Altogether, 32 points were available. However, the calculation of the marks was made on the basis of 28 points.

1. Binary decision diagrams.

The three parts of this exercise gave 2+3+5 points, respectively.

(a) Each of the two BDDs was worth one point. In the drawing below, common subgraphs of the two BDDs are already shared. Everybody, without exception, obtained these two points.

(b) The expected equation is as follows (half a point for each of the six cases):

\[
F \rightarrow G = \begin{cases} 
1 & \text{if } F = 0 \\
G & \text{if } F = 1 \\
\neg F & \text{if } G = 0 \\
1 & \text{if } G = 1 \\
1 & \text{if } F = G \\
it\text{e}(x, F[x/1] \rightarrow G[x/1], F[x/0] \rightarrow G[x/0]) & \text{otherwise}
\end{cases}
\]

(c) Here’s how the recursive algorithm would work in this case: Each box indicates one recursive call, the names are the nodes from (a). Next to each box the result is denoted (’!’ stands for negation). The five terminal cases correspond to those of the equation in (b). An incorrect result in one case resulted in the deduction of one point.
The resulting BDD is shown on the right-hand side. Notice that the right-hand side of the box \((C, B)\) collapses to 1. Failing to merge identical nodes or removing obsolete nodes resulted in the deduction of half a point. The negation of node \(C\) had to be made “manually”. While this was not particularly difficult, with hindsight I would have moved this case further down in the tree.

2. BDDs with complement

General remark: Every negation bit inverts the meaning of the CBDD starting at its target. Thus, to decide whether a given valuation is accepted by a CBDD, one can count the number of negations on the corresponding path from the initial edges to the unique leaf. If it is even, the answer is yes, otherwise no.

The three parts of this exercise gave 1+4+2 points, respectively.

(a) To show non-uniqueness, it suffices to provide an alternative for \(x\), the function represented by CBDD \(A\). Everybody obtained the point for this exercise.

(b) Suppose that we have a node \(n\) whose outgoing 0-edge has a positive negation bit. We mentioned above that along each path only the parity of the number of negation bits matters. Now, if we invert the negation bit on all adjacent edges of \(n\) (the 0-edge, the 1-edge, and all incoming edges), then the parity on all paths remains the same. On a logical level, if \(n\) is labelled by \(x\), then the CBDD starting at \(n\) represents a formula \(\text{ite}(x, F, G)\) (for some suitable \(F, G\)). This can be shown to be equivalent to \(\neg\text{ite}(x, \neg F, \neg G)\), which in turn is represented by the result of the transformation.

Naturally, this procedure may create negated 0-edges that did not exist before. However, these appear only on edges that are “higher up”, i.e. closer to the root. Therefore, one starts by successively eliminating those negated 0-edges that are closest
to the leafs. One can easily see that this terminates. (E.g., one could consider the tuples \((n, m)\) with lexicographic ordering, where \(m\) is the highest variable index of a node with an outgoing 0-edge, and \(n\) is the number of such nodes; this tuple strictly decreases during each operation.)

For the correction, I accepted a short explanation of the above method. However, failing to justify the correctness of the method (even briefly) led to the deduction of one point.

(c) For \(A\), see above. For \(C\), see below, it is the result of the procedure from (b). Four inversions were necessary for \(A\) and \(C\), an error in any of them meant a deduction of half a point.

\[\begin{array}{c}
\text{\includegraphics{petrinet1.png}} \\
\text{\includegraphics{petrinet2.png}}
\end{array}\]

3. Reachability in Petri nets.

The two parts of this exercise gave 4+3 points, respectively.

(a) Remark: To shown that the reachability problem for \(A\), the class of acyclic 1-safe nets, is in NP, we must show that a non-deterministic machine can answer ‘yes’ for a positive instance in polynomial time. It is therefore not sufficient to show that the possible executions of the net are of finite length. It is also not enough to say that the paths in the net are of polynomial length since the execution of a Petri net is not ‘linear’ but concurrent. Also, firing a transition may, e.g., increase the number of tokens.

However, one can show that in any execution of a net \(N \in A\), no transition can fire twice: First, 1-safeness implies that no transition has an empty preset, otherwise the transition could produce infinitely many tokens. (Strictly speaking, a transition could have empty preset and postset, but then it would become meaningless for reachability.) Therefore, if a transition \(t\) could fire twice in some execution, then some place \(p \in \mathit{\bullet}t\) must receive two tokens during the execution. Since the net is acyclic, firing \(t\) (consuming the first token) cannot be responsible, directly or indirectly, for the generation of the second token. Thus, any execution in which \(t\) fired twice could be reordered to put two tokens onto \(p\). But then the net \(N\) would not be 1-safe.

Thus, any execution of \(N\) consists of at most \(|T|\) transition firings. A nondeterministic Turing machine can simulate a firing sequence with space \(|P|\) (to remember the marking), answer ‘yes’ if the desired marking is reached, or ‘no’ if it is not reached after \(|T|\) steps.

(b) Recall that SAT is a restriction of the QBF problem to existential quantification. In class, we discussed how to reduce QBF to the reachability problem for 1-safe nets; this reduction produced acyclic nets when only existential quantification was involved. Thus, it was sufficient to reproduce said proof. The construction below is a minor modification of it.
Let \( X \) be a set of Boolean variables and \( F = \bigwedge_{i=1}^{k} \bigvee_{j=1}^{m_i} \ell_{i,j} \) be a formula in CNF, where the literals \( \ell_{i,j} \) are positive or negative instances from \( X \); denote by \( \mathcal{L} \) the set of those literals. We construct the following net \( N_F \):

- The places of \( N_F \) are \( P_X \uplus P_L \uplus P_C \uplus P'_C \uplus \{ p \} \), where
  - \( P_X := \{ p_x \mid x \in X \} \) (one place per variable);
  - \( P_L := \{ \ell_{i,j} \mid \ell_{i,j} \in \mathcal{L} \} \) (one place per literal);
  - \( P_C := \{ c_i \mid 1 \leq i \leq k \} \) and \( P'_C := \{ c'_i \mid 1 \leq i \leq k \} \) (two places per clause).
- The initially marked places are \( P_X \uplus P_C \).
- The transitions are \( T_T \uplus T_L \uplus T_L' \uplus \{ t \} \uplus T'_L \), where
  - \( T_T := \{ t^x \mid x \in X \} \), \( \bullet t^x = \{ p_x \} \), \( t^x \bullet = \{ p_{i,j} \mid \ell_{i,j} = x \} \) (choose to generate tokens for all positive \( x \)-literals);
  - \( T_L := \{ t_{i,j} \mid \ell_{i,j} \in \mathcal{L} \} \), \( \bullet t_{i,j} = \{ c_i, p_{i,j} \} \), \( t_{i,j} \bullet = \{ \ell'_{i,j} \} \) (mark clause \( c_i \) as satisfied if any literal is marked);
  - \( \bullet t := \{ \ell'_i \mid 1 \leq i \leq n \} \), \( t \bullet := \{ p \} \) (put a token on \( p \) if all clauses are satisfied);
  - \( T_L' := \{ t'_{i,j} \mid \ell_{i,j} \in \mathcal{L} \} \), \( \bullet t'_{i,j} = \{ p_{i,j} \} \), \( t'_{i,j} \bullet = \emptyset \) (allow the token for any literal to disappear).

It is easy to see that the size of \( N_F \) is polynomial (linear) w.r.t. the size of \( F \) and that the marking \( \{ p \} \) can be reached if and only if \( F \) is satisfiable. Moreover, the net \( N_F \) is acyclic and 1-safe.

4. Bisimulation

The four parts of this exercise gave 3+1+3+1 points, respectively. Only few people managed to do the parts (c) and (d) of the exercise due to time problems; they were therefore treated as bonus (see above.)

(a) The proof follows from the following facts:

(i) Trivially, the identity relation is a bisimulation.

(ii) If \( R \) is a bisimulation, then so is \( R' := \{ (t, s) \mid (s, t) \in R \} \) (obvious, due to the symmetric definition of bisimulation).

(iii) If \( R, R' \) are bisimulations, then so is \( R'' := R \circ R' \). Proof (only one part, the other is symmetric): Let \( (s, u) \in R'' \), then there exists \( t \) such that \( (s, t) \in R \) and \( (t, u) \in R' \). Suppose \( s \xrightarrow{a} s' \), then by definition of bisimulation there exists \( t' \) with \( t \xrightarrow{a} t' \) and hence \( u \xrightarrow{a} u' \); moreover, \( (s', t') \in R \) and \( (t', u') \in R' \), hence \( (s', u') \in R'' \).

Remark: Many answers wrongly claimed that each bisimulation was symmetric resp. transitive.

(b) The equivalence classes of \( \sim \) are \( \{1, 3, 5, 7\} \) and \( \{2, 4, 6\} \).

(c) Again, due to symmetry, it is only necessary to prove one part of the bisimulation property.

The proof is visualized by the matrix shown below. Suppose that (i) \((s, t) \in R_{\sim}\) and (ii) \( s \xrightarrow{a} s' \). To prove that \( R_{\sim} \) is a bisimulation, we must prove that there exists \( t' \) with \( t \xrightarrow{a} t' \) and \( (s', t') \in R_{\sim} \).

By (i), there exist \((s_0, t_0) \in R\) with \( s \sim s_0 \) and \( t \sim t_0 \). By (ii) and definition of \( \sim \), there exists \( s'_0 \) with (iii) \( s_0 \xrightarrow{a} s'_0 \) and \( s' \sim s'_0 \). From (iii) and the fact that \( R \) is a
bisimulation up to $\sim$, there exists $t'_0$ with (iv) $t_0 \xrightarrow{\sim} t'_0$ and (v) $(s'_0, t'_0) \in R_{\sim}$. Fact (iv) and $t \sim t_0$ imply the existence of some $t'$ with $t \xrightarrow{\sim} t'$ and $t' \sim t'_0$. Fact (v), by definition, implies that there exists a pair $(s''_0, t''_0) \in R$ such that $s'_0 \sim s''_0$ and $t'_0 \sim t''_0$. Now, $(s', t') \in R_{\sim}$ follows from the definition of $R_{\sim}$ and transitivity of $\sim$.

\[
\begin{array}{cccccc}
  s & \sim & s_0 & R & t_0 & \sim & t \\
  \downarrow a & & \downarrow a & & \downarrow a & \\
  s' & \sim & s'_0 & R_{\sim} & t'_0 & \sim & t' \\
  & & \sim & & \sim & \\
  & & s''_0 & R & t''_0 & \\
\end{array}
\]

(d) A minimal example is $R = \{(1, 3), (2, 4)\}$, in which case $R_{\sim} = \sim$. 