RWTH Aachen

Department of Computer Science
Technical Report

Byzantine Fault Tolerance on General Hybrid Adversary Structures

Klaus Kursawe and Felix C. Freiling

ISSN 0935–3232 · Aachener Informatik Berichte · AIB-2005-09

RWTH Aachen · Department of Computer Science · April 2005

Byzantine Fault Tolerance on General Hybrid Adversary Structures

Klaus Kursawe[1] and Felix C. Freiling[2]

[1] Dept. Elektotechniek ESAT - COSIC, Kasteelpark Arenberg 10, B 3001 Heverlee, Belgium
[2] Lehr- und Forschungsgebiet Informatik 4, RWTH Aachen, Germany

Abstract. Adversary structures are a generalization of the classical "at most t-out-of-n" threshold failure model which is used in many published Byzantine-tolerant protocols. An adversary structure basically lists all coalitions of parties whose corruption the protocol should tolerate. Using adversary structures it is possible to encode dependent failure models, such as "either all Linux machines fail or all Windows machines but not both at the same time". We describe a general technique that allows to transform an algorithm designed for the threshold model into an algorithm that works for general adversary structures. Our technique is based on several (partly informal) rules which describe how the algorithm and its proof must be augmented so that general adversary structures can be tolerated. We demonstrate the applicability of our approach by transforming an asynchronous Byzantine-tolerant reliable broadcast protocol into one that tolerates Byzantine adversary structures. We also consider similar transformations for hybrid failures (combinations of different fault models) and discuss ways to map adversary structures to the real world and manage them efficiently.

1 Introduction

The *Byzantine failure model* [22, 19] allows faulty components to behave in an arbitrary manner. It was introduced as a worst-case failure mode for components in safety-critical computing systems and is used as the basis for designs in aircraft control [27] and the control system of the International Space Station [21]. There, *Byzantine Agreement* protocols maintain consistency of the redundant computer systems which run the control software.

Recently, we are seeing an increasing use of the Byzantine failure model in the area of security [24, 16, 5, 7, 17, 28, 29]. This is because the worst case assumption of arbitrary behavior can also be regarded as malicious behavior of an attacker of the system. Hence, a Byzantine-tolerant system can also be regarded as secure against malicious attackers. However, there are several concerns that suggest care in assuming Byzantine failures in security related settings. The main objection is usually that the view of the world is too homogenous: component failures are assumed to happen independently. This is usually expressed in what we call the *threshold failure model* by the statement "any t out of n components can fail". In the original fault tolerance settings, this assumption is justifiable because hardware failures empirically show signs of independence [3] and using this assumption the overall failure rate of a system can be calculated from the failure probabilities of individual components.

In the area of security, component failures due to security threats (denial-of-service, system takeovers, web site defacements, etc.) do not happen independently [11]. For example, if a set of replicated servers that offer a service on the Internet all run the same operating system and a new vulnerability for this class of systems is discovered, then all may be attacked and fail simultaneously. In fact, the first papers on Byzantine fault tolerance where never meant to consider malicious faults for exactly this reason.

A promising approach to deal with the problem of independent failures is to use the concept of *general adversary structures*. The idea of this concept is to explicitly consider the real-world dependencies between failures and map these dependencies to sets of components which may fail concurrently. Instead of stating thresholds like "any t out of n components may fail" we now say that "either all Linux machines fail or all Windows machines but not both at the same time". This approach does not increase the number of tolerable failures, but adds a degree of flexibility which is needed to model real world situations.

Of course, even without malicious attackers, various factors influence the systems deployed today. Parties at the same physical location may fail simultaneously due to a power failure, and systems with the same operating-system may be victims of the same type of vulnerability. So adversary structures do not facilitate the calculations of failure probabilities, they merely offer a method to develop systems that have higher assumption coverage [23] than previous systems which are based on the threshold failure model.

In this paper, we use adversary structures to go beyond the traditional threshold failure model. We argue that many published algorithms that work for the old failure model also work for general adversary structures with only small modifications. We describe a general technique that allows to transform an algorithm designed for the threshold model into an algorithm that works for general adversary structures. The technique is based on several (partly informal) rules which describe how the algorithm and its proof must be augmented so that general adversary structures can be tolerated. The rules are derived from relative direct correspondences between the threshold model and adversary structures, which allow for the generalization of existing protocols.

We demonstrate the applicability of our approach by transforming an asynchronous Byzantine-tolerant reliable broadcast protocol [6] into one that tolerates Byzantine adversary structures. We argue that the approach taken here is rather general and also works for many other protocols for different problems, e.g., Byzantine Agreement and Interactive Consistency.

Furthermore, instead of only accepting Byzantine failures, we also consider the same type of transformation to automatically tolerate *hybrid failures*, a mixture of Byzantine and (silent) crash failures, thereby increasing the fault-tolerance of the protocol even further. More precisely, if c is the number of crash failures, and b is the number of Byzantine failures, then the necessary and sufficient condition is $2c + 3b < n$, where n is the total number of redundant components in the system.

The flexibility provided by the concept of adversary structures does not come for free. In general, an adversary structure can contain the set of all subsets of system components, resulting in an exponential number of possible failure coalitions. In most settings, the overhead required to identify, store and compute on these sets can easily become the bottleneck. Therefore, as further contribution, we discuss ways on how to efficiently manage adversary structures: We define means to restrict the flexibility offered by the adversary structures in a meaningful way, i.e., in a way that makes the sets easier to manage, while still mirroring failure dependencies that occur in the real world.

1.1 Related Work

Malkhi and Reiter [20] where the first to use generic sets rather than thresholds to describe Byzantine-tolerant protocols in their Byzantine quorum system. Junqueira and Marzullo [15] used a systematic approach by introducing the concept of *core* and *survivor* sets to replace the usual t-out-of-n threshold. These sets are sets of parties that satisfy certain properties required by the protocol (such as containing at least one non-faulty party). As long as these properties are satisfied, it does not matter how the sets are composed or what size they are. The core and survivor sets roughly correspond to the *small, big* and *full* sets introduced in Section 2.

We go a step further than Junqueira and Marzullo [15] by defining the minimal requirements on the set of corrupted parties which is needed for our protocols to work. This is a generalized equivalent to the typical requirement commonly stated for Byzantine fault tolerance, namely that $n > 3t+1$, where t is the number of failures and n is the overall number of parties. Once the stated requirements are met, we can also guarantee that the sets with the necessary requirements do exist and can easily be found.

Adversary structures as used here have first been defined for secret sharing schemes [14, 2, 26] in cryptography where an *access structure* Γ defines which sets (coalitions) of parties are allowed to combine the secret. The first usage of adversary structures beyond secret sharing goes back to Fitzi, Hirt and Maurer [13, 10] and since then has occasionally been used in the area of secure multiparty computation [12].

1.2 Paper Outline

We first present the model and some preliminary definitions in Section 2. We then describe in detail the rules of our transformation in Section 3. We apply these rules in Section 4 to a Byzantine-tolerant reliable broadcast protocol, yielding a protocol which tolerates general hybrid adversary structures. In Section 5 we discuss the practicality of adversary structures and present mechanisms to manage them efficiently.

2 Model and Properties

Asynchronous message-passing systems. A system consists of a set $\mathcal{P}$ of participants (sometimes called parties, players, processes, or processors) which communicate over reliable point-to-point channels using message passing. The system is *asynchronous*, i.e., there are no bounds on message delivery delays or relative process speeds.

Crash, Byzantine, and hybrid failures. While channels are reliable, participants may be faulty. Two types of faulty behavior are considered: *crash* and *Byzantine* failures. A process crashes by just ceasing operation without notice, i.e., a crashed process stops to execute steps of its algorithm. A process becomes Byzantine if it behaves in an arbitrary manner. Byzantine processes are often considered to be under the control of a malicious adversary who tries to prevent the protocol from satisfying its specification. Formally, Byzantine behavior is a real superset of crash behavior, however, a crashed process is usually not regarded to be under the control of an adversary and so these two failure modes are considered orthogonal in the area of security. A participant which fails is called *faulty*, *dishonest* or *corrupted*.

We say that we operate in the *crash model* if participants can only fail by crashing. Similarly, the *Byzantine model* assumes that participants fail only by becoming Byzantine. We define the *hybrid model* as the model in which processes can fail either by crashing or by becoming Byzantine.

Byzantine Agreement. In (binary) Byzantine Agreement, a set of parties $\mathcal{P} = \{P_1, \ldots, P_n\}$ wishes to reach agreement on a single output bit value 0 or 1. Every party P_i starts with an input bit ρ_i. A Byzantine Agreement protocol has to satisfy the following three properties:

Validity If all parties are honest and have the same input value ρ, then no honest party decides $1 - \rho$.
Agreement All honest parties that decide decide the same value.
Termination All honest parties eventually decide (with probability 1).

Note that we consider here the probabilistic version of Byzantine Agreement which for our purposes is the simplest one and avoids the impossibility result of Fischer, Lynch and Paterson [8] on the solvability of consensus in asynchronous systems.

Byzantine Agreement requires a certain minimal level of redundancy to be solvable in different models. For example, in the Byzantine model it is well-known that Byzantine Agreement requires that $n > 3t + 1$ where n is the total number of parties and t is the number of parties which can be corrupted. In this paper, such thresholds disappear. Instead, we allow the protocol designer to explicitely define each and every set of parties that may simultaneously fail, and we define the minimal requirements these sets need to satisfy.

Adversary structures and the predicate $\mathcal{Q}^{(3,2)}$. Let $\mathcal{P}$ be the set of all participants. An *adversary class* $\mathcal{C} = (B, C)$ is a pair of subsets of $\mathcal{P}$, i.e., $B, C \subset \mathcal{P}$. An adversary class $\mathcal{C}' = (B', C')$ is contained in an adversary class $\mathcal{C} = (B, C)$ if B' is a subset of B and C' is a subset of C. An *adversary structure* $\mathcal{Z}$ is a monotone set of adversary classes $\mathcal{C} = (B, C)$, i.e., all classes contained in $\mathcal{C}$ are also in $\mathcal{Z}$ [9].

As an example, let $\mathcal{P} = \{P_1, P_2, P_3\}$ and consider the following structures:

$$\mathcal{Z}_1 = \{(\{P_1, P_2\}, \{\})\}$$

$$\mathcal{Z}_2 = \{(\{P_1, P_2\}, \{\}), (\{P1\}, \{\}), (\{\}, \{\})\}$$

$$\mathcal{Z}_3 = \{(\{P_1\}, \{\}), (\{\}, \{P_2\}), (\{\}, \{P_3\}), (\{\}, \{\})\}$$

The structure $\mathcal{Z}_1$ is not monotone and therefore no adversary structure whereas both $\mathcal{Z}_2$ and $\mathcal{Z}_3$ are adversary structures.

An adversary structure $\mathcal{Z}$ satisfies the predicate $\mathcal{Q}^{(3,2)}(\mathcal{P}, \mathcal{Z})$ iff

$$\forall (B_1, C_1), (B_2, C_2), (B_3, C_3) \in \mathcal{Z} : \{B_1 \cup B_2 \cup B_3 \cup C_1 \cup C_2\} \neq \mathcal{P}$$

Roughly speaking, the predicate $\mathcal{Q}^{(3,2)}(\mathcal{P}, \mathcal{Z})$ states that no three elements from the adversary structure $\mathcal{Z}$ add up to the total set of participants $\mathcal{P}$. Considering the above examples, $\mathcal{Z}_2$ satisfies $\mathcal{Q}^{(3,2)}(\mathcal{P}, \mathcal{Z})$ while $\mathcal{Z}_3$ does not.

For an adversary structure $\mathcal{Z}$ and any $(B, C) \in \mathcal{Z}$, the set B corresponds to those participants which can become Byzantine and the set C to those which can crash. We now show that an adversary structure $\mathcal{Z}$ that satisfies $Q^{(3,2)}(\mathcal{P}, \mathcal{Z})$ is necessary sufficient to solve Byzantine agreement in an asynchronous environment.

Theorem 1. *To solve (probabilistic) asynchronous Byzantine agreement for a set of participants $\mathcal{P}$ on an adversary structure $\mathcal{Z}$, a necessary and sufficient condition is that $\mathcal{Z}$ satisfies $\mathcal{Q}^{(3,2)}(\mathcal{P}, \mathcal{Z})$.*

Proof. We will show only necessity here; sufficiency follows from the correctness of a corresponding (randomized) protocol that has been presented by Kursawe [18].

Let $\mathcal{Z}$ be an adversary structure and let $(B_1, C_1), (B_2, C_2), (B_3, C_3) \in \mathcal{Z}$ such that $\mathcal{P} = \{B_1 \cup B_2 \cup B_3 \cup C_1 \cup C_2\}$. Now fix two honest parties $P_1 \notin B_1 \cup C_1$ and $P_2 \notin B_2 \cup C_2$.

Now P_1 has to be able to terminate the protocol without ever having heard from any party in $B_1 \cup C_1$. Similarly, P_2 must be able to decide without ever having heard of any party in the set $B_2 \cup C_2$. Thus, the input-values P_1 sees are the values from the parties in the set

$$B_2 \cup B_3 \cup C_2,$$

while P_2 sees the input values of the parties in

$$B_1 \cup B_3 \cup C_1.$$

As all parties in B_1, B_2, and B_3 are Byzantine and might send different input values to different parties, these sets are disjunct, and an adversary can easily cause disagreement. □

Theorem 1 encompasses several special cases for the non-hybrid model (where an adversary structure is a set of subsets of $\mathcal{P}$ rather than a set of classes), and the threshold model:

- If only crash failures are possible, then it is necessary and sufficient that $\mathcal{Z}$ satisfies $Q^2(\mathcal{P}, \mathcal{Z})$, i.e., that no two sets in $\mathcal{Z}$ cover $\mathcal{P}$.
- If only Byzantine failures are possible, then it is necessary and sufficient that $\mathcal{Z}$ satisfies $Q^3(\mathcal{P}, \mathcal{Z})$, i.e., that no three sets in $\mathcal{Z}$ cover $\mathcal{P}$.
- If $\mathcal{Z}$ is homogeneous, i.e., all $\mathcal{Z}$ contains exactly all classes (C, B) with $|C| = c, |B| = b$, then $2c + 3b < n$ is a necessary and sufficient condition. This corresponds to the threshold model without adversary structures, where b is the maximum number of Byzantine failures and c is the maximum number of crash failures.

Since many problems like Broadcast or Interactive Consistency are equivalent to Byzantine Agreement, the predicate $\mathcal{Q}^{(3,2)}$ is an essential requirement for any useful adversary structure in crash, Byzantine or hybrid settings.

3 Protocol Transformation

Given a protocol in the threshold model, we ask the question, whether this protocol can be transformed into a protocol that works also for general adversary structures. We now give a set of (partly informal) rules on how such a transformation can be achieved.

For arguing about protocols in the new setting of adversary structures, we need to completely abstract away all thresholds. Instead of thresholds, we define sets with certain properties that we need in the algorithm and in the proofs. As an example, consider a line in a protocol that states:

wait for $n - t$ votes

Taken literally, this statement demands that we wait for $n-t$ messages to arrive. However, the deeper semantics of this statement look more closely at the reason *why* we need to wait for exactly $n-t$ votes, i.e., this number of votes has a certain property. For example, in an asynchronous protocol, it is the highest number of votes we can wait for without risking a deadlock.

In this direction, if we want to argue about complicated adversary structures, it is helpful to identify the properties of the thresholds that are used in today's protocols and their proofs. Once we defined the properties of the individual thresholds we prove certain properties of the resulting concepts. These properties can be used in the proofs of the transformed protocol. In this way, it is relatively easy to move a protocol from the simple threshold model to a more refined failure model.

We start by defining sets that correspond to the different thresholds usually needed in the conventional model.

Definition 1. *Let $\mathcal{Z}$ be an adversary structure that satisfies $\mathcal{Q}^{(3,2)}(\mathcal{P}, \mathcal{Z})$. A set $\mathcal{A} \subset \mathcal{P}$ is called a*

- full set*, if there exists a class $(B, C) \in \mathcal{Z}$, such that $\mathcal{A} \supseteq \mathcal{P} \setminus (B \cup C)$.*
- big set*, if there exists two classes (B_1, C_1) and $(B_2, C_2) \in \mathcal{Z}$, such that $\mathcal{A} \supseteq \mathcal{P} \setminus (B_1 \cup C_1 \cup B_2)$.*
- small set*, if there is no class $(B, C) \in \mathcal{Z}$ such that $B \supseteq \mathcal{A}$.*

Intuitively, $\mathcal{A}$ is a full set if all parties in $\mathcal{P} \setminus \mathcal{A}$ might be faulty; after receiving messages from a full set $\mathcal{A}$ of parties, a party cannot expect to receive more messages. This corresponds to the threshold $n-t$ in our previous notation. Note that the set of all uncorrupted parties is a full set.

Given a full set of parties, some of which are Byzantine, a *big set* is the set that remains when the Byzantine parties are taken away. This corresponds to the threshold $n - 2t$ in our conventional notation.

Finally, a small set of parties is a set that contains at least one non-Byzantine party. This corresponds to the threshold of $t + 1$.

Transformation rules. Given a protocol designed in the threshold model, consider all protocol statements where thresholds are hard-coded to govern the execution of the protocol. Now consider every such statement carefully and obey the following three rules:

1. If the threshold to be reached is of the type "$n-t$" (e.g., wait for $n-t$ messages before advancing to the next statement), then replace this threshold with a reference to a *full set* (e.g., wait for messages corresponding to any full set in $\mathcal{Z}$).
2. If the threshold to be reached is of type "$n-2t$" (e.g., wait for $n-2t$ messages before advancing to the next statement), then replace this threshold with a reference to a *big set*.
3. If the threshold to be reached is of type "$t+1$" (e.g., upon receiving $t+1$ messages of the same type, do something), then replace this threshold with a reference to a *small set*.

Transforming the proofs. We will now prove some elementary properties of above sets. The properties will be useful when transforming the proof of a protocol in the threshold model to the model of general adversary structures.

Lemma 1. *The following statements hold:*

P1. *Any big set and any small set contain at least one non-Byzantine party.*
P2. *Two full sets always have at least one non-Byzantine party in common.*
P3. *Each full set of parties contains at least a big set of non-Byzantine parties.*
P4. *If a full set of parties does an action X, and no non-Byzantine party does both actions X and Y, then no full set of parties does Y.*
P5. *Every big set is a small set.*
P6. *Each pair of a full set and a big set of parties intersects.*

Proof. **P1**: Suppose statement 1 does not hold for big sets. Then there is a big set $\mathcal{A}$ and a class (C, B) such that $\mathcal{A} \subseteq B$. By definition, there are two classes (C_1, B_1) and (C_2, B_2) such that $\mathcal{A} = \mathcal{P} \setminus (C_1 \cup B_1 \cup B_2)$. Thus, $\mathcal{P} \setminus (C_1 \cup B_1 \cup B_2) \subseteq B$, and therefore $\mathcal{P} \subseteq (C_1 \cup B_1 \cup B_2 \cup B)$. This contradicts the definition of a full set. For small sets, the statement holds by definition.

P2: To see statement 2, take two full sets $\mathcal{A}_1 = \mathcal{P} \setminus (B_1 \cup C_1)$ and $\mathcal{A}_2 = \mathcal{P} \setminus (B_2 \cup C_2)$. Thus,

$$\mathcal{A}_1 \cap \mathcal{A}_2 = \mathcal{P} \setminus (B_1 \cup C_1 \cup B_2 \cup C_2).$$

By the definition of $Q^{(3,2)}(\mathcal{P}, \mathcal{Z})$, the remaining set is larger than any B_3 that occurs in a class $(B_3, C_3) \in \mathcal{Z}$.

P3, P4, P5: Statement 3 follows directly from the definition of a full and a big set. Statement 4 follows directly from the Statement 2, while statement 5 follows directly from statement 1.

P6: Let $\mathcal{A} \supseteq \mathcal{P} \setminus (B_1 \cup C_1)$ be a full set, where $(B_1, C_1) \in \mathcal{Z}$. If $\mathcal{A}' = \mathcal{P} \setminus \mathcal{A} \subseteq (B_1 \cup C_1)$ is a big set, then there exist $(B_2, C_2), (B_3, C_3) \in \mathcal{Z}$ such that $\mathcal{A}' \supseteq \mathcal{P} \setminus (B_2 \cup B_3 \cup C_2)$, and thus $(B_1 \cup C_1) \supseteq \mathcal{P} \setminus (B_2 \cup B_3 \cup C_2)$, which implies $\mathcal{P} \subseteq (B_1 \cup B_2 \cup B_3 \cup C_1 \cup C_2)$, which contradicts $\mathcal{Q}^{(3,2)}(\mathcal{P}, \mathcal{Z})$. In other words, if a full set of parties is removed from the set of all parties, the remaining set is not big, which proves the claim that each pair of a full and a big set intersects. $\square$

Note that in the conventional threshold model, if we assume the maximum number of corruptions (i.e., $n = 3t + 1$) we get $n - 2t = t + 1$, i.e., there is no difference between a *big set* and a *small set*.

4 Case Study

4.1 Transforming Efficient Reliable Broadcast

Reliable broadcast. Reliable broadcast [4] provides a way for a party to send a message to all other parties. It requires that all honest parties deliver the same set of messages and that this set includes all messages broadcast by honest parties. However, it makes no assumptions if the sender of a message is corrupted and does not guarantee anything about the order in which messages are delivered.

More precisely, given a distinct party (the sender) with input m on a protocol instance ID, a protocol satisfies reliable broadcast if the following three conditions are satisfied [1, 6]:

- If the sender is uncorrupted, then all the uncorrupted parties eventually complete the protocol.
- If any uncorrupted party completes the protocol, then all uncorrupted parties eventually complete the protocol
- If the uncorrupted parties complete the protocol, then they do so with a common output m'. Furthermore, if the sender is uncorrupted, then $m' = m$.

Protocol RBC
  input: message m
  upon *initialization*
    $s_m \leftarrow 0$
    $r_m \leftarrow 0$
  upon $(ID, j, \text{r-broadcast}, m)$
    send $(ID, j, \text{r-send}, m)$ to all parties
  upon *receiving* message $(ID, j, \text{r-send}, m)$ from P_l for the first time
    if $j = l$ **then**
      send $(ID, j, \text{r-echo}, m)$ to all parties.
  upon *receiving* message $(ID, j, \text{r-echo}, m)$ from P_l for the first time
    $s_m \leftarrow s_m + 1$
    if $s_m = n - t$ and $r_m < t + 1$ **then**
      send $(ID, j, \text{r-ready}, m)$ to all parties.
  upon *receiving* message $(ID, j, \text{r-ready}, m)$ from P_l for the first time
    $r_m \leftarrow r_m + 1$
    if $r_m = t + 1$, and $s_m < n - t$, **then**
      send $(ID, j, \text{r-ready}, m)$ to all parties.
    else if $r_m = n - t$,
      output $(ID, j, \text{r-deliver}, m)$

Fig. 1. The reliable broadcast protocol RBC with sender P_j [6].

A reliable broadcast protocol. Now consider the algorithm in Figure 1. It is an efficient reliable broadcast protocol by Cachin *et al.* [6]. Whenever the sender P_j wants to broadcast a message m he triggers the local event $(ID, j, \text{r-broadcast}, m)$. This results in a message being sent over the reliable channels to every other party. All parties then relay the received message to all other parties using *echo* messages. Upon receiving $n - t$ echos, a party sends a *ready* message to all other parties. Finally, upon receiving $n - t$ *ready* messages, the message m is output (delivered) by the protocol.

It is rather straightforward to verify the three conditions of reliable broadcast for the protocol (a formal proof can be found in [18]):

- If the sender is uncorrupted, then all the uncorrupted parties receive a *send* message and will thus send the corresponding *echo* message. Thus, at least $n - t$ such messages will be generated for message m, triggering each honest party to send a *ready* messages. Again, $n - t$ such messages will be created, causing all parties to terminate.
- If any uncorrupted party completes the protocol, then it received $n - t$ *ready* messages, at least $t+1$ of which originate from uncorrupted parties. Those $t+1$ messages are thus received by all uncorrupted parties parties. On receiving $t+1$ *ready* messages, an uncorrupted party will send out a *ready* message itself (unless it already did so). Thus, $n-t$ *ready* messages are sent by uncorrupted parties, causing all uncorrupted parties to complete the protocol.
- Suppose an uncorrupted party P_i has completed the protocol with message m and another uncorrupted party $P_{i'}$ has completed with $m' \neq m$. Then P_i must have received *ready* messages containing m from at least $t + 1$ uncorrupted parties; the same holds for $P_{i'}$ with m'. An uncorrupted party generates a *ready* message only if it has received $n-t$ *echo* messages containing m or $t+1$ *ready* messages already containing m. Thus, at least one uncorrupted party has sent a *ready* message containing m upon receiving $n - t$ *r-echo* messages; at most t of them are from corrupted parties. Similarly, some uncorrupted party must have received $n - t$ *echo* messages containing m'. Thus, there are at least $2(n - t) \geq n + t + 1$ *echo* messages and at least $n - t + 1$ among them from uncorrupted parties. But no uncorrupted party generates more than one such message by the protocol.

  If the sender is honest, then all *echo* messages sent by honest parties are on m, which by the same logic implies that m is the only possible output value.

Transforming the protocol. To transform the protocol, we first identify the counting variables that keep track of incoming messages. In this protocol, those variables are s_m and r_m. In the new protocol, these counters are replaced by sets $\mathcal{S}_m$ and $\mathcal{R}_m$ which collect messages. Incrementing the counter is replaced by adding the message to the corrseponding set. Although this seems to increase the memory overhead, this is not the case in real implementations, since the original protocol needs to keep a list of incoming messages as well. If it would not do so, it could be fooled into counting a message from a particular sender twice.

The next step is to transform all comparisons, such as $r_m = n - t$. First, we must identify what set the comparison value (i.e., the $n - t$) corresponds too. This can sometimes be performed automatically, but some intelligent verification against the protocol logic might be required here.

In our protocol, the substitution is rather simple. The value $n-t$ corresponds to a full set, while $t + 1$ corresponds to a small set. Note that since the sets are monotone, if $\mathcal{C}_m$ satisfies the definition of a full set, so do all supersets of $\mathcal{C}_m$. Thus, the expression

$$\textit{the set } \mathcal{C} \textit{ is a big set}$$

corresponds to the numeric expression

$$\textit{the value } c \textit{ is equal or larger than } n - t.$$

For the protocol, this is not the expression we need, as it would trigger the corresponding definition every time a new message arrives. We solve this by using the term

$$\textit{the set } \mathcal{C} \textit{ is a big set for the first time}$$

which corresponds to:

$$\textit{the value } c \textit{ is equal to } n - t$$

With these modifications, we obtain the new protocol, as shown in Figure 2.

Protocol HQ32-RBC
input: message m
upon *initialization*
  $\mathcal{S}_m \leftarrow \emptyset$
  $\mathcal{R}_m \leftarrow \emptyset$
upon $(ID, j, \text{r-broadcast}, m)$
  send $(ID, j, \text{r-send}, m)$ to all parties
upon *receiving* message $(ID, j, \text{r-send}, m)$ from P_l for the first time
  if $j = l$ **then**
    send $(ID, j, \text{r-echo}, m)$ to all parties.
upon *receiving* message $(ID, j, \text{r-echo}, m)$ from P_l for the first time
  $\mathcal{S}_m \leftarrow \mathcal{S}_1 \cup \{m\}$
  if $\mathcal{S}_1$ is a full set for the first time, and $\mathcal{R}_m$ is not a small set **then**
    send $(ID, j, \text{r-ready}, m)$ to all parties.
upon *receiving* message $(ID, j, \text{r-ready}, m)$ from P_l for the first time
  $\mathcal{R}_m \leftarrow \mathcal{R}_1 \cup \{m\}$
  if $\mathcal{R}_m$ is a small set for the first time, and $\mathcal{S}_m$ is not a full set **then**
    send $(ID, j, \text{r-ready}, m)$ to all parties.
  else if $\mathcal{R}_m$ is a full set **then**
    output $(ID.j, out, \text{r-deliver}, m)$

Fig. 2. The transformed protocol HQ3-RBC.

Transforming the proof. Modifying the proof is somewhat more difficult, as the counting arguments have to be matched against the properties of the different sets. We provide only the resulting proof here, which reduces all counting arguments in the original proof to the set properties shown in Lemma 1.

- If the sender is uncorrupted, then all the uncorrupted parties receive a *send* messages and will thus send the corresponding *echo* message. Thus, a full set of such messages will be generated for message m, triggering each honest party to send an *ready* messages. Again, a full set of such messages will be created, causing all parties to terminate.
- If any uncorrupted party completes the protocol, then it received a full set of *ready* messages, a big set of which originate from uncorrupted parties (by P3). The messages from this big set are thus received by all uncorrupted parties. Every big set is a small set (by P5), and on receiving a small set of *ready*

messages, an uncorrupted party will send out a *ready* message itself (unless it already did so). Thus, all uncorrupted parties send *ready* messages, causing them to complete the protocol.

– Suppose an uncorrupted party P_i has completed the protocol with message m and another uncorrupted party $P_{i'}$ has completed with $m' \neq m$. Then P_i must have received *ready* messages containing m from a small set of uncorrupted parties; the same holds for $P_{i'}$ with m'. An uncorrupted party generates a *ready* message only if it has received $n - t$ *echo* messages containing m or a small set of *ready* messages already containing m. By P1, at least one uncorrupted party has sent an *ready* message containing m upon receiving a full set of *r-echo* messages. Similarly, some honest party must have received a full set of *echo* messages containing m'. As an uncorrupted party sends only one *echo* message, and by P2, this is a contradiction.

  If the sender is honest, then all *echo* messages sent by honest parties are on m, which by the same logic implies that m is the only possible output value.

4.2 Discussion

Different models. We have performed the case study in the asynchronous message passing model with probabilistic protocol properties. We note that our method is not tied to this model. The general technique used here can also be used with different models such as ones with broadcast channels and ones with deterministic protocol properties together with synchrony abstractions like failure detectors. While the properties proved in Lemma 1 may not be sufficient to transform those protocols, the gaps should be reasonably easy to fill.

Generalizing the Cryptographic Primitives. While most fault tolerant protocols are based on counting arguments and thus can be relatively easily transformed, cryptographic primitives — such as threshold signature schemes [25] or a coin tossing scheme [5] are a little more difficult to handle. The usual technique of proving such a protocol correct is by using the *simulator technique* — provided an adversary that can break the real scheme, we can provide him with a specially constructed mock-up simulation of the scheme that the adversary cannot distinguish from a real protocol run, but the result he obtains by breaking our simulation can be used to solve an algebraic problem that is assumed to be hard (e.g., computing a discrete logarithm). These proofs are significantly more sensitive to changes in the model, and thus are not as easily transformed into the adversary structure world.

One solution is to (virtually) run a separate protocol instance for each allowed coalition. As those protocol instances are independent, they do not interfere with each other, and the classical security proofs survive. However, the number of allowed instances can be rather large; as they all have to run in parallel, the effort in terms of computation and communication may quickly become unreasonable high. Techniques to reduce the complexity by slightly limiting the flexibility, as introduced in the next section, may help here. Nevertheless, security proofs of cryptographic protocols do need some special attention.

5 Managing Adversary Structures

The flexibility provided by the concept of adversary structures does not come for free. The number of possible coalitions can be expected to be exponential in the number of parties in most settings, the overhead required to identify, store and compute on these sets can easily become the bottleneck. Furthermore, once cryptographic primitives are used, any computation requiring exponential time in one of the protocol parameters can invalidate the security proofs.

Therefore, in this section, we define means to restrict the flexibility offered by the adversary structures in some meaningful way, i.e., in a way that makes the sets easier to manage, while mirroring failure dependencies that occur in the real world.

5.1 Attribute based Adversary Structures

For practical purposes, the flexibility provided by the general adversary structures might be too much; for n parties, there are 2^n different sets of parties that have to be managed. This might be both too much for theoretical proofs (if cryptographic primitives are used) and for practical purposes. Therefore, one might want to limit the flexibility of the structures a bit to increase the manageability. To reach this, we introduce an attribute based adversary structure.

There are l attributes that characterize a party. We impose no restriction on how the attributes are defined; in practice, it makes sense to use attributes such that parties with the same attribute are more likely to fail together than parties with different attributes. For example, the operating system or the geographical location of a party would be suitable attributes.

For every attribute, there are several possible values. For example, the attribute "Operating System" can have the values "Linux", "Windows", "OS/2", "MAC OS" and "BeOS". Let n_i be the number of possible values of attribute a_i. During the initialization of the system, for each attribute a_i two thresholds b_i and c_i are fixed, such that $n_i > 3b_i + 2c_i$. These parameters define the balance between tolerable Byzantine failures and crashes. For each attribute a_i, the adversary may choose b_i values and (byzantinely) corrupt all parties for which a_i has one of these values. Similarly, the adversary may choose c_i values such that and all parties for which a_i has one of these values are subject to crash failures.

We demand that for each possible combination of attribute values, there is one party with that combination.

Definition 2. *Given n parties $P_1, \ldots, P_n$ and l attributes $a_1, \ldots, a_l$, let $\mathcal{V}_i$ denote the set of possible values a_i can have. An attribute based adversary structure $\mathcal{Z}$ is the set of all classes (B, C) such that for all attributes a_i, the parties in set B cover at most b_i values of a_i, and the parties in the set C cover at most c_i values of a_i, where $3b_i + 2c_i < |\mathcal{V}_i|$.*

For all attributes a_i, the number values of a_i (i.e., $|\mathcal{V}_i|$) is also denoted n_i.

Lemma 2. *The attribute based adversary structure $\mathcal{Z}$ satisfies $Q^{(3,2)}(\mathcal{P}, \mathcal{Z})$*

Proof. Recall that an adversary structure $\mathcal{Z}$ over the set $\mathcal{P}$ of parties satisfies $Q^{(3,2)}(\mathcal{P}, \mathcal{Z})$ if

$$\forall (B_1, C_1), (B_2, C_2), (B_3, C_3) \in \mathcal{Z} : B_1 \cup B_2 \cup B_3 \cup C_1 \cup C_2 \neq \mathcal{P}.$$

Suppose that there is an attribute based adversary structure $\mathcal{Z}$ that does not satisfy $Q^{(3,2)}(\mathcal{P}, \mathcal{Z})$. Then, there are sets $(B_1, C_1), (B_2, C_2), (B_3, C_3)$ such that $B_1 \cup B_2 \cup B_3 \cup C_1 \cup C_2 = \mathcal{P}$. Fix an attribute a_i. Now each of the sets B_1, B_2 and B_3 covers at most b_i values of the attribute, while each of the sets C_1 and C_2 covers at most c_i of those values. As the total number n_i of possible values for a_i is bigger than $3b_j + 2c_j$, this implies that there is some value of a_i that is not covered by the parties in $B_1 \cup B_2 \cup B_3 \cup C_1 \cup C_2$.

As this holds for all attributes $a_1, \ldots, a_l$, and for each combination of attribute values there is some party for which this combination applies, there is at least one party that is not in $B_1 \cup B_2 \cup B_3 \cup C_1 \cup C_2$, which is a contradiction. □

As an example, consider a distributed system with two attributes. The attribute a_1 is the operating system with the values

$$V_1 = \{OS1, OS2, OS3, OS4\}$$

and the attribute a_2 is the geographical location with the values

$$V_2 = \{\text{Aland,Bkistan,United C, Dien}\}.$$

	Aland	Bkistan	United C	Dien
OS1	P_1	P_2	P_3, P_4, P_5	P_6
OS2	P_7	P_8	P_9	P_{10}, P_{11}
OS3	P_{12}	P_{13}	P_{14}	P_{15}
OS4	P_{16}	P_{17}	P_{18}	P_{19}

Table 1. Example attribute-based adversary structure.

Assume there are 19 parties with the attributes given in Table 1. Let $b_1 = b_2 = 1, c_1 = c_2 = 0$. The class $(B, C) = (\{P_1, P_9, P_{13}\}, \emptyset)$ is not an element of $\mathcal{Z}$, as it is not possible for the adversary to corrupt these parties by corrupting less than two locations and less than two operating systems. The class $(B, C) = (\{P_3, P_4, P_5, P_7, P_8, P_9, P_{10}, P_{11}, P_{14}, P_{18}\}, \emptyset)$ however is in $\mathcal{Z}$, as the adversary can corrupt these parties by corrupting one out of four locations (being United C) and one out of four operating systems (being OS 2).

5.2 Probability Based Structures

In a first iteration, this approach allows us to weigh individual parties; each party gets an individual failure probability, and the traditional $n > 3t$ is replaces by a fixed probability p with which the system is allowed to fail. The adversary coalitions are then simply all maximal sets such that the probability of all parties in that set failing is smaller than p.

While the sets can be defined automatically in a precomputation step, their number may still be rather large; it is not yet clear if an easy optimization exists that allows for an efficient way to model those sets.

Also, this approach is a huge step backwards in that failure dependencies are ignored. It is of course possible to add those to the computation of the sets. However, this leads to a potentially exponential set of dependencies (as each parties failure probability may depend on all other partys' failures). Thus, we merely get a different, potentially more suitable definition of out sets, but may still be left with a large management-overhead.

Acknowledgments

We wish to thank Zinaida Benenson for her comments on a previous version of this paper.

References

1. M. Bakes and C. Cachin. Reliable broadcast in a computational hybrid model with byzantine faults, crashes, and recoveries. Technical Report RZ 3466, IBM Research, November 2002.
2. J. Benaloh and J. Leichter. Generalized secret sharing and monotone functions. In S. Goldwasser, editor, *Proc. CRYPTO 88*, pages 27–36. Springer-Verlag, 1988. Lecture Notes in Computer Science No. 403.
3. A. Birolini. *Reliability Engineering: Theory and Practice*. Springer-Verlag, third edition, 1999.
4. G. Bracha and S. Toueg. Asynchronous consensus and broadcast protocols. *Journal of the ACM*, 32(4):824–840, Oct. 1985.
5. C. Cachin, K. Kursawe, and V. Shoup. Random oracles in constantinople: Practical asynchronous Byzantine agreement using cryptography. In *Proceedings of the Symposium on Principles of Distributed Computing*, pages 123–132, Portland, Oregon, 2000.
6. C. Cachin, K. Kursawe, F. Petzold, and V. Shoup. Secure and efficient asynchronous broadcast protocols. In *Advances in Cryptology – CRYPTO ' 2001*, Lecture Notes in Computer Science. International Association for Cryptologic Research, Springer-Verlag, 2001.
7. M. Correia, L. C. Lung, N. F. Neves, and P. Veríssimo. Efficient Byzantine-resilient reliable multicast on a hybrid failure model. In *Proc. of the 21st Symposium on Reliable Distributed Systems*, Suita, Japan, Oct. 2002.
8. M. J. Fischer, N. A. Lynch, and M. S. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM*, 32(2):374–382, Apr. 1985.
9. M. Fitzi, M. Hirt, and U. Maurer. General adversaries in unconditional multi-party computation. In *Proceedings of ASIACRYPT*, pages 232–246, 1999.
10. M. Fitzi and U. Maurer. Efficient byzantine agreement secure against general adversaries. In *Proceedings of the International Symposium on Distributed Computing (DISC)*, volume 1499 of *Lecture Notes in Computer Science*. Springer-Verlag, 1998.
11. F. C. Gärtner. Byzantine failures and security: Arbitrary is not (always) random. Technical Report IC/2003/20, Swiss Federal Institute of Technology (EPFL), School of Computer and Communication Sciences, Lausanne, Switzerland, Apr. 2003.
12. O. Goldreich. Secure multi-party computation. Internet: `http://www.wisdom.weizmann.ac.il/~oded/pp.html`, 2002.
13. M. Hirt and U. Maurer. Complete characterization of adversaries tolerable in secure multi-party computation (extended abstract). In *Proceedings of the Sixteenth Annual ACM Symposium on Principles of Distributed Computing*, pages 25–34, Santa Barbara, California, 21–24 Aug. 1997.
14. M. Ito, A. Saito, and T. Nishizeki. Secret sharing scheme realizing general access structure. In *Proceedings IEEE Globecom '87*, pages 99–102. IEEE, 1987.
15. F. P. Junqueria and K. Marzullo. Synchronous consensus for dependent process failures. In *Proceedings of the 23rd International Conference on Distributed Computing Systems (ICDCS'03)*, 2003.
16. A. W. Krings and M. A. McQueen. A Byzantine resilient approach to network security. In *Digest of FastAbstracts of the 29th International Symposium on Fault-Tolerant Computing (FTCS-29)*, Madison, Wisconsin, June 1999. `http://www.crhc.uiuc.edu/FTCS-29/pdfs/krings.pdf`.

17. K. Kursawe. Asynchronous Byzantine group communication. In *Proceedings of the 21st IEEE Symposium on Reliable Distributed Systems (SRDS), Workshop on Reliable Peer-to-Peer Distributed Systems*, pages 352–357, Osaka, Japan, Oct. 2002. IEEE Computer Society Press.
18. K. Kursawe. *Distributed Trust*. PhD thesis, University of Saarbrücken, March 2002.
19. L. Lamport, R. Shostak, and M. Pease. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, July 1982.
20. D. Malkhi and M. Reiter. Byzantine quorum systems. In *Proceedings of the 29th Annual ACM Symposium on the Theory of Computing (STOC '97)*, pages 569–578, New York, May 1997. Association for Computing Machinery.
21. J. Oberg. NASA's big push for the space station. *IEEE Spectrum*, 37(11):49–54, Nov. 2000.
22. M. Pease, R. Shostak, and L. Lamport. Reaching agreements in the presence of faults. *Journal of the ACM*, 27(2):228–234, Apr. 1980.
23. D. Powell. Failure mode assumptions and assumption coverage. In D. K. Pradhan, editor, *Proceedings of the 22nd Annual International Symposium on Fault-Tolerant Computing (FTCS '92)*, pages 386–395, Boston, MA, July 1992. IEEE Computer Society Press.
24. M. K. Reiter. A secure group membership protocol. *IEEE Transactions on Software Engineering*, 22(1):31–41, Jan. 1996.
25. V. Shoup. Practical threshold signatures. In B. Preneel, editor, *Advances in Cryptology: EUROCRYPT 2000*, Lecture Notes in Computer Science, pages 207–220. Springer, 2000.
26. G. J. Simmons. How to (really) share a secret. In S. Goldwasser, editor, *Proc. CRYPTO 88*, pages 390–449. Springer-Verlag, 1988. Lecture Notes in Computer Science No. 403.
27. J. H. Wensley, L. Lamport, J. Goldberg, M. W. Green, K. N. Levitt, P. M. Melliar-Smith, R. E. Shostak, and C. B. Weinstock. SIFT: Design and analysis of a fault-tolerant computer for aircraft control. *Proceedings of the IEEE*, 66(10):1240–1255, Oct. 1978.
28. J. Yin, J.-P. Martin, A. Venkataramani, L. Alvisi, and M. Dahlin. Byzantine fault-tolerant confidentiality. In *Proceedings of the International Workshop on Future Directions in Distributed Computing*, pages 12–15, June 2002.
29. L. Zhou, F. B. Schneider, and R. van Renesse. COCA: A secure distributed on-line certification authority. *ACM Transactions on Computer Systems*, 20(4):329–368, Nov. 2002.

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-Red^+ - 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
1998-08 * H. Aust: Sprachverstehen und Dialogmodellierung in natürlichsprachlichen Informationssystemen
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
2000-04 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

* These reports are only available as a printed version.
Please contact biblio@informatik.rwth-aachen.de to obtain copies.