Comparison between Bisimulation and Forced simulation

1 Introduction

Bisimulation theories were historically introduced by the computer science community for analyzing and reducing topological complexity of concurrent processes [1]. However, to many cases, a generic implementation cannot match a given specification. In this report, to deal with this problem, another simulation method called Forced simulation to perform automatic component adaptation to match the component to a specific design function [2] will be introduced and made comparison with Bisimulation. In the following sections, the Bisimulation and Forced simulation will be introduced and used to check one example. Then, another example made by the author will be presented. Finally, conclusion will be made in the last section.

2 Overview

To many cases, we can "force" processes which are not bisimliar to be "bisimliar". Given LTSs F of a design function (a specification) F and D of a device D, we wish to define an Forced simulation relation F $\sqsubseteq_{\Sigma}$ D (we use $\sqsubseteq_{\Sigma}$ to distinguish Forced simulation from other types of simulations), when F can be implemented by D using the notion of forcing. If an Forced simulation relation exists between F and D, then we say that D can implement F. [3]

In this section, some basic knowledge and theory will be introduced. And an example will be presented for our description of the two different simulation methods.

Definition 1[3]:

A labelled transition system (LTS) is defined to be a tuple $\{S, L, \xrightarrow{l} : l \in L\}$ where:

1. S is a set of states,

2. $\{s0 \in S\}$ is a unique start state.

3. L is a set of transition labels, where $L = eL \cup lL$, where eL denotes a set of external labels and lL denotes a set of local (internal) labels,

4. $\xrightarrow{l} \subseteq S \times S$ for each $l \in L$.

We shall use symbols a, b, c.. to range over external labels, Also, states will be denoted by s1; s2. We add the subscripts f and d to any of these symbols, if necessary, to make the context clear (of device D or function F).

Example 1.

As shown in the two diagrams, we consider two processes including a multi-functional component which can behave as a down counter, a rate generator or a square wave generator. [2,3]

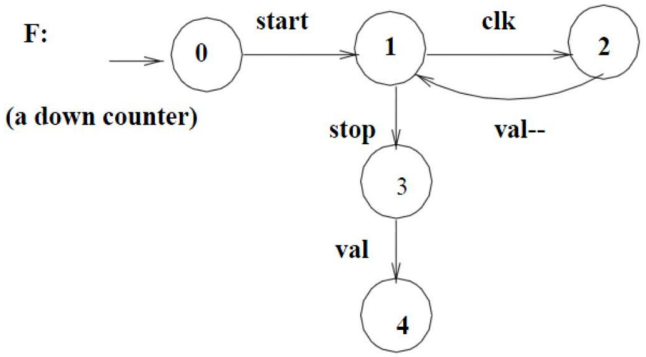


Fig. 1.

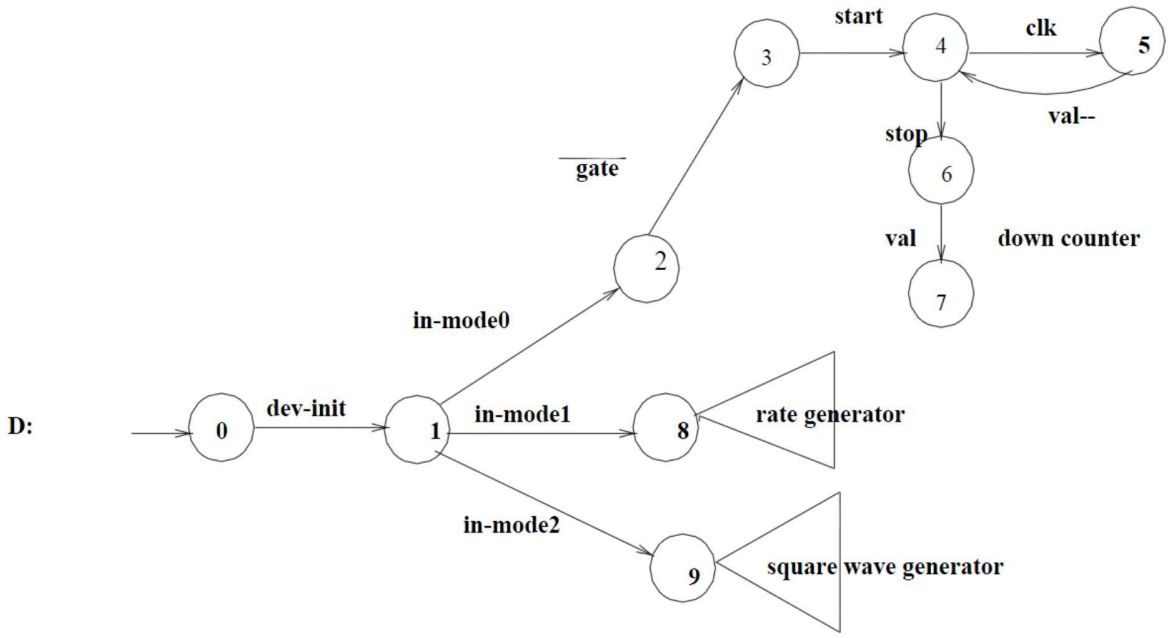


Fig. 2.

In next section, Bisimulation will be introduced and used to check this example.

3 Bisimulation

Bisimulation equivalence is a semantic equivalence relation on labelled transition systems, which are used to represent distributed systems. It identifies systems with the same branching structure.[4]

3.1 Strong Bisimulation.

Definition 2 [5]:

Let $< Q, A, \rightarrow >$ be an LTS. A relation $R \subseteq Q \times Q$ is a Bisimulation if, whenever $(p, q) \in R$ then the following conditions hold for any a, p', q' .

1 if $p \xrightarrow{a} p'$ then $q \xrightarrow{a} q'$ for some q' and $(p', q') \in R$.

2 if $q \xrightarrow{a} q'$ then $p \xrightarrow{a} p'$ for some p' and $(p', q') \in R$.

Processes P and Q are Bisimulation equivalent, or bisimilar, if there exists a Bisimulation relation R such that (P,Q)$\in$R, we treat P,Q are Bisimilar.

3.2 Weak Bisimulation.

Strong Bisimulation is often too discriminating an equivalence on transitions systems. For this reason weak Bisimulation which abstracts away from invisible actions, usually labelled τ is designed.[6]

Definition 3:

Let $< Q, A, \rightarrow >>$ be an LTS with $\tau \in$ A, and let q $\in$ Q.

1 If s $\in$ A* then $\hat{S} \in$ (A – {τ})* is the action sequence obtained by deleting all sequences of τ from s.

2 Let s $\in$ (A – {τ})*. Then $q \stackrel{s}{\Rightarrow} q'$ if there exists s0 such that $q \stackrel{s'}{\rightarrow} q'$ and s = ^s0.

Definition 4:

Let $< Q, A, \rightarrow >>$ be an LTS with $\tau \in$A. Then a relation $R \subseteq Q \times Q$ is a weak Bisimulation if, whenever (p, q) $\in$R, then the following hold for all a $\in$A and (p', q')$\in$ Q.

1 If $p \xrightarrow{a} p'$ then $q \stackrel{\hat{a}}{\Rightarrow} q'$ for some q' such that $(p', q') \in R$.

2 If $q \xrightarrow{a} q'$ then $p \stackrel{\hat{a}}{\Rightarrow} p'$ for some p' such that $(p', q') \in R$.

States p and q are observationally equivalent, or weakly bisimilar, if there exists a Bisimulation relation R containing (p, q). When this is the case, we write p $\approx$ q. [5]

Obviousely, if we consider the initial states of the 2 processes, and state D1 as examples, since dev-init and in-mode0 in D

do not exist in F, they do not meet the definition of the either Strong or Weak Bisimulation. However, if we can build an interface to formalize the process, they may be "forced" simulation. The Forced simulation will be introduced in next section.

4 Forced simulation

Forced simulation is a Bisimulation variant and hence standard partition-refinement based Bisimulation algorithms may be modified for computing forced simulation. [7] It is based on various notions-like devices, functions and interfaces and their representations which are formalized. Then, an interface process and show how to compose an interface with a device should be defined. Finally, Forced simulation can be implemented. This solves the component matching problem. [8]

Definition 5-- Forced simulation:[2].

In this definition, we assume:

1. LTSs may be non-deterministic.
2. Both F and D are capable of performing internal actions. Each has a set of internal actions and each internal action represents a different internal operation.

Then, the theory is formalized like below:

$$s_f \sqsubseteq_\Sigma s_d \stackrel{def}{=} \exists k \geq 0 : s_f \sqsubseteq_\Sigma^k s_d$$

$$s_f \sqsubseteq_\Sigma^0 s_d \stackrel{def}{=} subtree\_isomorphic(s_f, s_d) \wedge internally\_terminal(s_d, s_f)$$

$$s_f \sqsubseteq_\Sigma^{k+1} s_d \stackrel{def}{=} (s_f \sqsubseteq_\Sigma^k s_d) \vee ((subtree\_isomorphic(s_f, s_d) \wedge$$

$$(\forall i \in lL_d, \forall s'_d : s_d \rightarrow_i s'_d \wedge s_f \not\rightarrow_i s'_f : s_f \sqsubseteq_\Sigma^k s'_d)) \vee$$

$$((\exists a \in eL_d, \forall s'_d : s_d \rightarrow_a s'_d : s_f \sqsubseteq_\Sigma^k s'_d) \wedge (\forall i \in lL_d, \forall s'_d : s_d \rightarrow_i s'_d : s_f \sqsubseteq_\Sigma^k s'_d))$$

$$subtree\_isomorphic(s_f, s_d) \stackrel{def}{=} \forall l, s'_f :$$
$$[\ (s_f \rightarrow_l s'_f \Rightarrow \exists s'_d : s_d \rightarrow_l s'_d \wedge s'_f \sqsubseteq_\Sigma s'_d) \wedge (\forall s''_d : s_d \rightarrow_l s''_d : s'_f \sqsubseteq_\Sigma s''_d)]$$

$$internally\_terminal(s_d, s_f) \stackrel{def}{=} \text{the only internal transitions allowed out of } s_d \text{ are common internal transitions of } s_f$$

As we can see, to the example in Fig 1 and Fig 2, Since *dev-init* and *in-mode0* in D are external signals, the interface can generate them to place the device in state 2. A difference between F and the down counter in D is that the latter requires an extra control signal called gate to be set to low. This is an example where F is under-specified. Hence, for F to be simulated by D, the interface must force all transitions of D from state 0 through to state 3. Once the interface has placed the device in state 3, simulation of F is guaranteed. [2]

interface:

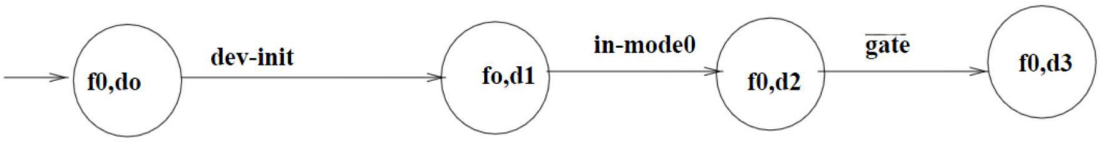


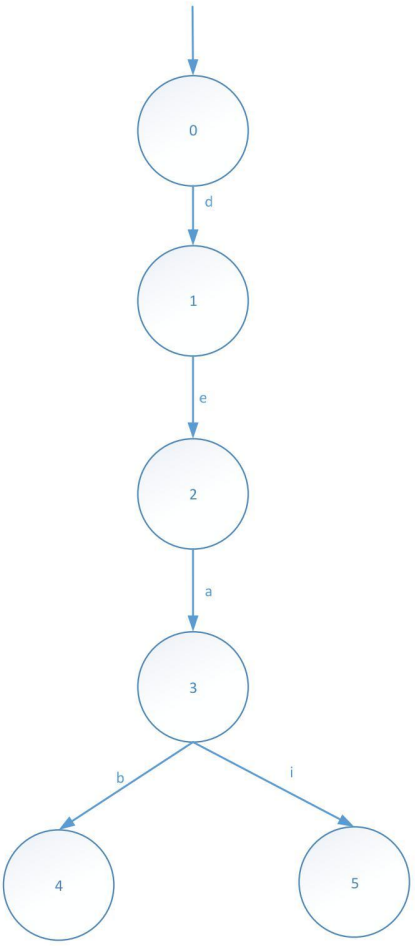
Fig. 3.

5 Another Example:

After making the literature review, I make one example:

There are two processes F and D in the figures below. Obviously, they do not satisfy Bisimulation. If we need F to be simulated by D, we need interface shown in the following figure. Because the actions d, e are external states the interface can generate them to place the device in state 2. Another difference between F and the D is that F needs action c to state 3 after state 1 following by action 'i' which is directly used in D. So we should force it by "true/{c}".

D:



F:

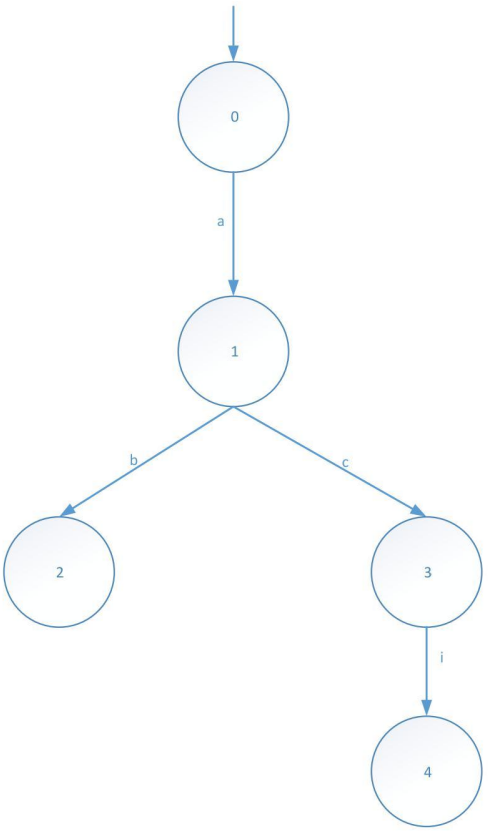


Fig. 4.

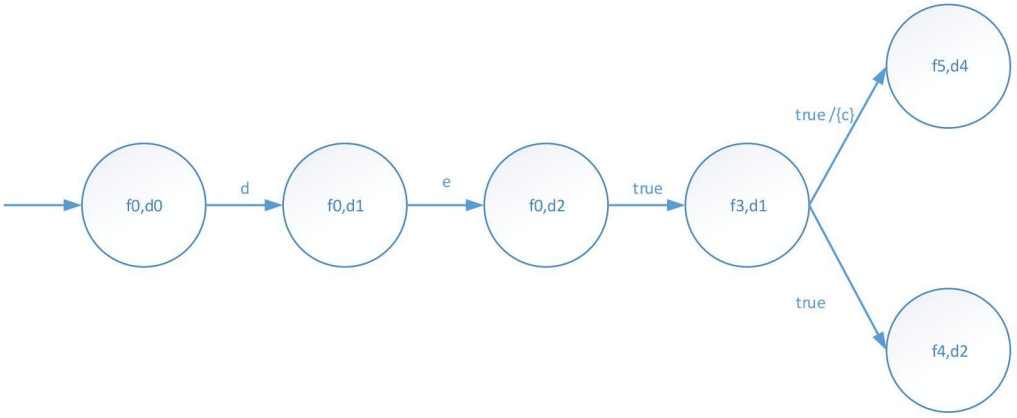


Fig. 5.

6 Conclution

In this report, the introduction and comparison between Bisimulation and Forced simulation is exemplified. Forced simulation is a Bisimulation variant which can "force" processes simulation by making interfaces to make better reuse performance. Some associate applications are introduced, too. Forced simulation is a very important development over existing simulation techniques, considering that many system level devices are multi-functional and hence the interface is required to guide the device along its appropriate functionality to match the specification [3]. There is still more work to do in the future, such as doing deeper research on reality use of Forced Simulation.

7 Reference

1. Pola G, Van Der Schaft A J, Di Benedetto M D. Bisimulation theory for switching linear systems[J]. 2004.
2. Roop P S, Sowmya A, Ramesh S. Automated component adaptation by forced simulation[C]//Computer Architecture Conference, 2000. ACAC 2000. 5th Australasian. IEEE, 2000: 74-81.
3. Roop P S, Sowmya A, Ramesh S. Automatic component matching using forced simulation[C]//VLSI Design, 2000. Thirteenth International Conference on. IEEE, 2000: 64-69.
4. Van Glabbeek R J, Goltz U. Equivalences and refinement[M]. Springer Berlin Heidelberg, 1990.
5. Partha S Roop "Concurrency and process algebras", The University of Auckland ,COMPSYS 705 ,a2015
6. Fiore M, Cattani G L, Winskel G. Weak Bisimulation and open maps[C]//Logic in Computer Science, 1999. Proceedings. 14th Symposium on. IEEE, 1999: 67-76ebras
7. Roop P S, Sowmya A, Ramesh S, et al. Tabled logic programming based IP matching tool using forced simulation[J]. IEE Proceedings-Computers and Digital Techniques, 2004, 151(3): 199-208.
8. Roop P S, Sowmya A, Ramesh S. Forced simulation: A technique for automating component reuse in embedded systems[J]. ACM Transactions on Design Automation of Electronic Systems (TODAES), 2001, 6(4): 602-628.