1. Textbook, Page 271, Exercise 7.5.

   \textbf{Proof.} Not satisfiable.
   
   \[(x \lor y) \land (x \lor \overline{y}) \land (\overline{x} \lor y) \land (\overline{x} \lor \overline{y}) \equiv ((x \lor (y \land \overline{y})) \land ((\overline{x} \lor (y \land \overline{y})))) \equiv x \land \overline{x} \equiv F\]


   a) Union: \(L_1 \in P, L_2 \in P \Rightarrow L_1 \cup L_2 \in P\)

   \textbf{Proof.} \(L_1 \in P, L_2 \in P \Rightarrow L_1, L_2 \) are decidable in polynomial time on deterministic single-tape TM \(M_1\) and \(M_2\). Suppose \(M_1\) runs in time \(O(n^s)\) and \(M_2\) runs in time \(O(n^l)\). A polynomial time deterministic single-tape TM \(M\) operates as follows:

   On input \(x\):
   
   1. Simulate \(M_1\). If \(M_1\) accepts \(x\), then ACCEPT, otherwise goto step 2.
   2. Simulate \(M_2\) on input \(x\). If \(M_2\) accepts \(x\), then ACCEPT, otherwise, REJECT.

   Apparently, this TM \(M\) decides language \(L_1 \cup L_2\). Now we analyze this algorithm to show that it runs in polynomial time. Step 1 takes \(O(n^s)\) time and Step 2 takes \(O(n^l)\) time. So \(M\) runs at most \(O(n^s + n^l)\) time. Hence \(M\) is a polynomial time deterministic Turing Machine.

   b) Concatenation: \(L_1 \in P, L_2 \in P, L = \{uv | u \in L_1, v \in L_2\} \Rightarrow L \in P\)

   \textbf{Proof.} \(L_1 \in P, L_2 \in P \Rightarrow L_1, L_2\) are decidable in polynomial time on deterministic single-tape TM \(M_1\) and \(M_2\). Suppose \(M_1\) runs in time \(O(n^s)\) and \(M_2\) runs in time \(O(n^l)\). A polynomial time deterministic single-tape TM \(M\) operates as follows:

   On input \(x\)
   
   FOR each division \(x = uv\) of the input DO
   
   \{
   1. Copy \(u\) on the tape. Go to step 2;
   2. Simulate \(M_1\) on input \(u\). If \(M_1\) accepts \(u\), then go to step 3, otherwise, BREAK;
   3. Copy \(v\) on the tape. Go to step 4;
   4. Simulate \(M_2\) on input \(v\). If \(M_2\) accepts \(v\), then ACCEPT, otherwise, BREAK;
   \}

   REJECT.

   Apparently, this TM \(M\) decides the concatenation of \(L_1\) and \(L_2\). Now we analyze this algorithm to show that it runs in polynomial time. In each loop, it costs at most \(O(n^s + n^l + n) = O(n^{\max(s,l)})\) time. There are \(n + 1\) divisions of input \(x\), so it repeats at most \(n + 1\) times. The running time of \(M\) is at most \(O(n^{\max(s,l)} \star (n + 1)) = O(n^{\max(s,l)} + 1)\). Hence \(M\) is a polynomial time deterministic Turing Machine.
c) Complement: \( L \in P \Rightarrow \overline{L} \in P \)

Proof. \( L \in P \Rightarrow L \) is decidable in polynomial time on a deterministic single-tape TM \( M \). Suppose \( M \) runs in time \( O(n^k) \) A polynomial time deterministic single-tape TM \( M' \) operates as follows:

On input \( x \):
1. Simulate \( M \). If \( M \) accepts \( x \), then REJECT, otherwise ACCEPT.

Apparently, this TM \( M' \) decides language \( L \) since \( M' \) runs in time \( O(n^k) \). \( \square \)

a) Union: \( L_1 \in NP, L_2 \in NP \Rightarrow L_1 \cup L_2 \in NP \)

Proof. \( L_1 \in NP, L_2 \in NP \Rightarrow L_1, L_2 \) are decided by nondeterministic polynomial time TM \( M_1 \) and \( M_2 \). Suppose \( M_1 \) runs in time \( O(n^s) \) and \( M_2 \) runs in time \( O(n^l) \). We can build a 3-tape nondeterministic TM \( M \) that recognize \( L_1 \cup L_2 \):

On input \( x \):
1. Copy input \( x \) to both Tape 2 and Tape 3;
2. Simultaneously simulate \( M_1(x) \) on Tape 2 and \( M_2(x) \) on Tape 3;
3. Wait until one of the simulation halts and accepts;
4. ACCEPT.

Apparently, this TM \( M \) recognizes language \( L_1 \cup L_2 \). Now we analyze this algorithm to show that it runs in polynomial time. Step 1 takes \( 2^n \) time and Step 2 takes \( O(n^{\max(s,l)}) \) time. So \( M \) runs at most \( O(n^{\max(s,l)}) \) time. Hence \( M \) is a polynomial time nondeterministic Turing Machine. \( \square \)

b) Concatenation: \( L_1 \in NP, L_2 \in NP, L = \{uv | u \in L_1, v \in L_2\} \Rightarrow L \in NP \)

Proof. \( L_1 \in NP, L_2 \in NP \Rightarrow L_1, L_2 \) are decided by nondeterministic polynomial time TM \( M_1 \) and \( M_2 \). Suppose \( M_1 \) runs in time \( O(n^s) \) and \( M_2 \) runs in time \( O(n^l) \). We can build a 3-tape nondeterministic TM \( M \) that recognize the concatenation of \( L_1 \) and \( L_2 \):

On input \( x \)
\[ s = 1; \]
REPEAT
FOR each division \( x = uv \) of the input DO
\{
1. Copy \( u \) to Tape 2 and \( v \) to Tape 3;
2. Simulate \( M_1 \) on Tape 2 for \( s \) steps;
3. Simulate \( M_2 \) on Tape 3 for \( s \) steps;
4. If both simulations ended in accept, then HALT and ACCEPT;
\}
\[ s = s + 1; \]
FOREVER.

Apparently, this TM \( M \) decides the concatenation of \( L_1 \) and \( L_2 \). Now we analyze this algorithm to show that it runs in polynomial time. In each FOR loop, it costs at most \( 2s \) steps. There are \( n + 1 \) divisions of input \( x \), so it repeats at most \( 2s(n + 1) \) times. \( s \) could be at most \( \max(O(n^s), O(n^l)) \) time. The running time of \( M \) is less than \( O(2(n^\max(s,l))^2(n + 1)) = O(n^{2\max(s,l)+1}) \). Hence \( M \) is a polynomial time deterministic
Turing Machine.