A Theory of Fault Recovery for Component-Based Models

Borzoo Bonakdarpour
School of Computer Science
University of Waterloo, Canada
borzoo@cs.uwaterloo.ca

Marius Bozga
VERIMAG, France
marius.bozga@imag.fr

Gregor Gössler
INRIA Grenoble – Rhône-Alpes, France
gregor.goessler@inria.fr

***Abstract*—This paper introduces a theory of *fault recovery* for component-based models. In our framework, a model is specified in terms of a set of atomic components that are incrementally composed and synchronized by a set of glue operators. We define what it means for such models to provide a recovery mechanism, so that the model converges to its normal behavior in the presence of faults. We identify *corrector* (atomic or composite) components whose presence in a model is essential to guarantee recovery after the occurrence of faults. We also formalize component-based models that effectively *separate* recovery from functional concerns.**

***Keywords*-Fault-tolerance; recovery; component-based; separation of concerns; modularity; BIP**

I. Introduction

Fault-tolerance has always been one of the active subjects of research in design and implementation of *dependable* computing systems. Intuitively, tolerating faults involves providing a system with the means to handle unexpected defects, so that the system meets its specification even in the presence of faults. In this context, the notion of specification may vary depending upon the guarantees that the system must deliver in the presence of faults. Such guarantees can be broadly characterized by *safety* and *liveness* properties. For instance, dependable mission-critical systems often employ monitoring or control techniques to ensure safety properties in the presence of faults, and, provide a *recovery* mechanism to meet liveness properties, if the system reaches an unexpected state.

The concept of fault-tolerance as described above addresses the overall behavior of the system and is independent of the structure the system. In order to associate fault-tolerance properties with the structure of a system and study their interdependence, one has to focus on a specific methodology. *Component-based* approach is a successful divide-and-conquer technique for designing and implementing large systems as well as for reasoning about their correctness. Ideally, in this approach, a system is designed incrementally by composing smaller components, each responsible for delivering a certain set of tasks to separate different concerns. Thus, component-based design and analysis of fault-tolerant systems is highly desirable in order to achieve systematic modularization of such systems. For instance, fault tolerance is becoming one of the key issues for efficient multi-core programming [1]. The likelihood of fault occurrences is in fact proportional with the number of cores available in the underlying platform. Traditional fault-detection and recovery mechanisms e.g., based on restore points and rollback, scale poorly and may even become unusable for many cores. That is, significant amount of core time and power are spent on fault recovery instead of performing useful computation. Such scenarios clarify the need for systematic and modularized approaches for fault recovery in large scale systems.

We believe that we currently lack a formal approach that rigorously relates a component-based methodology with fault-tolerance concerns. With this motivation, in this paper, we propose a novel formal framework for component-based design and analysis of *non-masking* models [2] where fault recovery and, hence, liveness is guaranteed in the presence of faults. We use the semantics of the BIP (Behavior, Interaction, Priority) framework [3], [4] to specify components and their composition. In BIP, the *behavior* of an atomic component is specified by a labeled transition system. A model (i.e., a composite component) is represented as the composition of a set of atomic components by using two types of operators: *interactions* describing synchronization constraints between components, and *priorities* to specify scheduling constraints.

The elegance of BIP is in its expressiveness and ability to model a wide range of computing systems using its sequential operational semantics. Moreover, given a BIP model, the tool chain can automatically generate a stand-alone, distributed, real-time, multi-threaded, or synchronous C++ implementation that is correct-by-construction (i.e., by preserving observational semantics of the original model) [5]–[9]. Thus, our results in this paper can be applied in model-based design and analysis of component-based fault recovery for a wide range of settings such as in distributed systems.

Our contributions are as follows:

- We formally define non-masking fault-tolerance for atomic and composite components based on their observational behavior. This is different from the approach in [2], where fault-tolerance is defined based on reachability of predicates.
- We define *corrector* components that establish a desirable observational behavior and show that the necessary condition for a composite component to be non-masking is to contain atomic or composite correctors.
- We introduce the notion of *pure correctors* that only exhibit recovery behavior and do not participate in functional tasks of a composite component. We show that models containing pure correctors can effectively separate functional from recovery concerns and, hence, can be

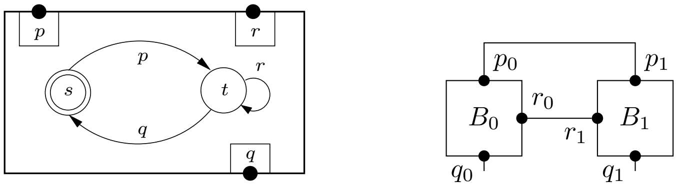

Figure 1. A BIP atomic component (left); a BIP composite component constructed from two copies of the atomic component.

compositionally verified.

As argued above, since a BIP model can be automatically transformed into (for example) a distributed implementation, a non-masking model with correctors or pure correctors can also be viewed as a system with a distributed fault recovery mechanism. We emphasize that the presented method in this paper is not limited to BIP and can be applied to any framework with two characteristics: (1) a component's behavior is specified in terms of an automaton or Petri net, and (2) components interact by synchronizing on labeled transitions using multi-party rendezvous or broadcast primitives. Thus, our method is applicable to most process algebras and synchronous languages as well.

II. Basic Semantic Models of BIP

Atomic Components We define *atomic components* as transition systems with a set of ports labeling individual transitions. These ports are used for communication between components.

Definition 1 *An* atomic component B *is a labeled transition system represented by a tuple* $(Q, P, \rightarrow, q^0)$ *where*

- Q *is a set of* states*,*
- P *is a set of* communication ports*,*
- $\rightarrow \subseteq Q \times P \cup \{\tau\} \times Q$ *is a set of* transitions *including* observable *transitions labeled by ports, and* unobservable τ *transitions, and*
- $q^0 \in Q$ *is the initial state.*

For any pair of states $q, q' \in Q$ and a port $p \in P \cup \{\tau\}$, we write $q \xrightarrow{p} q'$ iff $(q, p, q') \in \rightarrow$. When the label is irrelevant, we simply write $q \rightarrow q'$. Similarly, $q \xrightarrow{p}$ means that there exists $q' \in Q$, such that $q \xrightarrow{p} q'$. In this case, we say that p is *enabled* in state q. Figure 1 (left) shows an atomic component B, where $Q = \{s, t\}$, $q^0 = s$, $P = \{p, q, r\}$, and $\rightarrow = \{(s, p, t), (t, q, s), (t, r, t)\}$.

A *trace* of a component $B = (Q, P, \rightarrow, q^0)$ is a finite or infinite sequence of ports $\pi = p_0 p_1 p_2 \cdots$, such that for $\forall i \geq 0$:

1) $p_i \in P \cup \{\tau\}$,
2) there exists state sequence $q_0 q_1 \cdots$, such that:
   - $q_0 = q^0$ (i.e., q_0 is the initial state), and
   - $q_0 \xrightarrow{p_0} q_1 \xrightarrow{p_1} q_2 \cdots$

For a trace $\pi = p_1 \cdots p_n$, by $q \xrightarrow{\pi} q'$, we denote $\exists q_1 \cdots q_{n-1} \ : \ q \xrightarrow{p_1} q_1 \xrightarrow{p_2} \cdots \xrightarrow{p_{n-1}} q_{n-1} \xrightarrow{p_n} q'$.

Interaction For a given system built from a set of m atomic components $\{B_i = (Q_i, P_i, \rightarrow_i, q_i^0)\}_{i=1}^m$, we assume that their respective sets of ports are pairwise disjoint, i.e., for any two $i \neq j$ from $\{1..m\}$, we have $P_i \cap P_j = \emptyset$. We can therefore define the set $P = \bigcup_{i=1}^m P_i$ of all ports in the system. An *interaction* is a set $a \subseteq P$ of ports. When we write $a = \{p_i\}_{i \in I}$, we suppose that for $i \in I$, $p_i \in P_i$, where $I \subseteq \{1..m\}$.

Definition 2 *A* composite component *(or simply* model*) is defined by a composition operator parametrized by a set of interactions* $\gamma \subseteq 2^P$. $B \stackrel{def}{=} \gamma(B_1 \dots B_m)$*, is a transition system* $(Q, \gamma, \rightarrow, q^0)$*, where* $Q = Q_1 \times ... \times Q_m$, $q^0 = (q_1^0 \dots q_m^0)$*, and* $\rightarrow$ *is the least set of transitions satisfying the rule*

$$\frac{a = \{p_i\}_{i \in I} \in \gamma \qquad \forall i \in I.\ q_i \xrightarrow{p_i}_i q_i' \qquad \forall i \notin I.\ q_i = q_i'}{(q_1, \dots, q_m) \xrightarrow{a} (q_1', \dots, q_m')}$$

In a composite component, τ*-transitions do not synchronize but execute in an interleaving fashion.*

The inference rule in Definition 2 says that a composite component $B = \gamma(B_1, \dots, B_m)$ can execute an interaction $a \in \gamma$, iff for each port $p_i \in a$, the corresponding atomic component B_i can execute a transition labeled with p_i; the states of components that do not participate in the interaction stay unchanged.

In general, one can view a model $\gamma(\mathcal{B}_1, \mathcal{B}_2)$, where $\mathcal{B}_1$ and $\mathcal{B}_2$ are two sets of atomic components, as one component whose set of transitions is γ. Thus, $\gamma(\mathcal{B}_1, \mathcal{B}_2)$ denotes the composite component glued by γ, and, γ denotes the set of interactions of this composite component.[1]

Figure 1 (right) illustrates a composite component $\gamma(B_0, B_1)$, where both B_0 and B_1 are identical to the component in Figure 1 (left) and $\gamma = \{\{p_0, p_1\}, \{r_0, r_1\}, \{q_0\}, \{q_1\}\}$.

Similar to traces of an atomic component, a trace of a composite component $B = \gamma(B_1, \dots, B_n)$ is a finite or infinite sequence of interactions $a_0 a_1 a_2 \cdots$, such that for all $i \geq 0$ (1) a_i is an interaction of γ, and (2) there exists states $q_0 q_1 \cdots$ of B, such that $q_0 = q^0$ and $q_0 \xrightarrow{a_0} q_1 \xrightarrow{a_1} q_2 \cdots$.

III. Fault Model and Fault Recovery

In this section, we describe our fault model and the concept of fault recovery in the context of the component-based framework described in Section II.

A. Fault Model

Let $B = (Q, P, \rightarrow, q^0)$ be an atomic component. We classify the observable transitions in $\rightarrow$ into the following three pairwise disjoint sets:

- A set $\rightarrow_n$ of observable *normal* transitions that embodies the normal execution of the component.
- A set $\rightarrow_f$ of observable *fault* transitions that expresses the faulty behavior of the component.

[1]In practice, atomic components are extended with variables. Transitions and interactions are associated with guards on variables. Also, interactions can transfer data using an update function on variables bound to ports.

	n	r	f
n	N	R	F
f	F	F	F
r	R	R	F

Table I
INTERACTION TYPES BASED ON THE PARTICIPATING TRANSITIONS

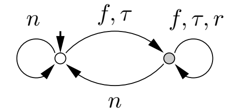

Figure 2. Non-masking atomic component; the gray state models an unstable period.

- A set $\rightarrow_r$ of observable *recovery* transitions that restore the normal behavior of the component or help other components to restore their normal behavior through participating in cross-component interactions.

Finally, $\rightarrow_\tau$ (i.e., τ-transitions of B) are *unobservable fault* transitions that expresses the local faulty behavior of the component.

Notation. Let $B = (Q, P, \rightarrow, q^0)$ be an atomic component. By $\rightarrow_x$, we denote the union of transitions of the types in x, where $x \in 2^{\{n,f,\tau,r\}}$. By B_x, we mean the component $(Q, P, \rightarrow_x, q^0)$.

Intuitively, a component normally executes transitions in $\rightarrow_n$. However, faults in $\rightarrow_{f,\tau}$ may perturb the state of B to a state that may or may not be reachable by other transitions and in particular, $\rightarrow_n$. We emphasize that the occurrence of a transition in $\rightarrow_{f,\tau}$ is not a bad thing (such as a failure or error) by itself, i.e., it is merely a state perturbation. We also note that such representation is possible notwithstanding the type of faults (be they stuck-at, crash, fail-stop, timing, performance, Byzantine, message loss, etc.), the nature of the faults (be they permanent, transient, or intermittent), or the ability of the program to observe the effects of the faults (be they detectable or undetectable).

Definition 3 *We say that $B = (Q, P, \rightarrow, q^0)$ is a* faulty component *if $\rightarrow_{f,\tau}$ is nonempty.*

Now, let $B = \gamma(B_1, \dots, B_m)$ be a composite component. Observe that in an interaction $a = \{p_i\}_{i \in I}$ in γ, for any two $j \neq k$ in $\{1..m\}$, transitions $\xrightarrow{p_k}_k$ and $\xrightarrow{p_j}_j$ may belong to any of the above classes of transitions of their respective components. Thus, we define the type of interactions of a composite component as follows (see Table I):

- Following Definition 2, an unobservable fault does not participate in an interaction; i.e., the corresponding component only takes a silent move from one state to another without synchronizing with other components.
- Otherwise, the type of the interaction is determined by the greatest type of the participating transitions in the total order $n < r < f$.

For example, the type of an interaction consisting only normal transitions is also normal, and, the type of an interaction consisting transitions of types f and r is faulty. In other words, occurrence of a fault transition in an atomic component B_i, implies occurrence of a fault in the composite component that contains B_i. Thus, we partition interactions of $B = \gamma(B_1, \dots, B_n)$ into γ_N, γ_R, and γ_F.

We emphasize that such representation is possible notwithstanding the type of the faults (be they stuck-at, crash, fail-stop, timing, performance, Byzantine, message loss, etc.), the nature of the faults (be they permanent, transient, or intermittent), or the ability of the program to observe the effects of the faults (be they detectable or undetectable). In fact, representation of faults in transition systems has been explored extensively. For instance, message loss, performance, and crash faults have been modeled in [2], [10], undetectable, Byzantine, and fail-stop faults in [11], and timing faults in [12], [13]. We refer the reader to [14] for a survey on using transition systems to model faults.

We also note that since our focus is on model-based analysis of fault recovery, in our framework the set of faults needs to be provided. Having said that, one can model the effect of *unanticipated* faults by specifying the set of faults that start from any state of a component and can reach any state of the component. This is in fact the core idea in self-stabilizing systems [15], where faults can perturb the system to any arbitrary state. Modeling self-stabilizing systems and unanticipated faults in BIP have been studied in [9]. All results in this paper hold regardless of the set of faults in a model.

B. Fault Recovery

Arora and Gouda [2] formally define the *levels of fault-tolerance* based on combinations of meeting safety and liveness in the presence of faults. In this paper, our focus is on *non-masking* fault-tolerance. Non-masking systems are only concerned with ensuring liveness in the presence of faults by guaranteeing deadlock-freedom through providing a finite-step *recovery* mechanism; i.e., the system always eventually reaches a good state even in the presence of faults. However, in such systems, when faults occur, safety may be temporarily violated during recovery, but not after the systems reaches a good state.

1) Non-masking Atomic Components: We characterize fault recovery of an atomic component by ω-regular expressions based on the behavior of transition types identified in Section III-A. For example, the ω-regular expression $f^* r n^\omega$ is the set of infinite traces of an atomic component where a finite number of observable fault transitions is followed by a recovery transition and an infinite sequence of normal transitions.

Definition 4 *We say that $B = (Q, P, \rightarrow, q^0)$ is a* non-masking atomic component *iff its set of traces satisfies the following ω-regular expression: $[n^*((f + \tau)r^*)^* n]^\omega$.*

The intuitive description of Definition 4 is the following (also see Figure 2 for an automaton-based description). If no faults occur, the program executes only normal transitions (i.e., the left state in Figure 2). If fault(s) occur, the component reaches a state from where execution of normal transitions may not be possible (the gray state in Figure 2). In this case, we say that the component enters a finite *unstable* period. After a finite number of steps, the component recovers and only executes normal transitions again. Also, note that according to Definition 4 the number of occurrences of faults in each unstable period is finite.

Observe that a non-masking component does not deadlock in the absence or presence of faults. Also, a non-masking component can use any recovery transition, be it safe or unsafe, to converge to its normal behavior. Thus, a non-masking component may temporarily violate safety in the presence of faults, but it eventually returns to its normal behavior, from where satisfaction of safety is assumed. Some network protocols are non-masking fault-tolerant. For instance, in some protocols a packet has to eventually reach the destination, but duplicate packets may be delivered.

2) Non-masking Composite Components: We characterize fault recovery of a composite component based on observational behavior of interaction types identified earlier; i.e., γ_N, γ_F, and γ_R. There is, however, an important difference between non-masking atomic and composite components. In a composite component, if a fault occurs in an atomic component, the fault may force a set of components to execute transitions other than their normal transitions, while a set of other atomic components can resume their normal operation. Thus, unlike non-masking atomic components, non-masking composite component may as well exhibit normal interactions in their unstable period.

Definition 5 *We say that $B = \gamma(B_0 \cdots B_m)$ is a* non-masking composite component *iff:*

1) *Its set of traces satisfies the following ω-regular expression: $(N^*(F + R + N)^*N)^\omega$.*
2) *If a trace prefix of B ends with NR, then there exists an atomic component B_i, $0 \leq i \leq m$, such that projection of the prefix on B_i results in a local prefix that ends with $n\tau^+$.*

Intuitively, in Definition 5, traces of a non-masking composite component behave similarly to those of non-masking atomic components, except that normal interactions can also occur during the unstable period[2]. Moreover, in a non-masking composite component if a recovery interaction occurs immediately after a normal interaction, then we require the existence of an atomic component in which an unobservable fault causes the execution of the recovery interaction.

Notice that in Definition 5, we do not require that atomic components of a non-masking composite component should be non-masking as well. This is because we would like our definition to cover cases where an atomic component is not subject to faults locally, but it participates in recovery interactions in the composite component that contains other faulty atomic components.

[2]If such behavior is not desirable (e.g., to assign all resources to speed-up recovery), one can apply the priority mechanism in BIP to exclude normal interactions while a part of the system is recovering.

IV. Corrector Components and Component-based Recovery

In this section, we present our theory of component-based recovery in non-masking models. To this end, we first define the notion of *corrector* components in Subsection IV-A Then, in Subsection IV-B, we show that the necessary condition for a model to be non-masking is to contain corrector components for faulty components.

A. Correctors

The concept of correctors is inspired by the work in [16], [17]. The definition of correctors in [16], [17] is based on correction of an invariant predicate, no matter how it is reached. Our definition of correctors in this paper is based on observation of recovery and normal transitions/interactions in atomic/composite components. In other words, our notion of correctors is tailored for component-based models.

Roughly speaking, a corrector is concerned with two types of transitions: recovery and normal. A corrector component ensures two properties: (1) once a fault occurs, the component recovers and eventually exhibits normal behavior, and (2) execution of normal transitions eventually stabilizes (i.e., once normal behavior is restored the component behaves normally unless another fault occurs). We now formally define the notion of corrector components.

Definition 6 *Let $B = (Q, P, \rightarrow, q^0)$ be an atomic component. We say that B is a* corrector *for the set $\rightarrow_n$ of normal transitions, if there exists the set $\rightarrow_r$ of recovery transitions, such that $\rightarrow_n \cap \rightarrow_r = \emptyset$ and any trace $\pi = p_0 p_1 \cdots$, where $p_i \in P$, satisfies the following two conditions:*

1) (Progress) *If there exists $i \geq 0$, such that transition $q_i \xrightarrow{p_i} q_{i+1}$ is not in $\rightarrow_{r,n}$, then there exists $j \geq i+1$, such that $q_j \xrightarrow{p_j} q_{j+1}$ is in $\rightarrow_n$.*
2) (Weak Stability) *For all $i \geq 0$, if $q_i \xrightarrow{p_i} q_{i+1}$ is in $\rightarrow_n$, then $q_{i+1} \xrightarrow{p_{i+1}} q_{i+2}$ is either (1) in $\rightarrow_n$, or (2) not in $\rightarrow_{r,n}$.*

A *composite corrector component* is defined in the same fashion for interactions of types R and N. A composite component may be a corrector for a set of transitions local to one of its atomic components. Such correctors are of interest where a faulty component achieves recovery to its normal behavior by the help of a set of other components.

Formally, let $B = \gamma(B_0 \cdots B_m)$ be a composite component and $B_i = (Q_i, P_i, \rightarrow_i, q_i^0)$, $0 \leq i \leq m$, be an atomic component. We say that B is a corrector for the set $\rightarrow_{i_n}$ of normal transitions of B_i if and only if by projecting any trace $\pi = a_0 a_1 \cdots$, where $a_j \in \gamma$ for all j, on component B_i and obtaining trace π', there exists recovery transitions $\rightarrow_{i_r}$, such that $\rightarrow_{i_r}$ and $\rightarrow_{i_n}$ satisfy Progress and Weak Stability.

B. Containment of Correctors in Non-masking Models

In this subsection, we show that the necessary condition for a model to be non-masking is to contain a subset of components that act a corrector for each components that is subject to faults. Recall that in Definition 5, we allowed components that do not interact with a faulty component to continue their normal behavior, while interacting components with the faulty component recover. We note that in our model, fault propagation is possible in the sense that components that do not interact with a faulty component may get involved in achieving recovery as well. In order to ensure that recovery makes progress in non-masking models, we assume that composite components are *weakly fair*.

Assumption 1 *Let $B = \gamma(B_0 \cdots B_m)$ be a composite component. We assume that if an interaction $\alpha \in \gamma$ is continuously enabled in a trace $\pi = a_0 a_1 \cdots$, then there exists $i \geq 0$, such that $a_i = \alpha$.*

Assumption 1 is necessary to show containment of correctors in non-masking models. The containment theorem is the following.

Theorem 1 *Let $B = \gamma(B_0 \cdots B_m)$ be a non-masking composite component. For each faulty atomic component $B_l = (Q_l, P_l, \rightarrow_l, q_l^0)$, where $0 \leq l \leq m$, there exists a set $\mathcal{C} \subseteq \{B_0 \cdots B_m\}$ of atomic components, such that $\gamma(B_l, \mathcal{C})$ is a corrector for $\gamma_N(B_l, \mathcal{C})$.*

V. Separation of Functional and Recovery Concerns

In this section, we study separation of functional and recovery concerns in non-masking models. To this end, we identify two types of components that are responsible for performing functional or recovery tasks independently. We call such components *pure* components.

Roughly speaking, a *purely functional component* is one that is responsible for performing normal computational tasks of the containing composite component. Such a component may be subject to faults, but is not concerned with achieving fault recovery. On the contrary, a *pure corrector* is a component that only helps a system restoring the normal through achieving recovery and it does not perform any functional tasks.

Definition 7 *Let $B = (Q, P, \rightarrow, q^0)$ be an atomic component. We say that B is* purely functional *iff its set of traces satisfies the ω-regular expression: $((n+\tau)^*(f+r)n)^\omega$.*

Intuitively, in a purely functional component a sequence of normal and unobservable fault transitions may occur. Then, the component executes one fault or recovery transition (normally in order to synchronize with a corrector) and reach normal behavior. If no fault occurs, a purely functional component continues executing normal transitions.

Definition 8 *Let $B = (Q, P, \rightarrow, q^0)$ be an atomic component. We say that B is a* pure corrector *for the set $\rightarrow_n$ of normal transitions, iff*

1) *B is a corrector for $\rightarrow_n$.*
2) (Strong Stability) *For any trace $\pi = p_0 p_1 \cdots$ of component B, for all $i \geq 0$, if $q_i \xrightarrow{p_i} q_{i+1}$ is in $\rightarrow_n$, then $q_{i+1} \xrightarrow{p_{i+1}} q_{i+2}$ is not in $\rightarrow_{n,r}$.*

When a normal transition is executed in a pure corrector, it does not execute any more normal transitions. This intuitively means that this normal transition marks the completion of recovery and the pure corrector stops working unless another fault occurs. Thus, we require that this normal transition synchronizes with some normal or recovery transition (normally a purely functional component) in the composite component.

We now show that in the absence of faults, a pure corrector plays no role in the behavior of a model that contains it. In other words, in the absence of faults, the existence of a pure corrector in a model can be overlooked.

Theorem 2 *Let $B = \gamma(B_0 \cdots B_m)$ be a composite component and B_i, $0 \leq i \leq m$, be the one and only pure corrector in B. The set of traces of $\gamma_N(B_0 \cdots B_m)$ and $\gamma(B_{0_n} \cdots B_{i-1_n}, B_{i+1_n} \cdots B_{m_n})$ are equal.*

A trivial but important consequence of Theorem 2 is that pure correctors do not *interfere* with pure functional components.

Corollary 1 *Let $B = \gamma(B_0 \cdots B_m)$ be a composite component and $B_i = (Q_i, P_i, \rightarrow_i, q_i^0)$, $0 \leq i \leq m$, be the one and only pure corrector in B. Let $\pi = a_0 a_1 \cdots$ be a trace of B. If for all $j \geq 0$, $a_j \in \gamma_N$, then no interaction in π involves a port in P_i.*

The other side of the coin is that when a fault occurs in a purely functional faulty component, it stops working until recovery from the fault is complete.

Theorem 3 *Let $B = \gamma(B_0 \cdots B_m)$ be a composite component and B_i, $0 \leq i \leq m$, be the one and only purely functional atomic component in B. The set of traces of $\gamma_R(B_0 \cdots B_m)$ and $\gamma(B_{0_r} \cdots B_{i-1_r}, B_{i+1_r} \cdots B_{m_r})$ are equal.*

An immediate application of Corollary 1 and Theorem 3 is in compositional analysis of fault-tolerant systems. For instance, in order to verify the correctness of functional (respectively, recovery) properties of a non-masking composite component, one can simply remove pure correctors (respectively, functional components) from the model and verify the remaining composite component with respect to functional (respectively, recovery) properties. Such decomposition clearly assists in reducing the size of state space in the context of model checking. In the context of monolithic programs represented in terms of guarded commands in the shared memory model, identifying correctors has shown to be effective in significantly reducing the cost of model checking [18].

However, dealing with decomposition of monolithic models is not as straightforward as the same task in our component-based model in this paper.

VI. Conclusion

In this paper, we proposed a generic formal framework for specifying and reasoning about fault recovery (also called *non-masking* fault-tolerance) for component-based models. We characterized component-based models based on the BIP (Behavior, Interaction, Priority) framework [3], [4]. However, our method is not limited to BIP. In BIP, the behavior of an atomic component is specified by a labeled transition system. A BIP model (i.e., a composite component) is obtained from a set of atomic components glued by interactions.

Unlike the approaches in [16], [17], [19]–[22] where a monolithic model is analyzed or components are defined in terms of properties of sets of computations, our method is based on observational behavior of a model in the presence of faults. Also, we use explicit components, each having its own private state space and behavior. We defined what it means for a component to be a *corrector* and showed that non-masking models must contain corrector components. These components correct the observational behavior of a faulty model and we illustrated they can be constructed as stand-alone components interacting with components that provide functional tasks. We described the application of this result in compositional model checking.

We plan to incorporate the results in this paper in our work on automated derivation of distributed implementation from BIP models [6], [7], where fault-tolerance plays an important role. An interesting future research direction is developing methods for incremental construction of non-masking models, and for transforming an arbitrary non-masking model into a well-structured model, where all atomic components are pure non-masking components so as to achieve separation of concerns. Further open problems include compositional synthesis and verification of recovery paths in non-masking models.

References

[1] F. Cappello, A. Geist, B. Gropp, L. Kale, B. Kramer, and M. Snir, "Toward exascale resilience," *Journal of High Performance Computing Applications*, vol. 23, pp. 374–388, 2009.

[2] A. Arora and M. G. Gouda, "Closure and convergence: A foundation of fault-tolerant computing," *IEEE Transactions on Software Engineering*, vol. 19, no. 11, pp. 1015–1027, 1993.

[3] G. Gössler and J. Sifakis, "Composition for component-based modeling," *Sci. Comput. Program.*, vol. 55, no. 1-3, pp. 161–183, 2005.

[4] A. Basu, M. Bozga, and J. Sifakis, "Modeling heterogeneous real-time components in BIP," in *Software Engineering and Formal Methods (SEFM)*, 2006, pp. 3–12.

[5] T. Abdellatif, J. Combaz, and J. Sifakis, "Model-based implementation of real-time applications," in *EMSOFT*, 2010, pp. 229–238.

[6] B. Bonakdarpour, M. Bozga, M. Jaber, J. Quilbeuf, and J. Sifakis, "From high-level component-based models to distributed implementations," in *EMSOFT*, 2010, pp. 209–218.

[7] ——, "Automated conflict-free distributed implementation of component-based models," in *IEEE Symposium on Industrial Embedded Systems (SIES)*, 2010, pp. 108–117.

[8] M. Bozga, V. Sfyrla, and J. Sifakis, "Modeling synchronous systems in BIP," in *EMSOFT*, 2009, pp. 77–86.

[9] A. Basu, B. Bonakdarpour, M. Bozga, and J. Sifakis, "Systematic correct construction of self-stabilizing systems: A case study," in *International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS)*, 2010, pp. 4–18.

[10] A. Arora and M. Gouda, "Distributed reset," *IEEE Transactions on Computers*, vol. 43, pp. 316–331, 1994.

[11] B. Bonakdarpour, S. S. Kulkarni, and F. Abujarad, "Symbolic synthesis of masking fault-tolerant programs," *Distributed Computing*, to appear.

[12] B. Bonakdarpour and S. S. Kulkarni, "Incremental synthesis of fault-tolerant real-time programs," in *International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS)*, ser. LNCS 4280, 2006, pp. 122–136.

[13] ——, "Masking faults while providing bounded-time phased recovery," in *International Symposium on Formal Methods (FM)*, 2008, pp. 374–389.

[14] J. Chen and A. S. Kulkarni, "Effectiveness of transition systems to model faults," in *Logical Aspects of Fault-Tolerance (LAFT)*, 2011, to appear.

[15] E. W. Dijkstra, "Self-stabilizing systems in spite of distributed control," *Communications of the ACM*, vol. 17, no. 11, 1974.

[16] A. Arora and S. S. Kulkarni, "Detectors and correctors: A theory of fault-tolerance components," in *International Conference on Distributed Computing Systems (ICDCS)*, 1998, pp. 436–443.

[17] B. Bonakdarpour, S. S. Kulkarni, and A. Arora, "Disassembling real-time fault-tolerant programs," in *EMSOFT*, 2008, pp. 169–178.

[18] B. Bonakdarpour and S. S. Kulkarni, "Compositional verification of real-time fault-tolerant programs," in *EMSOFT*, 2009, pp. 29–38.

[19] L. Lamport, "The temporal logic of actions," *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 16, pp. 872–923, 1994.

[20] Z. Liu and M. Joseph, "Specification and verification of fault-tolerance, timing, and scheduling," *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 21, no. 1, pp. 46–89, 1999.

[21] ——, "Specification and verification of recovery in asynchronous communicating systems," in *Formal techniques in real-time and fault-tolerant systems (FTRTFT)*, 1993, pp. 137–163.

[22] ——, "Transformation of programs for fault-tolerance," *Formal Aspects of Computing*, vol. 4, no. 5, pp. 442–469, 1992.