Hints:

- Please solve these exercises in **groups of two**!
- The solutions must be handed in **directly before (very latest: at the beginning of)** the exercise course on Wednesday, November 10th, 2010, in lecture hall AH 3. Alternatively you can drop your solutions into a box which located right next to Prof. Giesl’s office (until the exercise course starts).
- Please write the **names** and **immatriculation numbers** of all (two) students on your solution. Also please staple the individual sheets!
- As the topics related to the exercises on this sheet are relevant for all versions of the lecture, **all exercises** on this sheet are relevant for **all students**.

---

**Exercise 1 (Conversion to CNF):** (2 points)

Consider the following formula $\phi$ with $p_1, \ldots, p_6 \in \Delta_0$:

$$
\phi = ((p_1 \lor \neg p_2) \land (\neg p_3 \lor p_4)) \rightarrow (p_5 \land \neg p_6)
$$

Use the algorithm presented in the proof of Theorem 3.3.2 to convert $\phi$ to an equivalent formula in conjunctive normal form (CNF).

---

**Exercise 2 (Tseitin’s transformation):** (3 + 1 + 2 = 6 points)

If one uses the algorithm defined in the proof of Theorem 3.3.2 to convert a propositional formula $\psi$, in the worst case the size of $\psi'$ can be exponential in the size of $\psi$. Therefore, in practice instead one usually applies a different algorithm (attributed to Tseitin) in order to convert a formula to CNF. Here the idea is to introduce additional propositional variables $q_\theta$ for every non-atomic subformula $\theta$ of the formula $\psi$. Then one adds constraints in CNF which ensure that $q_\theta \leftrightarrow \theta$ holds. In the end one adds the singleton conjunct $q_\psi$ to the CNF to enforce that any model of the CNF will also satisfy $\psi$.

Consider the following example with $p_1, p_2, p_3 \in \Delta_0$:

$$
\tau = (p_1 \land p_2) \lor p_3
$$

The formula $\tau$ contains the three non-atomic subformulas $\tau_1$, $\tau_2$, and $\tau$. Hence, we introduce the three new predicate symbols $q_{\tau_1}$, $q_{\tau_2}$, and $q_\tau$ which are added to $\Delta_0$ (i.e., $q_{\tau_1}$, $q_{\tau_2}$, and $q_\tau$ are propositional variables which can be assigned to true or to false). Now we still need to ensure that they are assigned the correct truth values:

- $q_{\tau_1} \leftrightarrow \neg p_2 \land
- q_{\tau_2} \leftrightarrow p_1 \land q_{\tau_1} \land
- q_\tau \leftrightarrow q_{\tau_2} \lor p_3 \land

We eliminate the $\leftrightarrow$-junctors and obtain the following CNF:

$$
(q_{\tau_1} \lor p_2) \land (\neg q_{\tau_1} \lor \neg p_2) \land
(q_{\tau_2} \lor p_1) \land (\neg q_{\tau_2} \lor q_{\tau_1}) \land (q_{\tau_2} \lor \neg p_1 \land \neg q_{\tau_1}) \land
(q_\tau \lor \neg q_{\tau_2}) \land (q_\tau \lor \neg p_3) \land (\neg q_\tau \lor q_{\tau_2} \lor p_3) \land
$$
To define Tseitin’s transformation $ts$ formally, we need the set of all non-atomic subformulas $na(\psi)$ for a propositional formula $\psi$. In our example, we have $na(\tau) = \{\tau_1, \tau_2, \tau\}$. For each of these non-variable subformulas $\theta$, we use a corresponding new variable $q_\theta$.

In order to make sure that the new variables $q_\theta$ take the correct values, we define formulas $C_-$, $C_\wedge$, and $C_\vee$ as follows. Here, $q_\theta$ should be replaced by $\theta$ whenever $\theta$ is an atomic formula.

- $C_-(\gamma, \theta) = (\gamma \lor \theta) \land (\neg \gamma \lor \neg \theta)$
- $C_\wedge(\gamma, \theta_1, \theta_2) = (\neg \gamma \lor \theta_1) \land (\neg \gamma \lor \theta_2) \land (\gamma \lor \neg \theta_1 \lor \neg \theta_2)$
- $C_\vee(\gamma, \theta_1, \theta_2) = (\gamma \lor \neg \theta_1) \land (\gamma \lor \neg \theta_2) \land (\neg \gamma \lor \theta_1 \lor \theta_2)$

Note that if the arguments of $C_-$, $C_\wedge$, and $C_\vee$ are atomic formulas, then the resulting formulas are in CNF. Now we can define Tseitin’s transformation $ts$ as follows. Here, $q_\theta$ should be replaced by $\theta$ whenever $\theta$ is an atomic formula.

$$ts(\psi) = q_\psi \land \bigwedge_{\theta \in na(\psi)} C_-(q_\theta, q_\psi) \land \bigwedge_{\theta_1, \theta_2 \in na(\psi)} C_\wedge(q_{\theta_1}, q_\theta_2, q_{\theta_1}, q_{\theta_2}) \land \bigwedge_{\theta_1, \theta_2 \in na(\psi)} C_\vee(q_{\theta_1}, q_\theta_2, q_{\theta_1}, q_{\theta_2})$$

Without loss of generality, here we assume that $\psi$ does not contain the junctors $\rightarrow$ and $\leftrightarrow$ (they can be eliminated by a preprocessing step as described in the proof of Theorem 3.3.2).

Remark: Note that Tseitin’s transformation to CNF does not preserve equivalence (because it introduces new predicate symbols $q_\theta$ which were not present in the original formula and which influence the truth value of the resulting CNF). Nevertheless, an important property of Tseitin’s transformation from a propositional formula $\psi$ to a CNF $ts(\psi)$ is that any model of $ts(\psi)$ is also a model of $\psi$ (and moreover, any model of $\psi$ can be extended to a model of $ts(\psi)$). Since most modern satisfiability checkers for propositional logic require their input to be in CNF, this transformation is very useful for extending applicability of these SAT solvers to arbitrary propositional formulas.

a) Please apply first the necessary preprocessing to eliminate the junctor $\rightarrow$ and then Tseitin’s transformation to the formula $\varphi$ from Exercise 1. In your solution you do not need to expand the subformulas $C_\circ(\ldots)$ for $\circ \in \{\neg, \land, \lor\}$.

b) How many clauses does your formula contain (after expanding $C_\circ(\ldots)$)?

c) Please provide a tight bound for the size complexity (in $O$-notation) for $ts(\psi)$ for a given propositional formula $\psi$ of size $n$. Here the size of a formula is the number of its junctors and its atoms. Please also give a short explanation for your result.

Exercise 3 (Tseitin’s transformation in Prolog): (6 points)

In this exercise we are going to implement Tseitin’s transformation in Prolog. To represent propositional formulas in Prolog, we use the binary function symbols $\text{and}$ and $\text{or}$ and the unary function symbols $\text{not}$ and $\text{atom}$. Here $\text{and}$ and $\text{or}$ correspond to the usual Boolean junctors. The function symbol $\text{atom}$ marks an atomic formula (i.e., a propositional variable which can become either $\text{true}$ or $\text{false}$). For instance, the term $\text{atom}(p)$ corresponds to the propositional variable $p$. Propositional formulas are translated to Prolog terms by the function $D$:

$$D(p) = \text{atom}(p) \quad \text{for all atomic formulas } p$$
$$D((\theta_1 \land \theta_2)) = \text{and}(D(\theta_1), D(\theta_2))$$
$$D((\theta_1 \lor \theta_2)) = \text{or}(D(\theta_1), D(\theta_2))$$
$$D(\neg \theta) = \text{not}(D(\theta))$$

For instance, for the formula $\tau$ from Exercise 2 we have:

$$D(\tau) = \text{or}(\text{and}(\text{atom}(p_1), \text{not}(\text{atom}(p_2))), \text{atom}(p_3))$$
To represent the fresh predicate symbols $q\theta$ that are introduced by Tseitin's transformation, we define:

$$D(q\theta) = \text{atom}(q_\_D(\theta))$$

So for example if $p \in \Delta_0$, then $D(q_\neg p) = \text{atom}(q_\_\text{not}(\text{atom}(p)))$.

As usual, we represent of CNFs via sets of clauses. For instance, the CNF

$$q_\neg p \land (q_\neg p \lor p) \land (\neg q_\neg p \lor \neg p)$$

corresponds to the clause set:

$$\{\{q_\neg p\}, \{(q_\neg p, p)\}, \{\neg(q_\neg p, \neg p)\}\}$$

In our translation to Prolog, we represent clauses by lists and clause sets by lists of lists. So the above clause set would be represented by:

$$[[\text{atom}(q_\_\text{not}(\text{atom}(p)))], [\text{atom}(q_\_\text{not}(\text{atom}(p))), \text{atom}(p)], [\text{not}(\text{atom}(q_\_\text{not}(\text{atom}(p)))), \text{not}(\text{atom}(p))]]$$

Please implement a predicate `tseitin/2` which takes a given propositional formula in the first argument. It then computes a list representation of the clause set returned by Tseitin’s transformation in the second argument.

For instance, the query `?- tseitin(not(atom(p)), Cs).` should have the result

$Cs = [[\text{atom}(q_\_\text{not}(\text{atom}(p)))], [\text{atom}(q_\_\text{not}(\text{atom}(p))), \text{atom}(p)], [\text{not}(\text{atom}(q_\_\text{not}(\text{atom}(p)))), \text{not}(\text{atom}(p))]]$.

(where list elements may occur in a different order).

On the web page of the course, you can find a file `tseitin.pl`, where it says `FILLME` in several places. Please replace `FILLME` to complete the implementation for the predicate `tseitinList/3`, which is used by `tseitin/2`.

```prolog
% Please replace "FILLME" by your implementation!
/* We construct propositional formulas as follows:
 * - atom(X) is a propositional variable with the name X.
 * - not(F) is the negation of the formula F.
 * - and(F, G) is the conjunction of the formulas F and G.
 * - or(F, G) is the disjunction of the formulas F and G.
 */

% tseitin/2:
% tseitin(F, Cs) holds if any model of F can be extended to a model
% of Cs and conversely any model of Cs is also a model of F.
% Here F must be instantiated by a ground term in a query, whereas Cs has no restrictions.
% tseitin(F, [[X]|Cs]) :- tseitinList(F, X, Cs).

% tseitinList/3 has the following intended meaning:
% tseitinList(F, X, Cs) holds if X is the propositional variable that
% represents the formula F, and Cs are clauses required to make sure
% that X holds iff F does.
% % Example:
% The query ?- tseitinList(not(atom(p)),X,Cs). has the result
% X = atom(q_not(atom(p))).
% Cs = [[atom(q_not(atom(p))), atom(p)], [not(atom(q_not(atom(p)))), not(atom(p))]].

% prop. variables from the input formula are represented by themselves
% tseitinList(atom(Y), atom(Y), []).

% Tseitin variable "names" for complex formulas are labeled
% by the formulas
% tseitinList(not(F), atom(q_not(F)), FILLME) :- FILLME.
% tseitinList(or(F,G), atom(q_or(F,G)), FILLME) :- FILLME.
% tseitinList(and(F,G), atom(q_and(F,G)), FILLME) :- FILLME.
```
Exercise 4 (Resolution for propositional logic): (3 points)

Consider the following clause set $\mathcal{K}$ with $p_1, \ldots, p_4 \in \Delta_0$:

$$\mathcal{K} = \{\{p_1, p_2\}, \{\neg p_2\}, \{\neg p_1, p_2, \neg p_4\}, \{p_3, p_4\}, \{\neg p_3, \neg p_4\}, \{\neg p_3, \neg p_1\}\}$$

Please show that $\mathcal{K}$ is unsatisfiable by using resolution for propositional logic (cf. Definition 3.3.4 and Example 3.3.5).

*Hint:* It suffices to perform five resolution steps.