Scheduling of Washing Cycles

Romain Brenguier
Université Libre de Bruxelles – Brussels, Belgium
romain.brenguier@ulb.ac.be

***Abstract*—We present a case study of the specification of a washing systems, with cycles which can be launched in parallel and how we generated AIGER benchmarks for this problem.**

I. Introduction

The original AIGER format [1] is a safety property specification used for model checking. Its extended version was introduced in 2014 by Jacobs [2] to represent safety objectives for controller synthesis. It was first used in the Synthesis Competition 2014, a satellite event of the 26th International Conference on Computer Aided Verification 2014.

We consider a case study, where we want to schedule cycles for a washing system for vegetables or fruits. The system is composed of several tanks in parallel that can be controlled by the user pushing buttons. We specify the correct behavior of the system with rational expressions which gives safety constraints. These safety constraints have then be translated to non-deterministic automata and writen in the AIGER format.

II. The problem

We consider the design of a controller for a washing system. The user can press a button which should initiate the arrival of water in a tank where the produce has been placed. After a determined time, the controller should open the valve to remove the water from inside the tank. We consider several such tanks put in parallel with a centralized controller, which has one button per tank, and a light that should be on whenever one of the tanks contain water.

For each tank i, we have an (uncontrollable) input push_i which is on when the corresponding button is pressed by the user, a controllable input fill_i which release water into the tank when on, and a controllable input empty_i which removes water from the tank. This is illustrated in Fig. 1. We also have a controllable input light which is on when the light is on.

III. Encoding of the problem by safety specification

We encode the specification for the controllers using rationnal expressions which denote the occurence of an error in the execution. We assume there are n tanks, numbered from 1 to n.

a) Specification of each tanks: First, an error occurs if a button was pressed but the tank is not filled after some delay smaller than $d \in \mathbb{N}$:

$$A_i = \mathsf{true}^* \cdot \{\mathsf{push}_i\} \cdot \{\neg \mathsf{fill}_i\}^d.$$

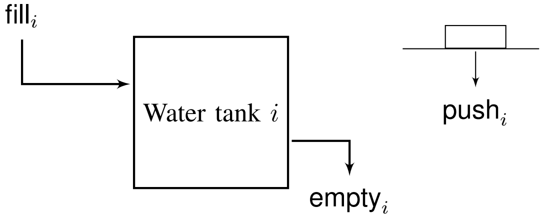

Fig. 1. A water tank. The input fill_i controls the arrival of water into the tank: if it is on, the valve is open and the water flows in. The input empty_i controls the departure of water out of the tank: if it is on, the valve is open and the water flows out. The washing cycle is initiated by pushing a button, which puts the inputs push_i to true.

The tank should also be filled only if the button was pushed:

$$B_i = \mathsf{true}^* \cdot \{\neg \mathsf{push}_i\}^d \cdot \{\mathsf{fill}_i\}.$$

There is an error if the valve is not open after a delay exactly k, or if the tank was emptied too early (before k steps).

$$C_i = \mathsf{true}^* \cdot \{\mathsf{fill}_i\} \cdot \mathsf{true}^k \cdot \{\neg \mathsf{empty}_i\}$$
$$C'_i = \mathsf{true}^* \cdot \{\mathsf{fill}_i\} \cdot (\mathsf{true} \mid \varepsilon)^{k-1}\{\mathsf{empty}_i\}$$

b) Light specification: Now the light should be on if, and only if, one of the n tanks is being filled, so the following expression denotes error:

$$D = \mathsf{true}^* \cdot \left\{ \mathsf{light} \neq \bigvee_{i \in [1,n]} \mathsf{fill}_i \right\}.$$

c) Mutual exclusion: Some of the tanks can be provided in water through the same pipe. In that case, the system should not fill to of them at the same time, othewise the tanks would not recieve enough water. This is illustrated in Fig. 2. We write P for the sets of pipe, and for a pipe $p \in P$, we write $i \in p$ to denote that the tank i is fed by pipe p. We encode the mutual exclusion constraint by raising an error for the following expression:

$$E = \mathsf{true}^* \cdot \left\{ \bigvee_{p \in P} \bigwedge_{i,j \in p} \mathsf{fill}_i \right\}$$

IV. Translation to AIGER

We consider the safety specification given by the language:

$$\mathcal{L} = \bigcup_{i \in [1,n]} (L(A_i) \cup L(B_i) \cup L(C_i) \cup L(C'_i)) \cup L(D) \cup L(E)$$

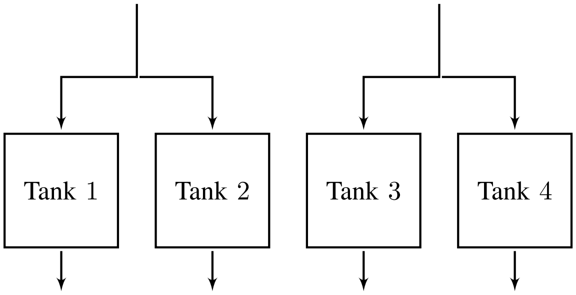

Fig. 2. A system with 4 water tanks. Tanks 1 and 2 are alimented by the same pipe and so do tanks 3 and 4.

Where $L(A)$ denotes the languages associated to expresssion A. The error output of the AIGER files will be set to true whenever the current prefix belong to the language $\mathcal{L}$. To produce the AIGER file, the rational expressions are translated to non-deterministic automata $\mathcal{A}$, then each state of $\mathcal{A}$ is encoded by a latch in the AIGER file. During an execution of the circuit, a latch l is on if, and only if there is an execution of $\mathcal{A}$ on the word that as been read that leads to the state corresponding to latch l. The error latch is put to true if, and only if, one of the latches corresponding to accepting states is on.

In the benchmark package, the AIGER files are named `cycle_sched_`n`_`d`_`p`.aag` where n is the number of tanks, d is the maximum delay between push_i and fill_i and is equal to k the exact delay between fill_i and empty_i, and p is the number of tanks that are alimented by the same pipes.

The program that has been used to generate AIGER files from rationnal expressions can be downloaded from the following address: https://github.com/romainbrenguier/Speculoos.

References

[1] A. Biere. Aiger format and toolbox. [Online]. Available: http://fmv.jku.at/aiger/

[2] S. Jacobs. (2014, February) Extended aiger format for synthesis (v0.1). [Online]. Available: http://www.syntcomp.org/wp-content/uploads/2014/02/Format.pdf