Concurrency Theory
Exercise Sheet 7

Franks, Billy Joe
b_franks12@cs.uni-kl.de

Kahraman, Kerem
keremkahraman@hotmail.de

Bergsträßer, Pascal
p_bergstra15@cs.uni-kl.de

December 10, 2018

1 Alternating Bit Protocol in SPIN

```
// taken and modified from
// http://spinroot.com/spin/Doc/SpinTutorial.pdf

#define k 5
mtype {MSG, ACK};

chan toS = [k] of {mtype, bit};
chan toR = [k] of {mtype, bit};

proctype Sender(chan in, out)
{
        bit sendbit, recvbit;
        do
        :: out ! MSG, sendbit ->
                in ? ACK, recvbit;
                if
                :: recvbit == sendbit ->
                        sendbit = 1-sendbit
                :: else
                fi
        od
}

proctype Receiver(chan in, out)
```

```
{
        bit recvbit;
        do
        :: in ? MSG(recvbit) ->
                out ! ACK(recvbit);
        od
}

init
{
run Sender(toS, toR);
run Receiver(toR, toS);
}
```

2 Communicating State Machines with Bags (instead of FIFO)

2.1 CSM to PN

The idea is to simulate the CSM directly, via one set of places for the states and one set of places for the message channels.

For each state in machine in CSM we add a place in PN. For each Message Channel we add $|\Sigma|$ places. Then we encode each transition of some machine in CSM or as a transition in PN as follows:

$(s, ?a, s')$ take token from place s, add token to place s', take token from place id, a.

$(s, id!a, s')$ take token from place s, add token to place s', add token to place id, a.

The marking that should be covered is simply a 0 in all places, except for the state that should be reached, put a 1 here.

The equivalence is trivial.

The resulting PN has $f(N, m, |\Sigma|) = N(|\Sigma| + m)$ places.

The resulting machine has the same number of transitions as all the transitions in the CSM. Or at maximum $Nm(|\Sigma| + N|\Sigma|)$.

2.2 PN to CSM

The idea is to simulate the marking with a bag channel(multiset). And simulate transitions via a cycle(of multiple states) in a machine in the CSM starting at the so called common state. Finally the marking that should be reached is a path from the common state reading each message defining a token in the marking.

For each place in PN add a message in Σ in the CSM. We will assume a machine can send to itself, otherwise we simply use the copy machine from the lecture. For each transition in the PN add a cycle of states starting at the common state, first reading each message defining a token, then writing each message defining a token into its channel. Finally for the marking that should be covered add a path from the common state reading each message defining a token in the marking until the marking is completely read.

The state that should be reached is the final state of this path.

Again the equivalence is trivial.

3 Lossy Channel Systems with Test for Empty Channel

LCS with test for empty channels are still WSTS. We keep the same $\leqslant$ relation as in the proof for LCS. It is still a WQO since the semantic rules are never used. We also get stuttering compatibility by dropping messages in the greater system until the contents of the channels are equal to those of the smaller system. Thus, if the smaller system applies an empty channel transition, we just drop all messages in the corresponding channels of the greater system and apply the same empty channel transition.