Reliability Engineering and System Safety 142 (2015) 184–191

Contents lists available at ScienceDirect

Reliability Engineering and System Safety

journal homepage: www.elsevier.com/locate/ress

Towards a sound semantics for dynamic fault trees

Antoine Rauzy *, Chaire Blériot-Fabre

Laboratoire de Génie Industriel, Ecole Centrale Paris, Grande voie des vignes, Châtenay-Malabry 92295, France

ARTICLE INFO

Article history:
Received 1 November 2012
Received in revised form
30 March 2015
Accepted 26 April 2015
Available online 16 May 2015

Keywords:
Dynamic fault trees
Guarded transitions systems

ABSTRACT

In this article, we study the semantics of dynamic fault trees and related formalisms. We suggest that there are actually three mechanisms at work in dynamic fault trees: first, changes of states due to occurrences of events, second bottom-up propagations of values as in static fault trees, and third top-down propagations of demands of activations of components. We propose a direct translation of dynamic fault trees into guarded transitions systems, the underlying mathematical model of the AltaRica 3.0 modeling language. This encoding provides a good basis for our study. We discuss also assessment algorithms at hand in light of this translation.

1. Introduction

In the recent past years, dynamic fault trees and related formalisms have focused a large attention in the reliability engineering literature (see e.g. [7,13,3,11,4,5,8,15,17]). By adding some extra-logical constructs to regular/static fault trees, one aims to describe dynamic behaviors, i.e. to put constraints on order of occurrences of events, while maintaining the conceptual (and graphical) simplicity of fault trees. This increase in expressive power comes indeed with a price: models cannot be interpreted anymore in the Boolean algebra framework.

In this article, we claim that there are actually three mechanisms at work in dynamic fault trees and Boolean Driven Markov Processes: first, changes of states of due to occurrences of events, second bottom-up propagations of values as in static fault trees, and third top-down propagations of demands of activations of components. To define a sound semantics of dynamic fault trees, we encode them into guarded transitions systems. Guarded transitions systems have been introduced in reference [19]. They are at the core of the new version of the high level modeling language AltaRica 3.0 (see e.g. [20]). Guarded transitions systems are (finite or infinite) state automata with input and outputs. They generalize most of the formalisms used for probabilistic safety analyses, including static fault trees, reliability block diagrams and generalized stochastic Petri nets. The definition of a semantic for dynamic fault trees involves actually multi-state components, immediate and timed (stochastic) transitions as in generalized stochastic Petri nets [1] (these concepts are not available in block diagrams or static fault trees), as well as block-wise construction and remote value propagation as in reliability block diagrams (these concepts are not available in generalized stochastic Petri nets).

The encoding we propose here is based on some preprocessing and a one-to-one correspondence between dynamic fault tree constructs and their counterparts in terms of guarded transitions systems. In other words, we design a library of reusable modeling components, one per dynamic fault tree construct. The design of a dynamic fault tree model consists then simply in assembling these predefined components. Proceeding this way presents at least three important advantages compared to specific approaches. First, it clarifies the semantics of each and every construct. Second, it makes it easy to extend the library with new constructs. Third, all assessment tools designed for guarded transition systems are instantly applicable to dynamic fault trees. Regarding this last point, the key question is to determine whether assessment algorithms can take advantage of the specificity of dynamic fault tree constructs. We give arguments to show that this question should be studied in light of the models chosen for basic components and that the answer is probably negative.

The original contribution of this article is twofold. First, we show that guarded transitions systems provide a suitable framework to clarify the semantics of dynamic fault trees. Second, we relate this semantic and assessment algorithms at hand with models chosen for basic events.

The remainder of the article is organized as follows. Guarded transitions systems are introduced Section 2. The translation of dynamic fault tree constructs into guarded transition systems is studied Section 3. Finally, algebraic interpretations and assessments algorithms are discussed Section 4.

2. Guarded transitions systems

2.1. Definition

A Guarded Transitions System is a quadruple, $\langle V = S \uplus F, E, T, A \rangle$ where,

* Corresponding author.
E-mail address: Antoine.Rauzy@centralesupelec.fr (A. Rauzy).

http://dx.doi.org/10.1016/j.ress.2015.04.017
0951-8320/

- S and F are two disjoint (finite) sets of variables. S is the set of state variables, F the set of flow variables. Variables have a type (Boolean, Integer, an Enumeration of symbolic constants…) and a default/initial value.
- E is a (finite) set of events. Events are either immediate or stochastic.
- T is a set of transitions. Transitions are triple ⟨G, e, P⟩, where G is Boolean condition built over state and flow variables, e is an event, and P is an assignment of a state variables, i.e. a set of individual assignments of the form s:=K, where s is a state variable and K is an expression built over variables. G and P are called respectively the guard and the action of the transition. For the sake of clarity, the transition ⟨G, e, P⟩ is denoted e: G→P.
- A is an assertion, i.e. a set of assignments in the form f:=L, where f is a flow variable and L is an expression built over variables. A flow variable is assumed to appear only once as the left member of an equation.

A transition is fireable when its guard is satisfied by the current values of variables. The firing of a transition is a two steps process: first, the action is performed, i.e. the values of (some) state variables are changed; second, the values of flow variables are updated by means of the assertion. Immediate transitions take no time while timed (or stochastic) transitions are assumed to take some (possibly infinitely small) amount of time. The underlying model of time is similar to the one of generalized stochastic Petri nets [1] if we assume that delay of stochastic transitions are Markovian (i.e. obey negative exponential distributions).

The update of flow variables after each transition firing is performed thanks to a fixpoint mechanism [19], i.e. values of left members of equations are recalculated until the system stabilizes. This stabilization is obtained in at most two passes, i.e. it is linear in the size of the assertion in the worst case. It is atomic, i.e. it is assumed to take no time (as immediate transitions). The important point here is that this fixpoint mechanism makes it possible to model remote interactions between components.

To illustrate the above definitions, let us consider first a simple non-repairable component. The guarded transition system for this component is pictured Fig. 1. It is made of the following elements:

- A state variable s, which takes its value into the enumeration {working, failed}. The initial value of s (working) is indicated with a small entering arrow.
- A Boolean flow variable: out.
- A stochastic event: failure.
- A transition transitions:
- failure: s=working→s:=failed
- An assertion made of an unique assignment:
- out:= (s=failed)

Now consider a spare, non-repairable component in cold redundancy. The guarded transition system for that component is pictured Fig. 2. It is made of the following elements:

- Two state variables a and s, which take respectively their values into enumerations {standby, active} and {working, failed}.
- Two Boolean flow variables: demand and out.
- Four events: start, failureOnDemand, failure and dormantFailure. start and failureOnDemand are immediate (pictured as dashed arrows). Failure and dormantFailure are stochastic transitions (pictured as plain arrows).
- Four transitions:
- start: a=standby and demand→a:= active
- failureOnDemand: a=standby and s=working and demand→a:= active, s:= failed
- failure: a=active and s=working→s:= failed
- dormantFailure: a=standby and s=working→s:= failed
- An assertion made of a unique assignment ("demand" is an input flow variable):
- out:=(s=failed)

This example is helpful to introduce a point we did not discussed so far. Three transitions leave the state "a=standby and s=working". Two of them are immediate ("start" and "failureOnDemand") and one is stochastic ("dormantFailure"). When the input flow "demand" gets true, these three transitions are in conflict. However, in the Markovian framework adopted in this article, the probability that the delay associated with the stochastic transition is null is zero. Therefore, immediate transitions have the priority. Still they are in conflict. The choice between "start" and "failureOnDemand" is non-deterministic. It is possible however to influence the probabilities with which transitions in conflict are fired by associating a weight (called expectation in AltaRica 3.0 jargon) with each of them. The probability that a particular transition is fired is then the weight of this transition divided by the sum of the weights of the transitions in conflict. By default, the weight of a transition is 1.

2.2. Composition

Guarded transition systems can be enclosed into blocks (as illustrated Figs. 1 and 2). Then blocks can be composed to create larger blocks, as illustrated Fig. 3 where the model for the simple component described Fig. 1 and the model for the spare component described Fig. 2 are composed with a block G representing an AND gate. The idea behind the encoding of dynamic fault trees into guarded transition systems is to design a library of generic blocks representing each type of basic events and gates and then to assemble instances of these blocks just as exemplified Fig. 3. This encoding provides a sound semantics for dynamic fault trees because the semantics of guarded transition systems is itself completely and formally defined [19]. Guarded transitions systems are actually richer than what we presented here. However, this presentation suffices for the purpose of the present article.

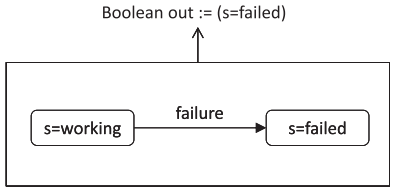

Fig. 1. The guarded transition system for a non-repairable component.

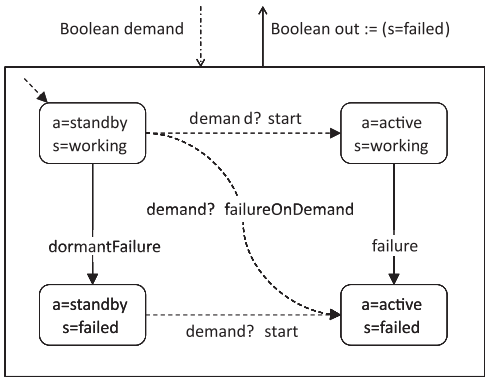

Fig. 2. The guarded transitions system for a spare non-repairable component.