3rd Seminar – Process models, cooperating behaviour models

1 Modelling a complex system

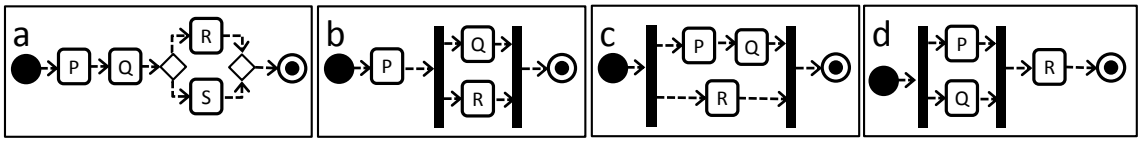
We are modelling a cloud-based data storage (e.g. Dropbox, Google Drive, Tresorit) with only one file. Both the server and the client (e.g. a laptop) has a replica of the file, initially with *identical* contents. Modifications of the file are transmitted via *synchronization*. A *conflict* occurs if both instances are modified before synchronization and the user has to resolve this conflict on the client.
The file can become modified locally on the client or on the server (e.g. due to the work of another client). The client and server may synchronize their content due to a user command or spontaneously from time to time. At that point the changes (if there is any) are transmitted to the other file and the replicas became *identical* again. However, if both files have been modified independently since the last synchronization, then a *conflict* occurs. In this case the client compares the local and remote versions of the file and the user must resolve the conflict.

a. Model the (partial) behaviour of the laptop client with a state machine. Initially the client is in *identical* state (the local file is identical to the one on the server at the time of the last synchronization), but the *write* input causes it to transition to *dirty* state (and it stays there after further *write* inputs). As a result of a *revert* input, it moves from any state to *identical* state.
b. The possible states of the server (regarding only the synchronization with the client) include *identical* and *updated*. A different user (or the same user using another client, e.g., a smart phone) with write permissions might update the replica on the server.
c. If the server is in *identical* state, then as a result of *synchronize* input the client uploads the local modifications (if there is any) to the server and moves to *identical* state. The server also receives the *synchronize* input. Where do the two automata cooperate?
d. If the server's current state is *updated*, then, as a result of *synchronize* input given on the client, the server transitions to *identical* state. The client either stays in *identical* state, or transitions from *dirty* to *conflict* state. What does this mean? What should happen in the conflict state? Where do the two automata cooperate?
e. Sometimes the client synchronizes with the server by itself, without any user input. What does it mean? Where do the two automata cooperate?
f. Create the product state space of the client-server system based on the two automata in the mixed product.
g. (*Bonus task*) The server and client can check each other's state directly in this model and the synchronization also happens instantaneously. However, the communication between the client and the server is implemented with messages in a real life distributed system. Some time elapses between sending the message and receiving an answer. How can we refine the model in order to incorporate this behaviour?

2 Process execution

We observed every step during the execution of a process. We detected the following event sequence:

> Process started, P started, P completed, Q started, R started, Q completed, R completed, Process completed.



Which ones can be valid models of the observed system from the business process models a, b, c, d?

3 Control flow based on source code

Consider the following function written in the C programming language.

```
1  unsigned long long f(int n)
2  {
3     if (n <= 0) {
4        return 0;
5     } else if (n == 1) {
6        return 1;
7     } else {
8        unsigned long long a = f(n - 1);
9        unsigned long long b = f(n - 2);
10       return a + b;
11    }
12 }
```

a. What is the control flow defined by the function?
b. Check whether the process is well-structured or not!
c. Identify the data dependencies (data flow) between the activities!
d. If the programming language or the runtime environment allows it, where can we parallelize the program?
e. (*Bonus task*) What ensures the termination of the function?

4 Control flow based on textual specification

The code repository (e.g. Git) of a big software foundation is home to the development of numerous open-source software. In addition to the reliable internal developers, outsiders can also send bugfixes or newly implemented features. It must be ensured that the published software only contains legitimately (e.g. with the consent of the employer) contributed source code.

a. If a developer would like to contribute to a project he or she must take some steps based on his or her status. Internal developers can write directly to the section of the code repository reserved for the specific project. Outsiders first have to submit their code to *code review*, after which an internal developer has to inspect it, and then either reject or approve it. If the code contributed by the outsider is sufficiently short (e.g. a small bugfix), then he or she only has to make a short statement of contribution in order to merge his or her code into the repository.
   In case of large outsider contributions (e.g. integrating a completely new module) the merging process is different. After the insider approval, the legal department of the foundation clarifies the legal status of intellectual property of the modifications via a dedicated administrative proceeding. Only after the successful closure of this process may the internal developer merge the code. Here, they make an exception in the case of freshly started projects, still before their first official release: merging the approved code into the repository does not need to wait until this legal proceeding is done. Create a process model based on these activities and constraints.
b. The development of a software project involves repeatedly modifying the source code, until the project management decides that the software is stable enough for an official release. At this point they publish a new stable version of the software, then it's the developers' turn again, and so on. Create a process model based on these activities.
c. (*Bonus task*) Check whether the processes are well-structured or not!
d. (*Bonus task*) What is the relationship between the process models designed in the previous sub-tasks?