Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich
Networked Systems Group (NSG)

HS 2021

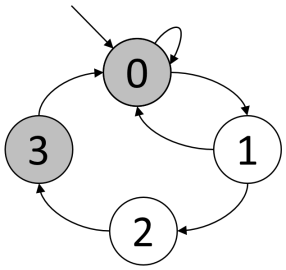
R. Jacob / T. Schneider
based on Prof. L. Thiele's material

Discrete Event Systems

Exercise Sheet 11

1 Temporal Logic

a) We consider the following automaton. Property a holds in the colored states (0 and 3).



For each of the following CTL formula, list all the states for which it holds true.

(i) EF a

(ii) EG a

(iii) EX AX a

(iv) EF (a `AND` EX `NOT`(a))

b) You are given a set of states S, the transition function $f : S \mapsto S$, encoded as the characteristic function $\psi_f(q, q')$ (which returns *true* only if $f(q) = q'$), and the set $Z \subseteq S$ with its characteristic function $\phi_Z(q)$ (which returns *true* only if $q \in Z$). Your goal is to come up with a simple algorithm to find the characteristic function $\psi_{\mathrm{AF}\,Z}(q)$, which encodes the set of all states satisfying AF Z.

(i) Give a relation between EG Z and AF Z.

(ii) Using this relation, formulate an iterative procedure to find the set of states that satisfy AF Z. Use regular set operations to find the procedure. You can use the predecessor function $Pre(Q, f)$, which returns the set of states from which we can reach states in Q using the transition function f in one step.

$$Pre(Q, f) = \{q' : \exists q,\, \psi_f(q', q) \cdot \psi_Q(q) = 1\}$$

(iii) Translate the iteration procedure from (ii) into an algorithm using boolean expressions. Assume that you are given, for each set Q (i.e., its characteristic function), the characteristic function $\psi_{Pre(Q,f)}$.

2 Safe Network-Wide Configuration Updates

In this exercise, you are a network engineer operating an extremely important computer network with many policies that cannot be violated. A client has noticed a weird behavior with some internet packets being dropped along the way. You managed to figure out that this behavior is due to sub-optimal configuration of some of the network devices, and you have prepared an updated configuration that fixes the problem. Your goal is to update the network (also known as performing a network migration), while verifying that the policies are always satisfied. All policies should hold not only for the initial and final configuration, but also in every intermediate state during the migration. In this exercise, you will see how to use either BDDs or CTL and model checkers to find migrations that are formally guaranteed to always satisfy the policies.

Computer networks are distributed systems that provide communication between different endpoints. Let us consider a very simple view of computer networks (see Figure 1). We are given an undirected graph $G = (V, E)$ with nodes V (network routers or sinks) and edges E (links between routers).

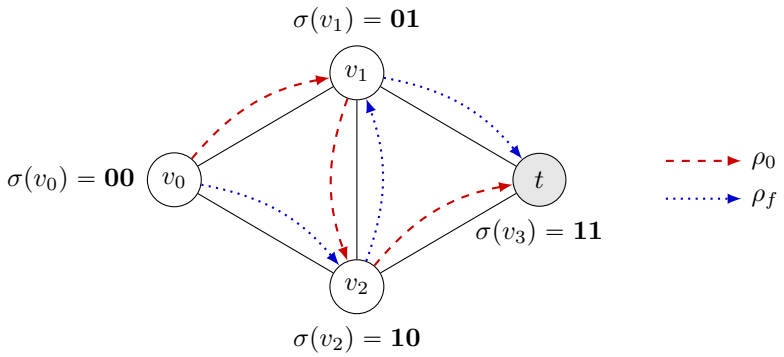


Figure 1: Simple network with 3 routers (v_0, v_1, and v_2) and one sink (t). The initial routing function ρ_0 is drawn as red dashed lines, and the final routing function ρ_f as blue dotted lines.

Any router in the network forwards packets either towards one of its neighbors, or drop the packets. We encode the initial routing state with the routing function ρ_0, as shown in Figure 1 as red dashed arrows. The operators intend to change it into the routing function ρ_f, drawn as blue, dotted arrows. Both routing functions are summarized in the following table.

router v	initial next-hop $\rho_0(r)$	final next-hop $\rho_f(r)$
v_0	$\rho_0(v_0) = v_1$	$\rho_f(v_0) = v_2$
v_1	$\rho_0(v_1) = v_2$	$\rho_f(v_1) = t$
v_2	$\rho_0(v_2) = t$	$\rho_f(v_2) = v_1$

Due to the limitations of the network being a distributed system, it is practically impossible to perform the update on all devices at the same time. Hence, the update needs to be performed for one device at a time. Your task is it to model the update, and to find an update sequence, that preserves important properties (more on that later).

To start, we encode the state of the network—the routing function—by encoding, for each router v_0, v_1, and v_2 how they treat incoming packets. We write the state as $\mathbf{Z} = [\mathbf{z}_0, \mathbf{z}_1, \mathbf{z}_2]$, where $\mathbf{z}_i$ describes the forwarding rule of r_i. For each router r_i, we use two bits $\mathbf{z}_i = [z_i^1, z_i^0]$, and its numeric value represents the next-hop the router r_i forwards traffic to (as indicated in Figure 1). If any node v_i selects itself as a next hop, this means that the packets are trapped in an infinite loop. For instance, we encode with $\mathbf{z_0} = 01$ that node v_0 forwards packets towards v_1. As an example, the initial forwarding state $\mathbf{Z}_0$ is be represented as:

variable	value
$\mathbf{z}_0$	01
$\mathbf{z}_1$	10
$\mathbf{z}_2$	11

The initial forwarding state $\mathbf{Z}_0$ can be expressed by the following encoding $\sigma(\rho_0)$ and boolean expression $\psi_{\rho_0}(\mathbf{Z})$ (which is only *true* if $\mathbf{Z}$ is the initial state Z_0):

$$\sigma(\rho_0) = 01\,10\,11 \qquad \psi_{\rho_0}(\mathbf{Z}) = \bar{z_0^1} z_0^0\, z_1^1 \bar{z_1^0}\, z_2^1 z_2^0$$

a) List all forwarding states where packets are trapped in an infinite loop.

b) The forwarding state is restricted by the physical topology. Routers can only choose a next-hop if they really are adjacent to that next-hop. Express the characteristic function $\psi_{topo}(\mathbf{Z})$, which should only return *true* if the state is actually valid.

c) Router v_0 is expected to route important traffic towards t. Find a characteristic function $\psi_t(\mathbf{Z})$ that returns true only for a network state in which v_0 reaches sink t. Remember, that in any specific network state $\mathbf{Z}$, a router always uses the same next-hop.

Hint: Start at v_0, and enumerate (by case distinction) all possible cases of how t can be reached.

d) In addition to this property, the network operator also requires traffic from v_0 to always traverse v_2. Similarly to before, find the characteristic function $\psi_{v_2}(\mathbf{Z})$ that only returns true for a network state where v_0 eventually reaches v_2.

e) Find the characteristic function $\psi_\phi(\mathbf{Z})$ which only returns *true* if, for a given network state $\mathbf{Z}$, it eventually reaches or traverses both t and v_2.

f) Verify that ψ_ϕ holds for both the initial and the final state.

g) As of now, we only have considered a single state of the network. However, we wish to transition from the initial $\mathbf{Z}_0$ to the final state $\mathbf{Z}_f$. Recall that we can only change the routing decision of a single router in the network at the same time. Express the transition function $\psi_{trans}(\mathbf{Z}, \mathbf{Z}')$, which should only return *true* if we can transition from state $\mathbf{Z}$ to state $\mathbf{Z}'$ by changing the forwarding of a single router. You are allowed to use quantifiers ($\forall$ and $\exists$), but you do not have to. You do not have to simplify the expression.

So far, we have described a state machine with $2^6 = 64$ different states. However some of them are not allowed by the topology (as derived in Task **b**). In addition, not all of them satisfy the policies (as derived in Task **e**). Finally, many state transitions are also not allowed (as derived in Task **g**). Your task now is to combine all these constraints, and to find a valid and safe migration, i.e., we eventually reach $\mathbf{Z}_f$ (valid) while all policies are never violated (save). In the final two tasks, we discuss two different methods to reach this goal. The first method (Task **h**) is to use a model checker in order to find a single migration that is valid and safe. The second method (Task **i**) is to use a computer to build an ROBDD that represents all valid and safe migrations at once.

h) How can we use that model checker to find a valid and safe sequence of states in order to perform the network migration? Don't actually use one, you just need to describe *how* you would encode the problem and use a model checker. You can assume that the model checker can check a CTL expression on a given state machine, or give you a counter-example if the expression is not satisfied. Please explain how you build the state machine (which states and transitions it should contain), and give the precise CTL formula you would ask the model checker to verify in order to find a sequence of states to perform the migration. You may use all results from the previous tasks.

i) We now wish to find an ROBDD that encodes all valid migrations. Assume that we are given the information that it is possible to transition from the initial to the final state in just three steps. Write down the boolean expression based on the symbolic variables $\mathbf{Z}_1$, $\mathbf{Z}_2$ and $\mathbf{Z}_3$, which returns *true* only for all sequences of states that transition safely from $\mathbf{Z}_0$ to $\mathbf{Z}_f$. You do not need to simplify the expression in any way, nor actually draw the ROBDD. Usually, we would use a computer to simplify the expression, and build the ROBDD automatically.

The Bigger Picture In this exercise, we have built a model which we can use to find a solution to a real-world problem. In fact, (almost) all steps above can be done automatically by a computer. The only step, that is yet a bit unclear is how to find the constraints ψ_ϕ. However, there are other approaches that show how this can be built without manual reasoning. (We have followed a different, manual approach, because the automatic method is hard to motivate, and difficult to understand in just two hours.) In fact, current research is discussing a very similar approach to performing graceful network state migration, highlighting the relevance of ROBDD, CTL and model checking (e.g., `https://snowcap.ethz.ch`).