Lester: Distributed Garbage Collection of Cyclic Structures (Draft version)
1992-19-20	J.C. Glas, R.F.H. Hofman, W.G. Vree: Parallelization of Branch-and-Bound Algorithms in a Functional Programming Environment
1992-19-21	S. Hwang, D. Rushall: The nu-STG machine: a parallelized Spineless Tagless Graph Reduction Machine in a distributed memory architecture (Draft version)
1992-19-22	G. Burn, D. Le Metayer: Cps-Translation and the Correctness of Optimising Compilers
1992-19-23	S.L. Peyton Jones, P. Wadler: Imperative functional programming (Brief summary)
1992-19-24	W. Damm, F. Liu, Th. Peikenkamp: Evaluation and Parallelization of Functions in Functional + Logic Languages (abstract)
1992-19-25	M. Kesseler: Communication Issues Regarding Parallel Functional Graph Rewriting
1992-19-26	Th. Peikenkamp: Charakterizing and representing neededness in functional loginc languages (abstract)
1992-19-27	H. Doerr: Monitoring with Graph-Grammars as formal operational Models
1992-19-28	J. van Groningen: Some implementation aspects of Concurrent Clean on distributed memory architectures
1992-19-29	G. Ostheimer: Load Bounding for Implicit Parallelism (abstract)
1992-20	H. Kuchen, F.J. Lopez Fraguas, J.J. Moreno Navarro, M. Rodriguez Artalejo: Implementing Disequality in a Lazy Functional Logic Language
1992-21	H. Kuchen, F.J. Lopez Fraguas: Result Directed Computing in a Functional Logic Language
1992-22	H. Kuchen, J.J. Moreno Navarro, M.V. Hermenegildo: Independent AND-Parallel Narrowing
1992-23	T. Margaria, B. Steffen: Distinguishing Formulas for Free
1992-24	K. Pohl: The Three Dimensions of Requirements Engineering
1992-25 *	R. Stainov: A Dynamic Configuration Facility for Multimedia Communications
1992-26 *	Michael von der Beeck: Integration of Structured Analysis and Timed Statecharts for Real-Time and Concurrency Specification
1992-27	W. Hans, St. Winkler: Aliasing and Groundness Analysis of Logic Programs through Abstract Interpretation and its Safety
1992-28 *	Gerhard Steinke, Matthias Jarke: Support for Security Modeling in Information Systems Design
1992-29	B. Schinzel: Warum Frauenforschung in Naturwissenschaft und Technik

1992-30	A. Kemper, G. Moerkotte, K. Peithner: Object-Orientation Axiomatised by Dynamic Logic
1992-32 *	Bernd Heinrichs, Kai Jakobs: Timer Handling in High-Performance Transport Systems
1992-33 *	B. Heinrichs, K. Jakobs, K. Lenßen, W. Reinhardt, A. Spinner: Euro-Bridge: Communication Services for Multimedia Applications
1992-34	C. Gerlhof, A. Kemper, Ch. Kilger, G. Moerkotte: Partition-Based Clustering in Object Bases: From Theory to Practice
1992-35	J. Börstler: Feature-Oriented Classification and Reuse in IPSEN
1992-36	M. Jarke, J. Bubenko, C. Rolland, A. Sutcliffe, Y. Vassiliou: Theories Underlying Requirements Engineering: An Overview of NATURE at Genesis
1992-37 *	K. Pohl, M. Jarke: Quality Information Systems: Repository Support for Evolving Process Models
1992-38	A. Zuendorf: Implementation of the imperative / rule based language PROGRES
1992-39	P. Koch: Intelligentes Backtracking bei der Auswertung funktional-logischer Programme
1992-40 *	Rudolf Mathar, Jürgen Mattfeldt: Channel Assignment in Cellular Radio Networks
1992-41 *	Gerhard Friedrich, Wolfgang Neidl: Constructive Utility in Model-Based Diagnosis Repair Systems
1992-42 *	P. S. Chen, R. Hennicker, M. Jarke: On the Retrieval of Reusable Software Components
1992-43	W. Hans, St.Winkler: Abstract Interpretation of Functional Logic Languages
1992-44	N. Kiesel, A. Schuerr, B. Westfechtel: Design and Evaluation of GRAS, a Graph-Oriented Database System for Engineering Applications
1993-01 *	Fachgruppe Informatik: Jahresbericht 1992
1993-02 *	Patrick Shicheng Chen: On Inference Rules of Logic-Based Information Retrieval Systems
1993-03	G. Hogen, R. Loogen: A New Stack Technique for the Management of Runtime Structures in Distributed Environments
1993-05	A. Zündorf: A Heuristic for the Subgraph Isomorphism Problem in Executing PROGRES
1993-06	A. Kemper, D. Kossmann: Adaptable Pointer Swizzling Strategies in Object Bases: Design, Realization, and Quantitative Analysis
1993-07 *	Graduiertenkolleg Informatik und Technik (Hrsg.): Graduiertenkolleg Informatik und Technik
1993-08 *	Matthias Berger: k-Coloring Vertices using a Neural Network with Convergence to Valid Solutions
1993-09	M. Buchheit, M. Jeusfeld, W. Nutt, M. Staudt: Subsumption between Queries to Object-Oriented Databases
1993-10	O. Burkart, B. Steffen: Pushdown Processes: Parallel Composition and Model Checking
1993-11 *	R. Große-Wienker, O. Hermanns, D. Menzenbach, A. Pollacks, S. Repetzki, J. Schwartz, K. Sonnenschein, B. Westfechtel: Das SUKITS-Projekt: A-posteriori-Integration heterogener CIM-Anwendungssysteme

1993-12 *	Rudolf Mathar, Jürgen Mattfeldt: On the Distribution of Cumulated Interference Power in Rayleigh Fading Channels
1993-13	O. Maler, L. Staiger: On Syntactic Congruences for omega-languages
1993-14	M. Jarke, St. Eherer, R. Gallersdoerfer, M. Jeusfeld, M. Staudt: ConceptBase - A Deductive Object Base Manager
1993-15	M. Staudt, H.W. Nissen, M.A. Jeusfeld: Query by Class, Rule and Concept
1993-16 *	M. Jarke, K. Pohl, St. Jacobs et al.: Requirements Engineering: An Integrated View of Representation Process and Domain
1993-17 *	M. Jarke, K. Pohl: Establishing Vision in Context: Towards a Model of Requirements Processes
1993-18	W. Hans, H. Kuchen, St. Winkler: Full Indexing for Lazy Narrowing
1993-19	W. Hans, J.J. Ruz, F. Saenz, St. Winkler: A VHDL Specification of a Shared Memory Parallel Machine for Babel
1993-20 *	K. Finke, M. Jarke, P. Szczurko, R. Soltysiak: Quality Management for Expert Systems in Process Control
1993-21	M. Jarke, M.A. Jeusfeld, P. Szczurko: Three Aspects of Intelligent Cooperation in the Quality Cycle
1994-01	Margit Generet, Sven Martin (eds.), Fachgruppe Informatik: Jahresbericht 1993
1994-02	M. Lefering: Development of Incremental Integration Tools Using Formal Specifications
1994-03 *	P. Constantopoulos, M. Jarke, J. Mylopoulos, Y. Vassiliou: The Software Information Base: A Server for Reuse
1994-04 *	Rolf Hager, Rudolf Mathar, Jürgen Mattfeldt: Intelligent Cruise Control and Reliable Communication of Mobile Stations
1994-05 *	Rolf Hager, Peter Hermesmann, Michael Portz: Feasibility of Authentication Procedures within Advanced Transport Telematics
1994-06 *	Claudia Popien, Bernd Meyer, Axel Kuepper: A Formal Approach to Service Import in ODP Trader Federations
1994-07	P. Peters, P. Szczurko: Integrating Models of Quality Management Methods by an Object-Oriented Repository
1994-08 *	Manfred Nagl, Bernhard Westfechtel: A Universal Component for the Administration in Distributed and Integrated Development Environments
1994-09 *	Patrick Horster, Holger Petersen: Signatur- und Authentifikationsverfahren auf der Basis des diskreten Logarithmusproblems
1994-11	A. Schürr: PROGRES, A Visual Language and Environment for PROgramming with Graph REwrite Systems
1994-12	A. Schürr: Specification of Graph Translators with Triple Graph Grammars
1994-13	A. Schürr: Logic Based Programmed Structure Rewriting Systems
1994-14	L. Staiger: Codes, Simplifying Words, and Open Set Condition
1994-15 *	Bernhard Westfechtel: A Graph-Based System for Managing Configurations of Engineering Design Documents
1994-16	P. Klein: Designing Software with Modula-3
1994-17	I. Litovsky, L. Staiger: Finite acceptance of infinite words

1994-18 G. Hogen, R. Loogen: Parallel Functional Implementations: Graphbased vs. Stackbased Reduction
1994-19 M. Jeusfeld, U. Johnen: An Executable Meta Model for Re-Engineering of Database Schemas
1994-20 * R. Gallersdörfer, M. Jarke, K. Klabunde: Intelligent Networks as a Data Intensive Application (INDIA)
1994-21 M. Mohnen: Proving the Correctness of the Static Link Technique Using Evolving Algebras
1994-22 H. Fernau, L. Staiger: Valuations and Unambiguity of Languages, with Applications to Fractal Geometry
1994-24 * M. Jarke, K. Pohl, R. Dömges, St. Jacobs, H. W. Nissen: Requirements Information Management: The NATURE Approach
1994-25 * M. Jarke, K. Pohl, C. Rolland, J.-R. Schmitt: Experience-Based Method Evaluation and Improvement: A Process Modeling Approach
1994-26 * St. Jacobs, St. Kethers: Improving Communication and Decision Making within Quality Function Deployment
1994-27 * M. Jarke, H. W. Nissen, K. Pohl: Tool Integration in Evolving Information Systems Environments
1994-28 O. Burkart, D. Caucal, B. Steffen: An Elementary Bisimulation Decision Procedure for Arbitrary Context-Free Processes
1995-01 * Fachgruppe Informatik: Jahresbericht 1994
1995-02 Andy Schürr, Andreas J. Winter, Albert Zündorf: Graph Grammar Engineering with PROGRES
1995-03 Ludwig Staiger: A Tight Upper Bound on Kolmogorov Complexity by Hausdorff Dimension and Uniformly Optimal Prediction
1995-04 Birgitta König-Ries, Sven Helmer, Guido Moerkotte: An experimental study on the complexity of left-deep join ordering problems for cyclic queries
1995-05 Sophie Cluet, Guido Moerkotte: Efficient Evaluation of Aggregates on Bulk Types
1995-06 Sophie Cluet, Guido Moerkotte: Nested Queries in Object Bases
1995-07 Sophie Cluet, Guido Moerkotte: Query Optimization Techniques Exploiting Class Hierarchies
1995-08 Markus Mohnen: Efficient Compile-Time Garbage Collection for Arbitrary Data Structures
1995-09 Markus Mohnen: Functional Specification of Imperative Programs: An Alternative Point of View of Functional Languages
1995-10 Rainer Gallersdörfer, Matthias Nicola: Improving Performance in Replicated Databases through Relaxed Coherency
1995-11 * M.Staudt, K.von Thadden: Subsumption Checking in Knowledge Bases
1995-12 * G.V.Zemanek, H.W.Nissen, H.Hubert, M.Jarke: Requirements Analysis from Multiple Perspectives: Experiences with Conceptual Modeling Technology
1995-13 * M.Staudt, M.Jarke: Incremental Maintenance of Externally Materialized Views
1995-14 * P.Peters, P.Szczurko, M.Jeusfeld: Oriented Information Management: Conceptual Models at Work

1995-15 *	Matthias Jarke, Sudha Ram (Hrsg.): WITS 95 Proceedings of the 5th Annual Workshop on Information Technologies and Systems
1995-16 *	W.Hans, St.Winkler, F.Saenz: Distributed Execution in Functional Logic Programming
1996-01 *	Jahresbericht 1995
1996-02	Michael Hanus, Christian Prehofer: Higher-Order Narrowing with Definitional Trees
1996-03 *	W.Scheufele, G.Moerkotte: Optimal Ordering of Selections and Joins in Acyclic Queries with Expensive Predicates
1996-04	Klaus Pohl: PRO-ART: Enabling Requirements Pre-Traceability
1996-05	Klaus Pohl: Requirements Engineering: An Overview
1996-06 *	M.Jarke, W.Marquardt: Design and Evaluation of Computer–Aided Process Modelling Tools
1996-07	Olaf Chitil: The Sigma-Semantics: A Comprehensive Semantics for Functional Programs
1996-08 *	S.Sripada: On Entropy and the Limitations of the Second Law of Thermodynamics
1996-09	Michael Hanus (Ed.): Proceedings of the Poster Session of ALP96 - Fifth International Conference on Algebraic and Logic Programming
1996-09-0	Michael Hanus (Ed.): Proceedings of the Poster Session of ALP 96 - Fifth International Conference on Algebraic and Logic Programming: Introduction and table of contents
1996-09-1	Ilies Alouini: An Implementation of Conditional Concurrent Rewriting on Distributed Memory Machines
1996-09-2	Olivier Danvy, Karoline Malmkjær: On the Idempotence of the CPS Transformation
1996-09-3	Víctor M. Gulías, José L. Freire: Concurrent Programming in Haskell
1996-09-4	Sébastien Limet, Pierre Réty: On Decidability of Unifiability Modulo Rewrite Systems
1996-09-5	Alexandre Tessier: Declarative Debugging in Constraint Logic Programming
1996-10	Reidar Conradi, Bernhard Westfechtel: Version Models for Software Configuration Management
1996-11 *	C.Weise, D.Lenzkes: A Fast Decision Algorithm for Timed Refinement
1996-12 *	R.Dömges, K.Pohl, M.Jarke, B.Lohmann, W.Marquardt: PRO-ART/CE* — An Environment for Managing the Evolution of Chemical Process Simulation Models
1996-13 *	K.Pohl, R.Klamma, K.Weidenhaupt, R.Dömges, P.Haumer, M.Jarke: A Framework for Process-Integrated Tools
1996-14 *	R.Gallersdörfer, K.Klabunde, A.Stolz, M.Eßmajor: INDIA — Intelligent Networks as a Data Intensive Application, Final Project Report, June 1996
1996-15 *	H.Schimpe, M.Staudt: VAREX: An Environment for Validating and Refining Rule Bases
1996-16 *	M.Jarke, M.Gebhardt, S.Jacobs, H.Nissen: Conflict Analysis Across Heterogeneous Viewpoints: Formalization and Visualization
1996-17	Manfred A. Jeusfeld, Tung X. Bui: Decision Support Components on the Internet

1996-18	Manfred A. Jeusfeld, Mike Papazoglou: Information Brokering: Design, Search and Transformation
1996-19 *	P.Peters, M.Jarke: Simulating the impact of information flows in networked organizations
1996-20	Matthias Jarke, Peter Peters, Manfred A. Jeusfeld: Model-driven planning and design of cooperative information systems
1996-21 *	G.de Michelis, E.Dubois, M.Jarke, F.Matthes, J.Mylopoulos, K.Pohl, J.Schmidt, C.Woo, E.Yu: Cooperative information systems: a manifesto
1996-22 *	S.Jacobs, M.Gebhardt, S.Kethers, W.Rzasa: Filling HTML forms simultaneously: CoWeb architecture and functionality
1996-23 *	M.Gebhardt, S.Jacobs: Conflict Management in Design
1997-01	Michael Hanus, Frank Zartmann (eds.): Jahresbericht 1996
1997-02	Johannes Faassen: Using full parallel Boltzmann Machines for Optimization
1997-03	Andreas Winter, Andy Schürr: Modules and Updatable Graph Views for PROgrammed Graph REwriting Systems
1997-04	Markus Mohnen, Stefan Tobies: Implementing Context Patterns in the Glasgow Haskell Compiler
1997-05 *	S.Gruner: Schemakorrespondenzaxiome unterstützen die paargrammatische Spezifikation inkrementeller Integrationswerkzeuge
1997-06	Matthias Nicola, Matthias Jarke: Design and Evaluation of Wireless Health Care Information Systems in Developing Countries
1997-07	Petra Hofstedt: Taskparallele Skelette für irregulär strukturierte Probleme in deklarativen Sprachen
1997-08	Dorothea Blostein, Andy Schürr: Computing with Graphs and Graph Rewriting
1997-09	Carl-Arndt Krapp, Bernhard Westfechtel: Feedback Handling in Dynamic Task Nets
1997-10	Matthias Nicola, Matthias Jarke: Integrating Replication and Communication in Performance Models of Distributed Databases
1997-11 *	R. Klamma, P. Peters, M. Jarke: Workflow Support for Failure Management in Federated Organizations
1997-13	Markus Mohnen: Optimising the Memory Management of Higher-Order Functional Programs
1997-14	Roland Baumann: Client/Server Distribution in a Structure-Oriented Database Management System
1997-15	George Botorog: High-Level Parallel Programming and the Efficient Implementation of Numerical Algorithms
1998-01 *	Fachgruppe Informatik: Jahresbericht 1997
1998-02	Stefan Gruner, Manfred Nagel, Andy Schürr: Fine-grained and Structure-Oriented Document Integration Tools are Needed for Development Processes
1998-03	Stefan Gruner: Einige Anmerkungen zur graphgrammatischen Spezifikation von Integrationswerkzeugen nach Westfechtel, Janning, Lefering und Schürr
1998-04 *	O. Kubitz: Mobile Robots in Dynamic Environments
1998-05	Martin Leucker, Stephan Tobies: Truth - A Verification Platform for Distributed Systems

1998-06 * Matthias Oliver Berger: DECT in the Factory of the Future

1998-07 M. Arnold, M. Erdmann, M. Glinz, P. Haumer, R. Knoll, B. Paech, K. Pohl, J. Ryser, R. Studer, K. Weidenhaupt: Survey on the Scenario Use in Twelve Selected Industrial Projects

1998-09 * Th. Lehmann: Geometrische Ausrichtung medizinischer Bilder am Beispiel intraoraler Radiographien

1998-10 * M. Nicola, M. Jarke: Performance Modeling of Distributed and Replicated Databases

1998-11 * Ansgar Schleicher, Bernhard Westfechtel, Dirk Jäger: Modeling Dynamic Software Processes in UML

1998-12 * W. Appelt, M. Jarke: Interoperable Tools for Cooperation Support using the World Wide Web

1998-13 Klaus Indermark: Semantik rekursiver Funktionsdefinitionen mit Striktheitsinformation

1999-01 * Jahresbericht 1998

1999-02 * F. Huch: Verifcation of Erlang Programs using Abstract Interpretation and Model Checking — Extended Version

1999-03 * R. Gallersdörfer, M. Jarke, M. Nicola: The ADR Replication Manager

1999-04 María Alpuente, Michael Hanus, Salvador Lucas, Germán Vidal: Specialization of Functional Logic Programs Based on Needed Narrowing

1999-05 * W. Thomas (Ed.): DLT 99 - Developments in Language Theory Fourth International Conference

1999-06 * Kai Jakobs, Klaus-Dieter Kleefeld: Informationssysteme für die angewandte historische Geographie

1999-07 Thomas Wilke: CTL+ is exponentially more succinct than CTL

1999-08 Oliver Matz: Dot-Depth and Monadic Quantifier Alternation over Pictures

2000-01 * Jahresbericht 1999

2000-02 Jens Vöge, Marcin Jurdzinski A Discrete Strategy Improvement Algorithm for Solving Parity Games

2000-03 D. Jäger, A. Schleicher, B. Westfechtel: UPGRADE: A Framework for Building Graph-Based Software Engineering Tools

2000-04 Andreas Becks, Stefan Sklorz, Matthias Jarke: Exploring the Semantic Structure of Technical Document Collections: A Cooperative Systems Approach

2000-05 Mareike Schoop: Cooperative Document Management

2000-06 Mareike Schoop, Christoph Quix (eds.): Proceedings of the Fifth International Workshop on the Language-Action Perspective on Communication Modelling

2000-07 * Markus Mohnen, Pieter Koopman (Eds.): Proceedings of the 12th International Workshop of Functional Languages

2000-08 Thomas Arts, Thomas Noll: Verifying Generic Erlang Client-Server Implementations

2001-01 * Jahresbericht 2000

2001-02 Benedikt Bollig, Martin Leucker: Deciding LTL over Mazurkiewicz Traces

2001-03 Thierry Cachat: The power of one-letter rational languages

2001-04	Benedikt Bollig, Martin Leucker, Michael Weber: Local Parallel Model Checking for the Alternation Free mu-Calculus
2001-05	Benedikt Bollig, Martin Leucker, Thomas Noll: Regular MSC Languages
2001-06	Achim Blumensath: Prefix-Recognisable Graphs and Monadic Second-Order Logic
2001-07	Martin Grohe, Stefan Wöhrle: An Existential Locality Theorem
2001-08	Mareike Schoop, James Taylor (eds.): Proceedings of the Sixth International Workshop on the Language-Action Perspective on Communication Modelling
2001-09	Thomas Arts, Jürgen Giesl: A collection of examples for termination of term rewriting using dependency pairs
2001-10	Achim Blumensath: Axiomatising Tree-interpretable Structures
2001-11	Klaus Indermark, Thomas Noll (eds.): Kolloquium Programmiersprachen und Grundlagen der Programmierung
2002-01 *	Jahresbericht 2001
2002-02	Jürgen Giesl, Aart Middeldorp: Transformation Techniques for Context-Sensitive Rewrite Systems
2002-03	Benedikt Bollig, Martin Leucker, Thomas Noll: Generalised Regular MSC Languages
2002-04	Jürgen Giesl, Aart Middeldorp: Innermost Termination of Context-Sensitive Rewriting
2002-05	Horst Lichter, Thomas von der Maßen, Thomas Weiler: Modelling Requirements and Architectures for Software Product Lines
2002-06	Henry N. Adorna: 3-Party Message Complexity is Better than 2-Party Ones for Proving Lower Bounds on the Size of Minimal Nondeterministic Finite Automata
2002-07	Jörg Dahmen: Invariant Image Object Recognition using Gaussian Mixture Densities
2002-08	Markus Mohnen: An Open Framework for Data-Flow Analysis in Java
2002-09	Markus Mohnen: Interfaces with Default Implementations in Java
2002-10	Martin Leucker: Logics for Mazurkiewicz traces
2002-11	Jürgen Giesl, Hans Zantema: Liveness in Rewriting
2003-01 *	Jahresbericht 2002
2003-02	Jürgen Giesl, René Thiemann: Size-Change Termination for Term Rewriting
2003-03	Jürgen Giesl, Deepak Kapur: Deciding Inductive Validity of Equations
2003-04	Jürgen Giesl, René Thiemann, Peter Schneider-Kamp, Stephan Falke: Improving Dependency Pairs
2003-05	Christof Löding, Philipp Rohde: Solving the Sabotage Game is PSPACE-hard
2003-06	Franz Josef Och: Statistical Machine Translation: From Single-Word Models to Alignment Templates
2003-07	Horst Lichter, Thomas von der Maßen, Alexander Nyßen, Thomas Weiler: Vergleich von Ansätzen zur Feature Modellierung bei der Softwareproduktlinienentwicklung
2003-08	Jürgen Giesl, René Thiemann, Peter Schneider-Kamp, Stephan Falke: Mechanizing Dependency Pairs
2004-01 *	Fachgruppe Informatik: Jahresbericht 2003

2004-02	Benedikt Bollig, Martin Leucker: Message-Passing Automata are expressively equivalent to EMSO logic
2004-03	Delia Kesner, Femke van Raamsdonk, Joe Wells (eds.): HOR 2004 – 2nd International Workshop on Higher-Order Rewriting
2004-04	Slim Abdennadher, Christophe Ringeissen (eds.): RULE 04 – Fifth International Workshop on Rule-Based Programming
2004-05	Herbert Kuchen (ed.): WFLP 04 – 13th International Workshop on Functional and (Constraint) Logic Programming
2004-06	Sergio Antoy, Yoshihito Toyama (eds.): WRS 04 – 4th International Workshop on Reduction Strategies in Rewriting and Programming
2004-07	Michael Codish, Aart Middeldorp (eds.): WST 04 – 7th International Workshop on Termination
2004-08	Klaus Indermark, Thomas Noll: Algebraic Correctness Proofs for Compiling Recursive Function Definitions with Strictness Information
2004-09	Joachim Kneis, Daniel Mölle, Stefan Richter, Peter Rossmanith: Parameterized Power Domination Complexity
2004-10	Zinaida Benenson, Felix C. Gärtner, Dogan Kesdogan: Secure Multi-Party Computation with Security Modules
2005-01 *	Fachgruppe Informatik: Jahresbericht 2004
2005-02	Maximillian Dornseif, Felix C. Gärtner, Thorsten Holz, Martin Mink: An Offensive Approach to Teaching Information Security: "Aachen Summer School Applied IT Security"
2005-03	Jürgen Giesl, René Thiemann, Peter Schneider-Kamp: Proving and Disproving Termination of Higher-Order Functions
2005-04	Daniel Mölle, Stefan Richter, Peter Rossmanith: A Faster Algorithm for the Steiner Tree Problem
2005-05	Fabien Pouget, Thorsten Holz: A Pointillist Approach for Comparing Honeypots
2005-06	Simon Fischer, Berthold Vöcking: Adaptive Routing with Stale Information
2005-07	Felix C. Freiling, Thorsten Holz, Georg Wicherski: Botnet Tracking: Exploring a Root-Cause Methodology to Prevent Distributed Denial-of-Service Attacks
2005-08	Joachim Kneis, Peter Rossmanith: A New Satisfiability Algorithm With Applications To Max-Cut
2005-09	Klaus Kursawe, Felix C. Freiling: Byzantine Fault Tolerance on General Hybrid Adversary Structures
2005-10	Benedikt Bollig: Automata and Logics for Message Sequence Charts
2005-11	Simon Fischer, Berthold Vöcking: A Counterexample to the Fully Mixed Nash Equilibrium Conjecture
2005-12	Neeraj Mittal, Felix Freiling, S. Venkatesan, Lucia Draque Penso: Efficient Reductions for Wait-Free Termination Detection in Faulty Distributed Systems
2005-13	Carole Delporte-Gallet, Hugues Fauconnier, Felix C. Freiling: Revisiting Failure Detection and Consensus in Omission Failure Environments
2005-14	Felix C. Freiling, Sukumar Ghosh: Code Stabilization
2005-15	Uwe Naumann: The Complexity of Derivative Computation

2005-16	Uwe Naumann: Syntax-Directed Derivative Code (Part I: Tangent-Linear Code)
2005-17	Uwe Naumann: Syntax-directed Derivative Code (Part II: Intraprocedural Adjoint Code)
2005-18	Thomas von der Maßen, Klaus Müller, John MacGregor, Eva Geisberger, Jörg Dörr, Frank Houdek, Harbhajan Singh, Holger Wußmann, Hans-Veit Bacher, Barbara Paech: Einsatz von Features im Software-Entwicklungsprozess - Abschlußbericht des GI-Arbeitskreises "Features"
2005-19	Uwe Naumann, Andre Vehreschild: Tangent-Linear Code by Augmented LL-Parsers
2005-20	Felix C. Freiling, Martin Mink: Bericht über den Workshop zur Ausbildung im Bereich IT-Sicherheit Hochschulausbildung, berufliche Weiterbildung, Zertifizierung von Ausbildungsangeboten am 11. und 12. August 2005 in Köln organisiert von RWTH Aachen in Kooperation mit BITKOM, BSI, DLR und Gesellschaft fuer Informatik (GI) e.V.
2005-21	Thomas Noll, Stefan Rieger: Optimization of Straight-Line Code Revisited
2005-22	Felix Freiling, Maurice Herlihy, Lucia Draque Penso: Optimal Randomized Fair Exchange with Secret Shared Coins
2005-23	Heiner Ackermann, Alantha Newman, Heiko Röglin, Berthold Vöcking: Decision Making Based on Approximate and Smoothed Pareto Curves
2005-24	Alexander Becher, Zinaida Benenson, Maximillian Dornseif: Tampering with Motes: Real-World Physical Attacks on Wireless Sensor Networks
2006-03	Uwe Naumann: Intraprocedural Adjoint Code Generated by the Differentiation-Enabled NAGWare Fortran Compiler
2006-04	Ebadollah Varnik, Uwe Naumann, Andrew Lyons: Toward Low Static Memory Jacobian Accumulution
2006-06	Joachim Kneis, Daniel Mölle, Stefan Richter, Peter Rossmanith: Divide-and-Color
2006-07	Thomas Colcombet, Christof Löding:: Transforming structures by set interpretations
2006-08	Uwe Naumann, Yuxiao Hu: Optimal Vertex Elimination in Single-Expression-Use Graphs
2006-10	Mesut Günes, Alexander Zimmermann, Martin Wenig, Jan Ritzerfeld, Ulrich Meis: From Simulations to Testbeds - Architecture of the Hybrid MCG-Mesh Testbed
2006-11	Bastian Schlich, Michael Rohrbach, Michael Weber, Stefan Kowalewski: Model Checking Software for Microcontrollers

* These reports are only available as a printed version.

Please contact `biblio@informatik.rwth-aachen.de` to obtain copies.