The Gossiping Persons Assignment
Modelling and Verification

Bogi Wennerstrøm
Morten Hartvigsen
Sigmundur Vang

April 2019

The Models

The Base Model

We chose to create a parameterized generic girl model such that each girl has a unique ID. This decision made changing the number of girls as easy as setting a variable. These models are the ones designated to model the gossiping girls with only minor variations between the different scenarios.

The idea behind the model structure was that there were four main events in one "cycle" of the girls: being in idle, calling another girl, getting called and sharing secrets. The life-cycle would then result in the girls first being in idle, then calling another girl or getting called, after which they will share secrets and then go back to idle.

We chose to use a binary format to represent the known secrets of each girl. Each girl started knowing its own bit; so girl 1 knew 0001 and girl 2 knew 0010 and so on. In this way, if girl 1 and 2 shared secrets, we could use the bit-wise *or* operator on their secrets together.

We chose not to centralize the communication with a distributor, as we found the notion of the girls communicating with each other by themselves more appealing.

A consequence of not having a distributor is that the girls should select another girl themselves. We use *select* for this. If girl 1 calls girl 2, then girl 1 should remember that she is talking to girl 2 until they hang up; we, therefore, store the ID 2 in girl 1 until she transitions to idle again

Linear Chain

To solve scenario one, we use the base model described in and add two additional functions, namely *next* and *prev.*

The *next* and *prev* functions are modelled such that they can accommodate for the linear arrangement of the girls. The functions are rather simple, *prev* just takes the current ID and subtracts one for the case where $ID > 0$ otherwise if $ID = 0$ *prev* adds one instead. The *next* function does the opposite of *prev*.

Circular Chain

The only difference between the linear chain and the circular chain is in how the *next* and *prev* functions handle the edge cases. When *next* got called in scenario 1 with an $ID = N$, where N is the number of girls, it just subtracted one from the ID instead of incrementing. In the circular chain arrangement, *next* handles the edge case by using modulo instead, which results in .

$$(\text{ID} + 1) \mod N$$

prev is computed in a similar fashion

Total Graph

In scenario 3, where every girl can call every other girl, we do not need a *next* and *prev* function as each girl now chooses a random girl with an ID between 0 and $N - 1$. If we add the constraint that a girl cannot call itself, then we have a sufficient model.

One Call at a Time

Minimal Number of Phone Calls

To find the minimal number of phone calls needed in order to ensure every girl knows every secret, we chose to use a counter declared at the global scope, that would then be incremented on each call.

By adding a guard for calling, ensuring that one cannot call if calls have reached some limit, then we can decrease the limit until there no longer exists a solution. When we reach the point where there no longer exists a solution, the limit + 1 will be the answer. The solutions for Scenario 1, 2, and 3 are shown in .

Table 1: Minimal Phone calls required for the different models

Linear	Circular	Any	Blocking
5	4	4	4

Timed Version

Ensuring each phone-call being exactly 60 time-units is achieved by guarding the edge going from in-call to idle. This guard ensures that 60 time-units have passed since the start of the call.

After updating the model to reflect the requirement of each phone-call being exactly 60 time-units, we did almost the same thing as in for finding the minimal time needed. The only thing changed, was instead of restricting calling, we restricted hanging up. The solutions for Scenario 1, 2, and 3 are shown in .

Table 2: Minimal time required for the different models

Linear	Circular	Any	Blocking
180	120	120	240

Experimentation and Message Sequence Chart

Upping the girl-count to 5 did increase the search time for all, but Scenario 3 sky-rocketed in time. We also tried with six girls, but this took too much time. The solutions for Scenario 1, 2, and 3 are shown in .

Table 3: Minimal Phone calls required for the different models

Type	Linear	Circular	Any	Blocking
Calls	7	7	6	6
Timed	300	240	240	360

Final Thoughts

Working with UPPAAL was really fun partly because it allowed us to put into the practice the theory that we have been learning in the past semester in the course. What made it really fun is that we got to work with a side of computer science that allowed us to conclude that if the premises of our model is upheld then the model satisfies some property with an 100% guarantee. We have never been able to say that in our (short) career as computer scientist. So to go through the trouble of modelling the system that we wanted to test and then see the green lamps in the verifier panel was really satisfying.