Uppgift 2

The Smoker’s Problem

Parallella processer G1F

Grupp A4

Adam Näslund a13adana  
Victor Karlsson c12vicka  
Alexander Milton b13alemi

## 1 Inledning

*In this section, you should give a short description of what you have done and why.*

Uppgiften bestod i att simulera en situation där ett antal individer står samlade runt ett bord. En av individerna är en agent och de övriga är rökare. Varje rökare har en oändlig mängd av en av tre resurser som krävs för att röka; tobak, papper eller tändstickor. Agenten har däremot en oändlig mängd av samtliga resurser och lägger upp två olika resurser på bordet så länge det är tomt som rökarna måste plocka upp och konsumera.

Programmet skall exekvera i en parallell programmeringsmiljö och verifieras i enlighet med ett antal regler och restriktioner. Krav finns på hur programmet får struktureras för att inte kringgå problemet.

## 2 Problembeskrivning

*In this section, you shall describe the problem that you try to solve. It is the problem presented in the assignment but you must describe it in your own words and relate it to the classic problems. This is also the correct place to include a discussion about necessary properties of a solution (e.g., global invariant) and any kind of difficulties that you foresee.*

*Smoker’s Problem* består av en miljö med ett antal rökare och en agent. Agenten förfogar över en oändlig mängd av de resurser som krävs för att producera en cigarett; tobak, papper och tändstickor. Agenten placerar iterativt två resurser av olika typer på bordet och meddelar med hjälp av meddelandepassering rökarna om vilka resurser som placerats på bordet. Rökarna har i sin tur en oändlig mängd av endast en resurstyp. Utan att kommunicera med varandra måste rökarna avgöra om de själva behöver de båda resurserna som ligger på bordet för att kunna producera en cigarett och i så fall plocka upp och konsumera dem.

### 2.1 Initialt program

I den första versionen av programmet ska enbart ett antal

Programmet ska stödja ett arbiträrt antal rökare som alla är i behov av olika resurstyper, även om flera rökare behöver samma två resurser. Det kommer alltid i alla delar av uppgiften finnas minst tre rökare, en för varje typ av resurs.

Uppgiften kan liknas vid det klassiska problemet *Producer/Consumer* där två olika generella typer av processer existerar; en typ som producerar några eller någon typ av resurs och en annan typ som konsumerar de eller den resursen. Antalet processer oavsett typ kan variera (t.ex. kan flera producenter generera resurser till en ensam konsument eller vice versa). I det här faller agerar agenten som en typ av producent som kontinuerligt levererar två olika typer av resurser medan rökarna, trots att de förser sig själva med den tredje resursen, är agerar konsumenter i sammanhanget. Med hjälp av producentens resurser möjliggörs konsumtion hos rökarna.

Ett antal globala invarianter kan etableras för att beskriva vilka förhållanden som måste hålla under hela programmets exekvering:

<Globala invarianter>

## 3 Method/Solution

### 3.1 Design decisions

*In this section, you shall discuss your approach to solve the problems identified in previous section. Include the design decisions in a stepwise manner. The difference between a design decision and an implementation has to do with the level of details. For example, processes, responsibilities and communication (who talks with whom) are all design decisions.*

### 3.2 Verification decisions

*In this section, you shall discuss which properties that your solution must have to be correct. ou shall also discuss how you are going to verify each of these properties, e.g., which tests to run and which queries to give the model checker.*

## 4 Results

### 4.1 Implementation

*The source code need not be included in the report since you are supposed to submit the sr files to your drop box. However, this is the place where you can discuss the code you have written. Do you have any comments to your source code?*

### 4.2 Verification

*This section contains your arguments for your solution meeting, or not meeting, the requirements. Are you confident with your solution? With what inputs did you test the program? To what extent have you used uppaal for verification? What queries did you run and what was the result of this? List your queries with the results you got from the model checker and explain what these results mean. For example: A[] !deadlock, result true, means that you have absence of deadlock. Given the verification, under what circumstances can you be sure that your program is correct?*

## 5 Discussion

*This section contains a discussion of whether the problem was successfully solved. Have you identified any problems with your solution? Can your solution be improved and if so how? Was it anything that surprised you during the verification? Note: it’s seldom a good approach to conclude that the program is perfect. I want to see that you can estimate your success.*