Formal Methods in Computer Science SS 2019

Exercise Sheet for Block 2

This sheet contains a pool of exercises to practice the material presented in the lecture. You are free to complete as many exercises as you want. If you upload your solutions in TUWEL no later than ***April 28, 2019*** then you will get feedback to the exercises you tried. The corrected exercises will be handed back in the second consolidation class on ***May 7, 2019***.

We strongly recommend solving *at least* the following exercises: Exercise 2.1, Exercise 4, Exercise 5, Exercise 6, Exercise 11, Exercise 12, Exercise 13, and Exercise 15.

The questions of Block 2 in the final exam are going to have a strong emphasis on understanding and proofs. All exercises are relevant for the final exam.

Exercise 1. *Resolution*

1.1 Let $F = \{C_1, \ldots, C_5\}$ be a set of clauses where $C_1 = (c \vee b)$, $C_2 = (a \vee \neg b)$, $C_3 = (\neg a \vee \neg c)$, $C_4 = (a \vee b \vee \neg c)$, and $C_5 = (\neg a \vee \neg b \vee c)$. Use resolution to show that F is unsatisfiable. Explain why your approach is correct.

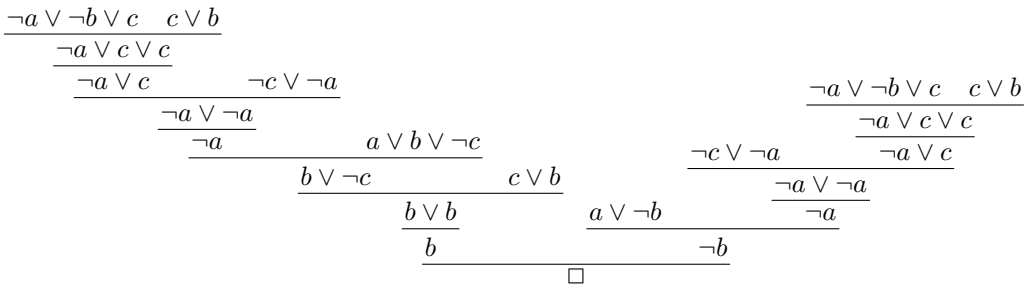
1.2 Given the formula $\lambda = a \wedge \neg b \wedge (a \leftrightarrow b)$, show that λ is unsatisfiable using resolution.

1.3 Let γ be a propositional formula. How can we show that γ is valid. Explain your approach.

1.4 Let $F = C_1 \wedge C_2 \wedge C_3$ be a propositional formula in CNF where $C_1 = (a \vee b)$, $C_2 = (\neg a \vee c)$, and $C_3 = (\neg b \vee d \vee \neg e)$. Show that the propositional formula $F \rightarrow (a \vee b \vee c)$ is valid. Is it possible to derive the clause $(a \vee b \vee c)$ from F by resolution? Explain your solution.

Solution 1.

1.1 We derive the empty clause $\square$ by resolution. In the following derivation tree, binary inferences are applications of the resolution rule whereas unary inferences are applications of the factoring rule.



Since we derived the empty clause $\square$, it follows that F is unsatisfiable.

1.2 We transform λ into a set of clauses $L = \{(a), (\neg b), (\neg a \vee b), (\neg b \vee a)\}$ by the standard translation procedure to clause normal form. Alternatively, the Tseitin translation could be applied.

$$\dfrac{\dfrac{(a) \qquad (\neg a \vee b)}{(b)} \qquad (\neg b)}{\square}$$

We derive by resolution the empty clause from L, thus L is unsatisfiable. Since L is equivalent to λ, it follows that λ is unsatisfiable.

1.3 A propositional formula is valid if and only if its negation is unsatisfiable, i.e., γ is valid iff $\neg\gamma$ is unsatisfiable. Transform $\neg\gamma$ to a set G of clauses such that G is sat-equivalent to γ (e.g. using Tseitin). If we can derive by resolution the empty clause from G, then G is unsatisfiable, hence $\neg\gamma$ is unsatisfiable and consequently γ is valid.

1.4 No solution given.

Exercise 2. *Tseitin Transformation (recommended)*

2.1 For the formula $\psi = (a \vee \neg b) \wedge (\neg c \rightarrow a)$ use Tseitin translation to compute a sat-equivalent CNF.

2.2 Transform the propositional formula $f(x_1, x_2, x_3) := ((x_1 \oplus x_2) \oplus x_3)$ into a sat-equivalent CNF using Tseitin's transformation, where '$\oplus$' denotes the XOR operation.

2.3 Related to exercise 2.2: let $n \geq 2$ and consider the propositional formula

$$f(x_1, \ldots, x_n) := (\ldots(x_1 \oplus x_2) \oplus \ldots) \oplus x_{n-1}) \oplus x_n).$$

What is the number of clauses in terms of n in a sat-equivalent CNF of $f(x_1, \ldots, x_n)$ obtained by Tseitin's transformation?

Solution 2.

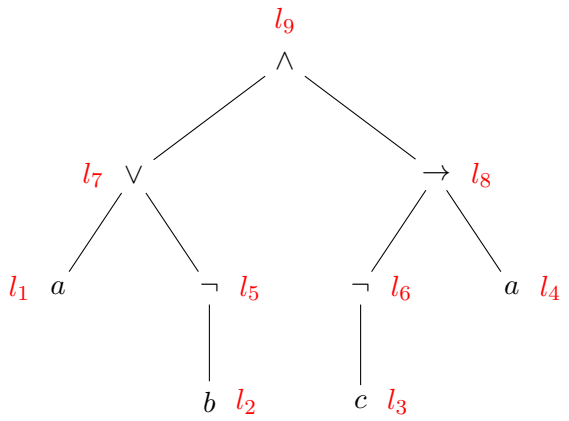


Figure 1: Formula tree for ψ and assigned labels in red.

2.1 The formula tree and the assigned labels for ψ are given in Figure 1.

The resulting equivalences are:

$$\begin{array}{lll} \ell_1 \leftrightarrow a & \ell_2 \leftrightarrow b & \ell_3 \leftrightarrow c \\ \ell_4 \leftrightarrow a & \ell_5 \leftrightarrow \neg\ell_2 & \ell_6 \leftrightarrow \neg\ell_3 \\ \ell_7 \leftrightarrow (\ell_1 \vee \ell_5) & \ell_8 \leftrightarrow (\ell_6 \rightarrow \ell_4) & \ell_9 \leftrightarrow (\ell_7 \wedge \ell_8) \end{array}$$

Transforming those to CNF yields:

$$\begin{array}{lll} \neg\ell_1 \vee a & \ell_1 \vee \neg a & \\ \neg\ell_2 \vee b & \ell_2 \vee \neg b & \\ \neg\ell_3 \vee c & \ell_3 \vee \neg c & \\ \neg\ell_4 \vee a & \ell_4 \vee \neg a & \\ \neg\ell_5 \vee \neg\ell_2 & \ell_5 \vee \ell_2 & \\ \neg\ell_6 \vee \neg\ell_3 & \ell_6 \vee \ell_3 & \\ \neg\ell_7 \vee \ell_1 \vee \ell_5 & \ell_7 \vee \neg\ell_1 & \ell_7 \vee \neg\ell_5 \\ \neg\ell_8 \vee \neg\ell_6 \vee \ell_4 & \ell_8 \vee \ell_6 & \ell_8 \vee \neg\ell_4 \\ \neg\ell_9 \vee \ell_7 & \neg\ell_9 \vee \ell_8 & \ell_9 \vee \neg\ell_7 \vee \neg\ell_8 \end{array}$$

If we add the single clause ℓ_9 to the above set of clauses, then the resulting set of clauses is sat-equivalent to ψ.

2.2 A CNF encoding can be obtained by successively transforming the XOR operators to CNF based on the following clause pattern. Let l be the label of $x \oplus y$, i.e., $l \leftrightarrow x \oplus y$, then the corresponding CNF encoding is: $(\neg l \vee x \vee y) \wedge (\neg l \vee \neg x \vee \neg y) \wedge (l \vee \neg x \vee y) \wedge (l \vee x \vee \neg y)$.

2.3 We get two clauses from encoding equivalences of the form $l_i \leftrightarrow x_i$, four clauses from encoding an XOR operator, and one clause as the final clause representing the output of the function, i.e., $1 + 2 \times n + 4 \times (n - 1) = 6n - 3$.

Exercise 3. *Implication Graphs (with sample solution)*
This exercise illustrates conflict-driven clause learning (CDCL), the generation of asserting learned clauses and conflict-driven backtracking. The sample solution provided below is intended to illustrate the basic concepts as a preparation for Exercise 4.

Let $\mathcal{C}$ be a clause set consisting of the following clauses:

C_1:	$(\neg a \vee b)$	C_2:	$(\neg a \vee \neg b \vee c)$	C_3:	$(a \vee b)$
C_4:	$(\neg f \vee \neg b \vee \neg g)$	C_5:	$(g \vee \neg e)$	C_6:	$(g \vee d)$
C_7:	$(c \vee e \vee \neg d)$	C_8:	$(\neg a \vee c)$		

3.1 Draw an implication graph for $\mathcal{C}$. Use the decision $c = 0@1$, and $f = 1@2$ until you reach a conflict.

3.2 Determine all UIPs in the implication graph, find the first UIP and use resolution to learn an asserting clause corresponding to the first UIP.

3.3 Add the learned clause, apply conflict-driven backtracking and draw the resulting implication graph.

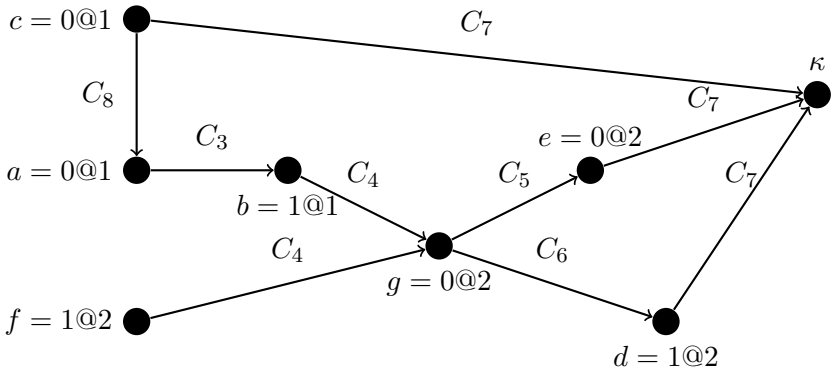


Figure 2: Implication Graph for $\mathcal{C}$ with decisions $c = 0@1$ and $f = 1@2$.

Solution 3.
3.1 The resulting implication graph is given in Figure 2.

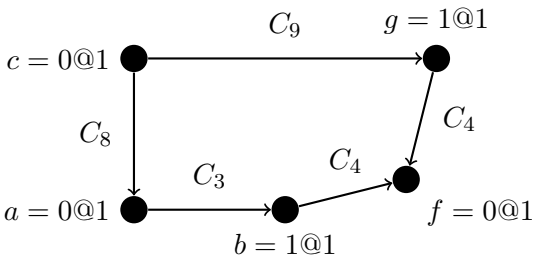


Figure 3: Implication Graph for $\mathcal{C}$ with learned clause C_9 after conflict-driven backtracking and BCP.

3.2 UIPs (nodes which appear on every path from the current decision to the conflict κ) are the nodes $f = 1@2$ and $g = 0@2$ where the latter is the first UIP (closest to the conflict).

We resolve C_7, C_5, and C_6 and obtain:

$$\begin{aligned} R_1 := res(C_7, C_5, e) &= (c \vee g \vee \neg d) \\ R_2 := res(R_1, C_6, d) &= (c \vee g \vee g) \\ fac(R_2) &= (c \vee g) \end{aligned}$$

Therefore the learned asserting clause according to the first UIP scheme is $C_9 \colon (c \vee g)$. This clause is asserting because it contains exactly one literal, i.e., g, whose variable is assigned at the highest decision level.

3.3 For conflict-driven backtracking, we backtrack to the second highest DL in the learned clause, i.e., we backtrack to $DL = 1$. For this kind of backtracking, we keep all assignments at $DL = 1$ (i.e., we keep the decision and all implications at $DL = 1$) but delete all others with $DL > 1$. After BCP the resulting implication graph is shown in Figure 3.

Exercise 4. *Implication Graphs (recommended)*

Let $\mathcal{C}$ be a clause set consisting of the following clauses:

$C_1 \colon$	(x_1)	$C_2 \colon$	$(\neg x_4 \vee x_5)$	$C_3 \colon$	$(\neg x_4 \vee x_6)$
$C_4 \colon$	$(\neg x_1 \vee \neg x_5 \vee \neg x_6 \vee x_7)$	$C_5 \colon$	$(\neg x_2 \vee \neg x_7 \vee x_8)$	$C_6 \colon$	$(\neg x_7 \vee \neg x_8)$
$C_7 \colon$	$(x_5 \vee x_7)$	$C_8 \colon$	$(\neg x_1 \vee x_7 \vee x_8)$	$C_9 \colon$	$(\neg x_5 \vee x_7 \vee \neg x_8)$
$C_{10} \colon$	$(x_2 \vee \neg x_7 \vee x_8)$				

Apply conflict-driven clause learning (CDCL) to solve $\mathcal{C}$ by assigning x_2 true at decision level 1 (i.e., $x_2 = 1@1$) and assigning x_4 true at decision level 2 (i.e., $x_4 = 1@2$). Each time when reaching a conflict, carry out the following steps:

- Draw the implication graph and mark all unique implication points (UIPs).

- Learn an asserting clause corresponding to the *first UIP* and illustrate the resolution steps necessary to derive the learned clause.
- Backtrack to the asserting level according to the learned clause and continue BCP.

Repeat the above steps until CDCL terminates. Is $\mathcal{C}$ satisfiable or unsatisfiable?

Hints:

- Note that $\mathcal{C}$ contains a unit clause which must be properly handled before assigning any variables as decisions.
- Recall that all learned clauses are added to $\mathcal{C}$ and hence may also become unit or conflicting when the run of CDCL proceeds after backtracking.
- During BCP, different orderings in which variables are assigned as implications may result in different implication graphs. For this exercise, it is sufficient to select one ordering and draw the corresponding implication graph.

Solution 4.
We sketch the solution.

- Unit literals are assigned at decision level 0. A conflict is generated at decision level 2. Node x_7 is the 1-UIP, the first learned clause is $c_{11} = (\neg x_2 \vee \neg x_7)$ by resolving the clauses c_5 and c_6. The learned clause c_{11} is asserting at decision level 1 and hence we backtrack to level 1, where the learned clause c_{11} becomes unit.
- A conflict is generated at decision level 1. Node x_7 is the 1-UIP, the second learned clause is $c_{12} = (\neg x_1 \vee x_7)$ by resolving the clauses C_8, C_9 and then c_7. The learned clause c_{12} is asserting at decision level 0 and hence we backtrack to level 0, where the learned clause c_{12} becomes unit.
- A conflict is generated at decision level 0. Since there are no decisions that could be flipped, we can stop and conclude unsatisfiability. (Optionally, we can derive the empty clause by resolving all antecedent clauses of unit literals at decision level 0, in reverse assignment ordering, starting from the conflict.)

Exercise 5. *Sparse Method (recommended)*
Apply the Sparse Method including preprocessing on the formula φ below to obtain a propositional formula. Note that φ is not yet in NNF (Negation Normal Form).

$$(x_1 = x_2 \to x_2 = x_3) \wedge \left[\neg(x_2 = x_4 \vee x_3 \neq x_4 \vee x_4 \neq x_5) \vee (x_6 \neq x_5 \wedge x_6 = x_7 \wedge x_7 = x_3)\right]$$

Solution 5.
In the first step, we transform φ into NNF. We substitute $\rightarrow$ and apply De Morgan to obtain φ^E, which now is in NNF:

$$(x_1 \neq x_2 \vee x_2 = x_3) \wedge \big[(x_2 \neq x_4 \wedge x_3 = x_4 \wedge x_4 = x_5) \vee (x_6 \neq x_5 \wedge x_6 = x_7 \wedge x_7 = x_3)\big]$$

Then, we draw the equality graph $G^E(\varphi^E)$ of φ^E, given in Figure 4. Dashed lines represent equality edges while solid lines represent disequality edges.

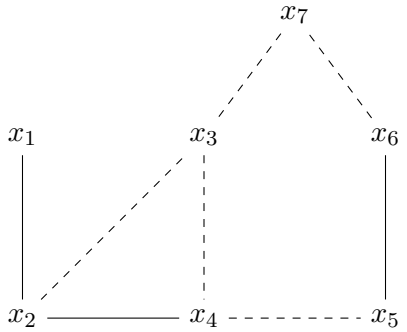


Figure 4: Equality graph $G^E(\varphi^E)$, dashed lines represent equality, solid lines disequality.

The edge (x_1, x_2) is not part of a simple contradictory cycle, therefore we set $(x_1 \neq x_2)$ to true and obtain φ_2^E:

$$(\mathit{true} \vee x_2 = x_3) \wedge \big[(x_2 \neq x_4 \wedge x_3 = x_4 \wedge x_4 = x_5) \vee (x_6 \neq x_5 \wedge x_6 = x_7 \wedge x_7 = x_3)\big]$$

Propositional simplification yields φ_3^E:

$$\big[(x_2 \neq x_4 \wedge x_3 = x_4 \wedge x_4 = x_5) \vee (x_6 \neq x_5 \wedge x_6 = x_7 \wedge x_7 = x_3)\big]$$

The equality graph $G^E(\varphi_3^E)$ then is as shown in Figure 5.

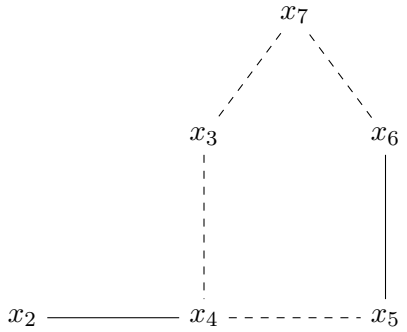


Figure 5: Equality graph $G^E(\varphi_3^E)$, dashed lines represent equality, solid lines disequality.

Edge (x_2, x_4) now is not in a simple contradictory cycle, therefore we set $(x_2 \neq x_4)$ to true and apply propositional simplification to obtain φ_4^E:

$$\big[(x_3 = x_4 \wedge x_4 = x_5) \vee (x_6 \neq x_5 \wedge x_6 = x_7 \wedge x_7 = x_3)\big]$$

The equality graph $G^E(\varphi_4^E)$ is given in Figure 6.

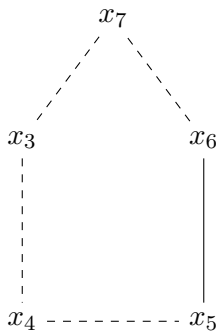


Figure 6: Equality graph $G^E(\varphi_4^E)$, dashed lines represent equality, solid lines disequality.

All edges of $G^E(\varphi_4^E)$ are part of a simple contradictory cycle, so we stop with preprocessing and build the propositional skeleton $e(\varphi_4^E)$:

$$(e_{3,4} \wedge e_{4,5}) \vee (\neg e_{5,6} \wedge e_{6,7} \wedge e_{3,7})$$

For transitivity constraints B_t we make the nonpolar equality graph $G^E_{NP}(\varphi_4^E)$ chordal as shown in Figure 7. Observe that edges (x_4, x_7) and (x_5, x_7) are introduced.

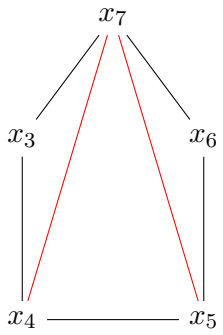


Figure 7: Nonpolar equality graph $G^E_{NP}(\varphi_4^E)$, made chordal by additional edges (in red).

The according transitivity constraints B_t are then:

$$(e_{3,4} \wedge e_{4,7} \rightarrow e_{3,7}) \wedge (e_{4,7} \wedge e_{3,7} \rightarrow e_{3,4}) \wedge (e_{3,7} \wedge e_{3,4} \rightarrow e_{4,7}) \wedge$$
$$(e_{4,5} \wedge e_{5,7} \rightarrow e_{4,7}) \wedge (e_{5,7} \wedge e_{4,7} \rightarrow e_{4,5}) \wedge (e_{4,7} \wedge e_{4,5} \rightarrow e_{5,7}) \wedge$$
$$(e_{5,6} \wedge e_{6,7} \rightarrow e_{5,7}) \wedge (e_{6,7} \wedge e_{5,7} \rightarrow e_{5,6}) \wedge (e_{5,7} \wedge e_{5,6} \rightarrow e_{6,7})$$

The resulting formula in propositional logic then is $e(\varphi_4^E) \wedge B_t$.

Exercise 6. *Ackermann's Reduction (recommended)*
Apply Ackermann's reduction on the following EUF-formula φ to obtain an E-formula:

$$F(F(x_1)) \neq F(x_1) \wedge G(x_1, x_2) = F(x_2) \wedge F(G(x_2, F(x_2))) \neq F(F(x_1))$$

Solution 6.
We first number the instances of the UFs inwards-to-outwards, left-to-right:

$$F_2(F_1(x_1)) \neq F_1(x_1) \wedge G_1(x_1, x_2) = F_3(x_2) \wedge F_4(G_2(x_2, F_3(x_2))) \neq F_2(F_1(x_1))$$

This already gives $\mathcal{T}$ for the numbered instances. For example:

$$\begin{aligned}
\mathcal{T}(F_1(x_1)) &= f_1 \\
\mathcal{T}(F_2(F_1(x_1))) &= f_2 \\
\mathcal{T}(F_3(x_2)) &= f_3 \\
\mathcal{T}(F_4(G_2(x_2, F_3(x_2)))) &= f_4 \\
\mathcal{T}(G_1(x_1, x_2)) &= g_1 \\
\mathcal{T}(G_2(x_2, F_3(x_2))) &= g_2
\end{aligned}$$

So $flat^E := f_2 \neq f_1 \wedge g_1 = f_3 \wedge f_4 \neq f_2$.
Based on $\mathcal{T}$ we construct $FC^E :=$

$$\begin{aligned}
&(x_1 = f_1 \rightarrow f_1 = f_2) \wedge \\
&(x_1 = x_2 \rightarrow f_1 = f_3) \wedge \\
&(x_1 = g_2 \rightarrow f_1 = f_4) \wedge \\
&(f_1 = x_2 \rightarrow f_2 = f_3) \wedge \\
&(f_1 = g_2 \rightarrow f_2 = f_4) \wedge \\
&(x_2 = g_2 \rightarrow f_3 = f_4) \wedge \\
&((x_1 = x_2 \wedge x_2 = f_3) \rightarrow g_1 = g_2)
\end{aligned}$$

Finally $\varphi^E := FC^E \rightarrow flat^E$.

Exercise 7. *Correctness of Resolution*
Let F be a propositional formula in CNF and C be the resolvent of two clauses $C_1 \in F$ and $C_2 \in F$. Prove that $F \equiv F \wedge C$.

Solution 7.
We have $F \equiv F \wedge C$ if and only if $Models(F) = Models(F \wedge C)$, where $Models(\psi)$ denotes the class of all models of the propositional formula ψ.

Further, we have $Models(F) = Models(F \wedge C)$ if and only if $Models(F) \subseteq Models(F \wedge C)$ and $Models(F) \supseteq Models(F \wedge C)$. We prove $\subseteq$ and $\supseteq$ separately.

- Case $\mathit{Models}(F) \supseteq \mathit{Models}(F \wedge C)$:

  Let $I \in \mathit{Models}(F \wedge C)$ be an arbitrary model of $F \wedge C$. Hence $I \models F \wedge C$, i.e., $I(F \wedge C) = 1$ and in particular $I(F) = 1$ and $I(C) = 1$ due to the semantics of $\wedge$. Since $I(F) = 1$, $I \in \mathit{Models}(F)$, and hence the $\supseteq$-relation holds.

- Case $\mathit{Models}(F) \subseteq \mathit{Models}(F \wedge C)$:

  Let $I \in \mathit{Models}(F)$ be an arbitrary model of F. Then $I(F) = 1$ and in particular $I(C_1) = 1$ and $I(C_2) = 1$ since F is a CNF and I must satisfy every clause in F. Note that C is obtained from C_1 and C_2 by resolution. Let p be the pivot variable of that resolution step. Without loss of generality, assume that $p \in C_1$ and $\neg p \in C_2$.

  - Subcase: assume $I(p) = 1$. Since $I(C_2) = 1$, there exists a literal $l \in C_2$ such that $l \neq \neg p$ and $I(l) = 1$. Then l appears also in the resolvent C, and hence $I(C) = 1$, further $I(F \wedge C) = 1$ and $I \in \mathit{Models}(F \wedge C)$.
  - Subcase: assume $I(p) = 0$ The proof is symmetric to the subcase above.

  Hence the $\subseteq$-relation holds.

Exercise 8. *Equivalent Replacement Theorem*
Recall the notation $\varphi[\psi]$ from the lecture. Prove the following statement:

> Let φ, ψ_1, and ψ_2 be propositional formulas. If $\psi_1 \equiv \psi_2$, then $\varphi[\psi_1] \equiv \varphi[\psi_2]$.

Solution 8.
A proof of the equivalent replacement lemma together with a proof of the equivalent replacement theorem can be found in Theorem 2.5.1 and Theorem 2.5.2 in M. Fitting's book on *First-order Logic and Automated Deduction*. We give a different proof here.

Let us first discuss the (degenerated) case when φ has no occurrence of ψ_1. Then $\varphi[\psi_1] = \varphi$ as well as $\varphi[\psi_2] = \varphi$ and thus $\varphi[\psi_1] \equiv \varphi[\psi_2]$ holds trivially. In the following we assume that φ has at least one occurrence of ψ_1

We define the *logical complexity*, $lc(\cdot)$, of a formula.

$$lc(\varphi) = \begin{cases} 0 & \text{if } \varphi \text{ is atomic;} \\ lc(\varphi_1) + 1 & \text{if } \varphi \text{ is } \neg\varphi_1; \\ lc(\varphi_1) + lc(\varphi_2) + 1 & \text{if } \varphi \text{ is } \varphi_1 \circ \varphi_2 \end{cases}$$

where $\circ \in \{\wedge, \vee, \rightarrow, \leftrightarrow, \oplus\}$.

Let $P(n)$ denote the following statement:

> *For all* $\varphi[\psi_1]$ *with* $lc(\varphi[\psi_1]) = n$ *and all formulas* ψ_2, $\psi_1 \equiv \psi_2 \rightarrow \varphi[\psi_1] \equiv \varphi[\psi_2]$.

We prove $P(n)$ for all n by strong mathematical induction on n.

Base case: $n = 0$. Then $lc(\varphi[\psi_1]) = 0$, $\varphi[\psi_1]$ is an atom (say p) and ψ_2 is an arbitrary formula. Assume $\psi_1 \equiv \psi_2$. Then $p = \varphi[\psi_1] = \psi_1 \equiv \psi_2 = \varphi[\psi_2]$ and therefore $\varphi[\psi_1] \equiv \varphi[\psi_2]$. Then it holds that $\psi_1 \equiv \psi_2 \rightarrow \varphi[\psi_1] \equiv \varphi[\psi_2]$ for all formulas $\varphi[\psi_1]$ with $lc(\varphi[\psi_1]) = 0$ and all formulas ψ_2.

Induction hypothesis (IH): Let $n \geq 0$ and suppose that $P(0), P(1), \ldots, P(n)$ hold.

Induction step: We have to show that $P(n+1)$ holds.

Let us choose an arbitrary formula $\varphi[\psi_1]$ with $lc(\varphi[\psi_1]) = n+1$ and an arbitrary formula ψ_2. If $\psi_1 \not\equiv \psi_2$ then we are done because $\psi_1 \equiv \psi_2 \rightarrow \varphi[\psi_1] \equiv \varphi[\psi_2]$ holds. Since $\varphi[\psi_1]$ with $lc(\varphi[\psi_1]) = n+1$ as well as ψ_2 has been arbitrarily chosen, $P(n+1)$ holds.

So assume $\psi_1 \equiv \psi_2$.

Case 1: $\varphi[\psi_1] = \psi_1$. Then $\varphi[\psi_2] = \psi_2$ and $\varphi[\psi_1] \equiv \varphi[\psi_2]$ follows.

Let ψ_1 be a proper subformula of $\varphi[\psi_1]$.

Case 2: Assume $\varphi[\psi_1]$ is of the form $\neg\varphi_1[\psi_1]$. Then by (IH), $\varphi_1[\psi_1] \equiv \varphi_1[\psi_2]$ and for any assignment I, $I \models \varphi_1[\psi_1]$ iff $I \models \varphi_1[\psi_2]$. But then $I \models \neg\varphi_1[\psi_1]$ iff $I \models \neg\varphi_1[\psi_2]$ by the semantics of $\neg$ and $\varphi[\psi_1] \equiv \varphi[\psi_2]$ follows.

Case 3: $\varphi[\psi_1]$ is of the form $\varphi_1[\psi_1] \circ \varphi_2[\psi_1]$ for $\circ \in \{\wedge, \vee, \rightarrow, \leftrightarrow, \oplus\}$. By (IH), we have (i) $\varphi_1[\psi_1] \equiv \varphi_1[\psi_2]$ and (ii) $\varphi_2[\psi_1] \equiv \varphi_2[\psi_2]$. Recall that all connectives are truth functions and that propositional logic is truth-functional, i.e., the truth value of a formula is determined by the truth value(s) of its immediate subformula(s) and the main connective. By functional consistency, if we have $s_1 = t_1$ and $s_2 = t_2$, then we have $f(s_1, s_2) = f(t_1, t_2)$. Since (i) and (ii) hold, we have for each assignment I that $I(\varphi_1[\psi_1]) = I(\varphi_1[\psi_2])$ and $I(\varphi_2[\psi_1]) = I(\varphi_2[\psi_2])$. But then $I(\varphi_1[\psi_1] \circ \varphi_2[\psi_1]) = I(\varphi_1[\psi_2] \circ \varphi_2[\psi_2])$ and therefore $\varphi[\psi_1] \equiv \varphi[\psi_2]$ follows.

Exercise 9. *Negation Normal Form*

9.1 Transform the following formula into negation normal form (NNF):

$$\big(a \wedge (b \rightarrow c)\big) \vee \neg(a \vee d)$$

9.2 Let φ be a propositional formula in negation normal form (NNF).

Prove: If φ contains only pure literals, then φ is satisfiable.

Hint 1: A literal is pure in a formula, if it occurs only positively or negatively in that formula, but not both.

Hint 2: The number n of binary connectives occurring in φ may be arbitrary high! Which proof principle is suitable for such cases?

9.3 Let φ be as in task 9.2, is it always the case that φ is valid? Argue formally.

Solution 9.

9.1 We first replace $\phi_1 \to \phi_2$ by the equivalent formula $\neg\phi_1 \vee \phi_2$. We then push negation down using De Morgan's laws: $\neg(\phi_1 \wedge \phi_2)$ becomes $(\neg\phi_1 \vee \neg\phi_2)$ and $\neg(\phi_1 \vee \phi_2)$ becomes $(\neg\phi_1 \wedge \neg\phi_2)$. If we encounter double negation anywhere, we remove it, since $\neg\neg\phi$ is equivalent to ϕ.

The resulting NNF of the above formula is $\big(a \wedge (\neg b \vee c)\big) \vee (\neg a \wedge \neg d)$.

9.2 Since φ is in NNF, the only logical connectives are $\wedge, \vee$, and $\neg$ where the latter only occurs directly in front of a propositional variable. Thus the only binary connectives are $\wedge$ and $\vee$.

Let $\mathcal{BV}$ be the set of propositional variables that occur in φ. We define I as follows: if $b \in \mathcal{BV}$ occurs positively in φ, then $I(b) = 1$, else $I(b) = 0$. Since all literals in φ are pure, this definition of I does not contradict itself.

To show that φ is satisfiable, we use induction on the number n of binary connectives in φ, i.e., we show that every subformula of φ is true in I.

Base case $n = 0$: ψ is a literal, hence $I \models \psi$.

Induction hypothesis (IH): $I \models \psi$ for every subformula ψ of φ with less than or equal to n binary connectives.

Induction step: $n \mapsto n + 1$: Let ψ contain $n + 1$ binary connectives, we distinct on the topmost binary connective of ψ:

- $\wedge$: Then $\psi = \phi_1 \wedge \phi_2$ where ϕ_1 and ϕ_2 both contain less than or equal to n binary connectives. By IH it thus holds that $I \models \phi_1$ and $I \models \phi_2$, hence $I \models \psi$ by the semantics of $\wedge$.
- $\vee$: Then $\psi = \phi_1 \vee \phi_2$ and again by IH it holds that $I \models \phi_1$, hence $I \models \psi$ by the semantics of $\vee$.
- Nothing else is possible since φ is in NNF.

Therefore $I \models \psi$ for every subformula ψ of φ, specifically $I \models \varphi$ since φ is also a subformula of φ.

9.3 No, φ need not be valid. Consider for example $\varphi = p$ and $I(p) = 0$. Then $I \not\models \varphi$, i.e., φ is in NNF, it contains only pure literals, and φ is not valid.

Exercise 10. *First-Order Theories*

In the lecture, we discussed reasoning under different theories. Here we are concerned with LISP-like lists and the theory $\mathcal{T}^E_{cons} = \mathcal{T}_{cons} \cup \mathcal{T}_E$. In a verification attempt of some program, we have to prove the following:

> *For non-atomic lists ℓ_1, ℓ_2, if the "car" of both lists are equal and the "cdr" of both lists are equal, then ℓ_1 is equal to ℓ_2.*

We formalize the above statement as follows:

$$\varphi\colon \quad \big[\neg atom(\ell_1) \wedge \neg atom(\ell_2) \wedge car(\ell_1) \doteq car(\ell_2) \wedge cdr(\ell_1) \doteq cdr(\ell_2)\big] \;\to\; \ell_1 \doteq \ell_2$$

Prove the statement $\mathcal{T}^E_{cons}$-valid, i.e., show that $\mathcal{T}^E_{cons} \models \varphi$ holds. Use the semantics proof method introduced in the lecture.

Hint: Besides the equality axioms reflexivity, symmetry and transitivity, the following axioms from $\mathcal{T}^E_{cons}$ are sufficient for a proof:

(1) Substitution axioms (functional congruence) for *cons*:

$$\forall x_1 \forall x_2 \forall y_1 \forall y_2 \,.\, [(x_1 \doteq x_2 \wedge y_1 \doteq y_2) \rightarrow cons(x_1, y_1) \doteq cons(x_2, y_2)]$$

(2) Construction:

$$\forall x \,.\, [\neg atom(x) \rightarrow cons(car(x), cdr(x)) \doteq x]$$

Solution 10.
The proof is by contradiction. Suppose there exists a $\mathcal{T}^E_{cons}$-interpretation $\mathcal{I}$ with $\mathcal{I} \not\models \varphi$.

1.	$\mathcal{I} \not\models \varphi$	assumption
2.	$\mathcal{I} \models car(\ell_1) \doteq car(\ell_2)$	1., $\rightarrow$, $\wedge$
3.	$\mathcal{I} \models cdr(\ell_1) \doteq cdr(\ell_2)$	1., $\rightarrow$, $\wedge$
4.	$\mathcal{I} \models \neg atom(\ell_1)$	1., $\rightarrow$, $\wedge$
5.	$\mathcal{I} \models \neg atom(\ell_2)$	1., $\rightarrow$, $\wedge$
6.	$\mathcal{I} \not\models \ell_1 \doteq \ell_2$	1., $\rightarrow$
7.	$\mathcal{I} \models cons(car(\ell_1), cdr(\ell_1)) \doteq cons(car(\ell_2), cdr(\ell_2))$	2., 3., functional congruence
8.	$\mathcal{I} \models cons(car(\ell_1), cdr(\ell_1)) \doteq \ell_1$	4., construction
9.	$\mathcal{I} \models cons(car(\ell_2), cdr(\ell_2)) \doteq \ell_2$	5., construction
10.	$\mathcal{I} \models \ell_1 \doteq \ell_2$	7., 8., 9., symmetry + transitivity
11.	$\mathcal{I} \models \bot$	6., 10.

The assumption is false: φ is therefore $\mathcal{T}^E_{cons}$-valid.

Exercise 11. *Theory of Lists: Stepwise Induction Proofs (recommended)*
Consider the theory $\mathcal{T}^E_{cons}$ of lists from the previous exercise (see also the lecture slides). *Stepwise induction* in $\mathcal{T}^E_{cons}$ is defined by the following schema

$$(\forall u.\ atom(u) \rightarrow F[u]) \wedge (\forall u, v.\ F[v] \rightarrow F[cons(u, v)]) \rightarrow \forall x.\ F[x]$$

where $F[x]$ is a $\mathcal{T}^E_{cons}$-formula with the only free variable x. To prove $\forall x.\ F[x]$, that is, to prove that $F[x]$ is $\mathcal{T}^E_{cons}$-valid for all lists x, it is sufficient to apply the *stepwise induction principle* for lists based on the following steps:

1. For the *base case*, prove that $F[u]$ is $\mathcal{T}^E_{cons}$-valid for an arbitrary atom u.
2. For the *inductive step*, assume as *inductive hypothesis* that $F[v]$ is $\mathcal{T}^E_{cons}$-valid for some arbitrary list v. Then prove that $F[cons(u, v)]$ is $\mathcal{T}^E_{cons}$-valid for some arbitrary list u.

Now consider the theory $\mathcal{T}^{E,+}_{cons}$ obtained by augmenting $\mathcal{T}^E_{cons}$ with the following axioms.

- $\forall u, v.\ atom(u) \rightarrow concat(u, v) = cons(u, v)$ (concat atom)
- $\forall u, v, x.\ concat(cons(u, v), x) = cons(u, concat(v, x))$ (concat list)
- $\forall u.\ atom(u) \rightarrow rvs(u) = u$ (reverse atom)
- $\forall x, y.\ rvs(concat(x, y)) = concat(rvs(y), rvs(x))$ (reverse list)
- $\forall u.\ atom(u) \rightarrow \mathit{flat}(u)$ (flat atom)
- $\forall u, v.\ \mathit{flat}(cons(u, v)) \leftrightarrow atom(u) \wedge \mathit{flat}(v)$ (flat list)

Function *concat* concatenates two lists, function *rvs* reverses a list, and *flat* is a predicate which evaluates to true if and only if every element of a list is an atom.

Prove that the formulas

$$\forall u.\ \mathit{flat}(u) \rightarrow rvs(rvs(u)) = u \tag{L1}$$
$$\forall u, v.\ \mathit{flat}(u) \wedge \mathit{flat}(v) \rightarrow \mathit{flat}(concat(u, v)) \tag{L2}$$

are $\mathcal{T}^{E,+}_{cons}$-valid by applying the stepwise induction principle.

Solution 11.

A detailed proof of (L1) can be found in `extra-sheet3-1.pdf` which is loctaed in the `tuwel` resource directory for Block 2.

We prove (L2) here by stepwise induction over list u. Within the proof, we use fc for functional consistency and pc for predicate consistency. In the base case as well as in the step case, we use the semantic argument method known from the lecture.

Let $F[u]$ denote

$$\forall v.\ \mathit{flat}(u) \wedge \mathit{flat}(v) \rightarrow \mathit{flat}(concat(u, v)).$$

Base case. u is an atom, i.e. $atom(u)$ holds, which is an extra assumption available in the proof process. We have then to prove the formula

$$\psi\colon\ atom(u) \wedge \mathit{flat}(u) \wedge \mathit{flat}(v) \rightarrow \mathit{flat}(concat(u, v)).$$

The proof is by contradiction. Suppose there exists a $\mathcal{T}^{+}_{cons}$-interpretation structure $\mathcal{I}$ with $\mathcal{I} \not\models \psi$.

1.	$\mathcal{I} \not\models \psi$	assumption
2.	$\mathcal{I} \models atom(u) \wedge \mathit{flat}(u) \wedge \mathit{flat}(v)$	1., semantics of $\rightarrow$
3.	$\mathcal{I} \not\models \mathit{flat}(concat(u, v))$	1., semantics of $\rightarrow$
4.	$\mathcal{I} \models atom(u)$	2., semantics of $\wedge$
5.	$\mathcal{I} \models \mathit{flat}(v)$	2., semantics of $\wedge$
6.	$\mathcal{I} \models atom(u) \wedge \mathit{flat}(v)$	4., 5., semantics of $\wedge$
7.	$\mathcal{I} \models \mathit{flat}(cons(u, v))$	6., flat list
8.	$\mathcal{I} \models concat(u, v) \doteq cons(u, v)$	4., concat atom
9.	$\mathcal{I} \models \mathit{flat}(concat(u, v)) \leftrightarrow \mathit{flat}(cons(u, v))$	8., pc *flat*
10.	$\mathcal{I} \models \mathit{flat}(concat(u, v))$	7., 9., logical equivalence
11.	$\mathcal{I} \models \bot$	10., 3., contradiction

Hence $F[u]$ holds for an arbitrary atom u.

Induction hypothesis. Assume $F[v]$ is true for some list v, i.e.,

$$\forall w.\ \mathit{flat}(v) \land \mathit{flat}(w) \rightarrow \mathit{flat}(\mathit{concat}(v, w))$$

holds.

Induction step. We show that for arbitrary lists u, $F[\mathit{cons}(u, v)]$ is true. Therefore, we prove that the following formula holds.

$$\varphi\colon \forall w.\ \mathit{flat}(\mathit{cons}(u, v)) \land \mathit{flat}(w) \rightarrow \mathit{flat}(\mathit{concat}(\mathit{cons}(u, v), w))$$

Let us first fix an arbitrary value w for the universal quantifier (this could also be done in the semantic argument proof, but we do it already here). The resulting formula is φ'. The proof is then by contradiction. Suppose there exists a $\mathcal{T}^{+}_{\mathit{cons}}$-interpretation structure $\mathcal{I}$ with $\mathcal{I} \not\models \varphi'$.

1.	$\mathcal{I} \not\models \varphi'$	assumption
2.	$\mathcal{I} \models \mathit{flat}(\mathit{cons}(u, v))$	1., semantics of $\rightarrow$ and $\land$
3.	$\mathcal{I} \models \mathit{flat}(w)$	1., semantics of $\rightarrow$ and $\land$
4.	$\mathcal{I} \not\models \mathit{flat}(\mathit{concat}(\mathit{cons}(u, v), w))$	1., semantics of $\rightarrow$
5.	$\mathcal{I} \models \mathit{atom}(u) \land \mathit{flat}(v))$	2., flat list: $\mathit{flat}(\mathit{cons}(u, v)) \leftrightarrow \mathit{atom}(u) \land \mathit{flat}(v)$
6.	$\mathcal{I} \models \mathit{atom}(u)$	5., semantics of $\land$
7.	$\mathcal{I} \models \mathit{flat}(v)$	5., semantics of $\land$
8.	$\mathcal{I} \models \mathit{flat}(\mathit{concat}(v, w))$	7., 3., induction hypothesis
9.	$\mathcal{I} \models \mathit{flat}(\mathit{concat}(\mathit{cons}(u, v), w)) \leftrightarrow \mathit{flat}(\mathit{cons}(u, \mathit{concat}(v, w)))$	concat list, pc *flat*
10.	$\mathcal{I} \not\models \mathit{flat}(\mathit{cons}(u, \mathit{concat}(v, w)))$	9., 4., logical equivalence
11.	$\mathcal{I} \not\models \mathit{atom}(u) \land \mathit{flat}(\mathit{concat}(v, w))$	10., flat list
12.	$\mathcal{I} \models \mathit{atom}(u) \land \mathit{flat}(\mathit{concat}(v, w))$	6., 8., semantics of $\land$
13.	$\mathcal{I} \models \bot$	11., 12., contradiction

The assumption is false: φ' and therefore φ is $\mathcal{T}^{+}_{\mathit{cons}}$-valid! This concludes the step case and the induction proof.

Exercise 12. *Theory of Arrays: Extensionality (recommended)*
Consider the theory $\mathcal{T}^{=}_{A}$ of arrays with extensionality as presented in the lecture. Prove that the formula φ

$$a[i] \doteq e \rightarrow a\langle i \lhd e\rangle \doteq a$$

is $\mathcal{T}^{=}_{A}$-valid.

Solution 12.
The proof is by contradiction. Suppose there exists a $\mathcal{T}_A^{=}$-interpretation $\mathcal{I}$ with $\mathcal{I} \not\models \varphi$.

1.	$\mathcal{I} \not\models \varphi$	assumption
2.	$\mathcal{I} \models a[i] \doteq e$	1., $\rightarrow$
3.	$\mathcal{I} \not\models a\langle i \lhd e\rangle \doteq a$	1., $\rightarrow$
4.	$\mathcal{I} \models a\langle i \lhd e\rangle \not\doteq a$	3., $\neg$
5.	$\mathcal{I} \models \neg(\forall j.a\langle i \lhd e\rangle[j] \doteq a[j])$	4., (extensionality)
5.	$\mathcal{I} \not\models \forall j.a\langle i \lhd e\rangle[j] \doteq a[j]$	5., $\neg$

The rest of the proof has been presented in the third slide set on first-order logic and theories.

Exercise 13. *Tseitin Transformation (recommended)*
We consider a restriction of the Tseitin transformation where the input formula is only composed of propositional variables, negation, and conjunction. Let ψ be a propositional formula and let $\delta(\psi)$ be the set of clauses resulting from Tseitin's transformation on ψ. Prove that the following holds:

$$\text{If } \psi \text{ is satisfiable then } \delta(\psi) \text{ is satisfiable.} \tag{1}$$

You only need to prove this for the connectives $\wedge$ and $\neg$. Use the following clause schemes, which introduce a new label for every Boolean variable.

a is atomic	$(\neg\ell_a \vee a)$	$(\ell_a \vee \neg a)$	
ϕ is $\phi_1 \wedge \phi_2$	$(\neg\ell_\phi \vee \ell_{\phi_1})$	$(\neg\ell_\phi \vee \ell_{\phi_2})$	$(\ell_\phi \vee \neg\ell_{\phi_1} \vee \neg\ell_{\phi_2})$
ϕ is $\neg\phi_1$	$(\neg\ell_\phi \vee \neg\ell_{\phi_1})$	$(\ell_\phi \vee \ell_{\phi_1})$	

What can you say about (1) for unrestricted formulas ψ?

Solution 13.
Let $\hat{\delta}(\psi)$ be the set of all clauses from the labeling of ψ and the additional clause (ℓ_ψ), i.e., $\delta(\psi) = \hat{\delta}(\psi) \cup \{(\ell_\psi)\}$. We have to show the following.

$$\text{If } \psi \text{ is satisfiable then } \delta(\psi) \text{ is satisfiable.}$$

In other words: If there exists $I \in Mod(\psi)$ then there exists a model $I' \in Mod(\delta(\psi))$, that is for every $C \in \delta(\psi)$ holds $I'(C) = 1$.

To prove this statement, we assume that ψ is satisfiable. Then we have to show that for some interpretation I' of $\delta(\psi)$ it holds that all clauses $C \in \delta(\psi)$ evaluate to true, i.e., $\forall C \in \delta(\psi) : I'(C) = 1$.

As we assumed ψ to be satisfiable, there exists a model I of ψ. We extend I to an interpretation I' for $\delta(\psi)$ as follows:

C1 $I'(a) = I(a)$ for every propositional variable a occurring in ψ.

C2 $I'(\ell_\phi) = I(\phi)$ for every subformula occurrence ϕ of ψ, i.e., $\phi \in \Sigma(\psi)$, where ℓ_ϕ is the label assigned to ϕ.

It remains to show that I' is also a model of $\delta(\psi)$.

For the following proof, we assume without further notice that ϕ is a subformula occurrence of ψ, i.e., $\phi \in \Sigma(\psi)$.

As every clause in $\delta(\psi) \setminus \{(\ell_\psi)\}$ results from the translation of one subformula occurrence ϕ of ψ, we first show by structural induction on ψ that, for all $C \in \delta(\psi) \setminus \{(\ell_\psi)\}$, it that holds $I'(C) = 1$. The Induction Hypothesis (IH) which we use is as follows:

IH: If ϕ' is a proper subformula of ϕ (i.e, $\phi' \neq \phi$) then I' satisfies all clauses in $\hat{\delta}(\psi) = \delta(\psi) \setminus \{(\ell_\psi)\}$ that stem from the translation of ϕ'.

- Base case: $\phi = a$ where a is propositional variable. The clauses in $\delta(\psi)$ stemming from the translation of ϕ are $(\neg\ell_a \vee a)$ and $(\ell_a \vee \neg a)$. To show that they evaluate to true under I' consider all cases for $I(a)$:
  - $I(a) = 1$: Then $I'(a) = 1$ by **C1** and $I'(\ell_a) = 1$ by **C2**, thus $I'(\neg\ell_a \vee a) = 1$ and $I'(\ell_a \vee \neg a) = 1$.
  - $I(a) = 0$: Then $I'(a) = 0$ by **C1** and $I'(\ell_a) = 0$ by **C2**, thus $I'(\neg\ell_a \vee a) = 1$ and $I'(\ell_a \vee \neg a) = 1$.

  Therefore all clauses for $\phi = a$ are satisfied by I'.

- Induction step: Case $\phi = \phi_1 \wedge \phi_2$. The clauses are $(\neg\ell_\phi \vee \ell_1)$, $(\neg\ell_\phi \vee \ell_2)$, $(\ell_\phi \vee \neg\ell_1 \vee \neg\ell_2)$ where the label for ϕ_1 is ℓ_1, respectively for ϕ_2 is ℓ_2.
  We consider all cases for $I(\phi)$:
  - $I(\phi) = 1$: Thus $I(\phi_1) = I(\phi_2) = 1$ by the semantics of $\wedge$, so $I'(\ell_1) = I'(\ell_2) = 1$ by **C2** as well as $I'(\ell_\phi) = 1$. Therefore $I'(\neg\ell_\phi \vee \ell_1) = I'(\neg\ell_\phi \vee \ell_2) = I'(\ell_\phi \vee \neg\ell_1 \vee \neg\ell_2) = 1$.
  - $I(\phi) = 0$: Thus $I(\phi_1) = 0$ or $I(\phi_2) = 0$. Without loss of generality we assume $I(\phi_1) = 0$. Then $I'(\ell_\phi) = I'(\ell_1) = 0$ by **C2**. Therefore $I'(\neg\ell_\phi \vee \ell_1) = I'(\neg\ell_\phi \vee \ell_2) = I'(\ell_\phi \vee \neg\ell_1 \vee \neg\ell_2) = 1$.

  As all clauses for ϕ_1 and ϕ_2 are satisfied by I' according IH, it follows that all clauses for $\phi = \phi_1 \wedge \phi_2$ are satisfied by I'.

- Induction step: Case $\phi = \neg\phi_1$. The clauses are $(\neg\ell_\phi \vee \neg\ell_1), (\ell_\phi \vee \ell_1)$ where ℓ_1 is the label for ϕ_1.
  We consider all cases for $I(\phi)$:
  - $I(\phi) = 1$: Thus $I(\phi_1) = 0$ and $I'(\ell_\phi) = 1$ and $I'(\ell_1) = 0$ by **C2**. Therefore $I'(\neg\ell_\phi \vee \neg\ell_1) = I'(\ell_\phi \vee \ell_1) = 1$.
  - $I(\phi) = 0$: Thus $I(\phi_1) = 1$ and $I'(\ell_\phi) = 0$ and $I'(\ell_1) = 1$ by **C2**. Therefore $I'(\neg\ell_\phi \vee \neg\ell_1) = I'(\ell_\phi \vee \ell_1) = 1$.

  As all clauses for ϕ_1 are satisfied by I' according to IH, all clauses for $\phi = \neg\phi_1$ are satisfied by I'.

The only remaining clause not covered by structural induction is (ℓ_ψ) where ℓ_ψ is the label assigned to ψ. As $I \in Mod(\psi)$ holds $I(\psi) = 1$ and thus by **C2** $I'(\ell_\psi) = 1$ holds.

Therefore all clauses are satisfied by I' and we have proven: if ψ is satisfiable then $\delta(\psi)$ is satisfiable. □

Shorter Alternative: One can show that the clauses for the cases $\phi = a$ and $\phi = \neg\phi_1$ evaluate to true in shorter terms. Instead of the case distinction for $I(\phi)$, directly use the relationship between ϕ and its assigned label, as shown in the following:

- Case $\phi = a$: $I'(a) = I'(\ell_a)$ holds by **C2** and therefore $I'(\neg\ell_a \vee a) = 1 - I'(\ell_a) + I'(\ell_a) = 1$ and $I'(\ell_a \vee \neg a) = I'(\ell_a) + 1 - I'(\ell_a) = 1$, so all clauses are satisfied.
- Case $\phi = \neg\phi_1$: By **C2** and the semantics of negation it holds that $I'(\ell_\phi) = 1 - I'(\ell_1)$ therefore $I'(\neg\ell_\phi \vee \neg\ell_1) = 1 - (1 - I'(\ell_1)) + 1 - I'(\ell_1) = 1$ and $I'(\ell_\phi \vee \ell_1) = 1 - I'(\ell_1) + I'(\ell_1) = 1$. As the clauses for ϕ_1 are satisfied by IH, all clauses for ϕ are satisfied.

Notice: The rest of the proof (assumption I that is a model, induction hypothesis, etc.) remains the same.

To answer the last question, reconsider the equivalent replacement theorem. We can translate any formula λ to its restricted form λ' such that $\lambda \equiv \lambda'$. Thereby, we use the equivalences $(\varphi \vee \psi) \equiv \neg(\neg\varphi \wedge \neg\psi)$, $(\varphi \rightarrow \psi) \equiv \neg(\varphi \wedge \neg\psi)$, and $(\varphi \leftrightarrow \psi) \equiv (\neg(\varphi \wedge \neg\psi) \wedge \neg(\psi \wedge \neg\varphi))$. Consequently, (1) holds also for unrestricted formulas.

Exercise 14. *Implication Graphs*

14.1 Prove: *During the run of a SAT solver, the implication graph G_k at step k is acyclic.*

Hints:

1. Perform a proof by induction on k.
2. Consider the following events that can occur:
   a) making a decision,
   b) unit propagation (one step of BCP),
   c) a clause is unsatisfiable,
   d) backtracking.

14.2 Show that in a conflict graph the first UIP is uniquely defined, i.e., there is exactly one node in the implication graph which is the first UIP.

14.3 Let $\mathcal{C}$ be a set of clauses and G a conflict graph with respect to $\mathcal{C}$. Prove: If C_l is the first clause that is learned following the first-UIP scheme, then C_l is a consequence of $\mathcal{C}$.

Bonus questions: How can this statement be used to show that all clauses that are learned (following the first-UIP scheme) are a consequence of $\mathcal{C}$?

Solution 14.

14.1 Let $P(n)$ be the property that G_n is acyclic.

Base case $n = 0$: G_0 is the empty graph, hence it is acyclic. Therefore $P(0)$ holds.

Induction Hypothesis (IH): Let n be an integer ≥ 0 and suppose $P(0), \ldots, P(n)$ are all true.

Induction step: Consider $P(n+1)$ which we have to show to be true. By (IH) it holds that $G_i = (V_i, E_i)$ is acyclic $(0 \leq i \leq n)$. We continue by a case distinction wrt. all possible steps of the SAT solver.

1. "Making a decision": Without loss of generality, let the decision be $X = f@d$. Then $V_{n+1} = V_n \cup \{X\}$ and $E_{n+1} = E_n$. Since G_n is acyclic, $G_{n+1} = (V_{n+1}, E_{n+1})$ also is acyclic because no edge is added to the graph.
2. "Unit propagation": Without loss of generality, let the unit clause comes from $(\ell_1 \vee \cdots \vee \ell_m)$ and let ℓ_m be the unassigned variable. Then $V_{n+1} = V_n \cup \{\ell_i\}$ and $E_{n+1} = E_n \cup \{(\ell_j, \ell_m) \mid 1 \leq j \leq m-1\}$. Since all added edges go to the newly added node ℓ_m, it follows by (IH) that G_{n+1} is acyclic.
3. "A clause is unsatisfiable": Without loss of generality, let the unsatisfiable clause be $(\ell_1 \vee \cdots \vee \ell_m)$, then $V_{n+1} = V_n \cup \{\kappa\}$ and $E_{n+1} = E_n \cup \{(\ell_j, \kappa) \mid 1 \leq j \leq m\}$. Note that no new edge has been added other than those going to κ. Hence G_{n+1} only contains a cycle, if G_n contains a cycle. Therefore it follows by (IH) that G_{n+1} is acyclic.
4. "Backtracking": Since backtracking only removes a part of the implication graph $G_n = (V_n, E_n)$, we have that $G_{n+1} = (V_n \setminus V', E_n \setminus E')$ for some sets $V' \subset V_n, E' \subset E_n$. Removing edges cannot make a graph cyclic. Therefore, by (IH) follows that G_{n+1} is acyclic.

Since $P(n+1)$ holds for any step under the assumption that (IH) holds, it therefore follows that $P(n)$ holds for any $n \geq 0$. Consequently, the implication graph G_k at step k is acyclic.

14.2 Proof by contradiction: Assume there are two nodes v, v' where both v and v' are first-UIPs. Let d be the node of the last decision and κ the conflict node.

A UIP is by definition a node where all paths from d to κ go through. As v and v' are first-UIPs, they both are UIPs, so all paths from d to κ go through v and also through v'.

Therefore there either is a path $d, \ldots, v, \ldots, v', \ldots, \kappa$ from v to v' or there is a path from v' to v. Without loss of generality, let the path be from v to v'. As all paths from d to κ go through v and v', all paths are of the form $d, \ldots, v, \ldots, v', \ldots, \kappa$, because the implication graph is acyclic.

As $v \neq v'$ the distance $d(v', \kappa)$ between v' and κ is smaller than the distance $d(v, \kappa)$, i.e., $d(v', \kappa) < d(v, \kappa)$. But this contradicts the assumption that v is a first-UIP, because v' is closer to the conflict κ than v.

Therefore there can be only one first UIP.

As d, the current decision node, is always a UIP, there always exists a at least one UIP, hence there also exists a UIP closest to the conflict, i.e., there exists a first UIP.

14.3 Consider how a new clause is learned: Find the first-UIP u and resolve with clauses from the conflict k to u. Let $S \subseteq \mathcal{C}$ denote those clauses that occur as edge-labels in the implication graph G from the first UIP u to the conflict node k.

As C_l is learned following the first UIP schema, there is a resolution derivation $K_1, K_2, \ldots, K_n$ of C_l from S where $K_n = C_l$ and for each K_ℓ holds: either $K_\ell \in S$ or K_ℓ is the resolvent of two K_i and K_j with $i, j < \ell$ and $1 \leq \ell \leq n$. As resolution is correct it follows that $S \models C_l$.

By monotonicity of propositional logic it then follows that $F \cup S \models C_l$ for any set of formulas F, specifically $\mathcal{C} \cup S \models C_l$. And as $S \subseteq \mathcal{C}$ it follows that $\mathcal{C} \models C_l$.

Bonus question: For a sequence $C_1, \ldots, C_n$ of learned clauses, we can inductively apply the statement as follows. C_1 is a consequence of $\mathcal{C}$ by the above proof. C_2 is a consequence of $\mathcal{C} \cup \{C_1\}$ again by the above proof; since C_1 is a consequence of $\mathcal{C}$ it therefore holds by transitivity of the consequence relation that C_2 also is a consequence of $\mathcal{C}$. The proof sketch for C_3 to C_n then is analogous. Note that this is not a full inductive proof, but just a quick and incomplete proof sketch.

Exercise 15. *Ackermann's Reduction (recommended)*
Transform the EUF-formula φ^{EUF} below to an E-formula φ^{E} using Ackermann's reduction. Note that φ^{EUF} contains an uninterpreted predicate, which requires special treatment first.

$$\varphi^{EUF}: \quad F(F(x_1)) \doteq G(x_2, G(x_1, x_3, x_4), F(x_2)) \rightarrow p(x_1, y).$$

Solution 15.
First, we replace the predicate (as described in the lecture) and obtain:

$$\varphi' : \quad F(F(x_1)) \doteq G(x_2, G(x_1, x_3, x_4), F(x_2)) \rightarrow H_p(x_1, y) \doteq x_p.$$

Label-ling function occurrences inside-out yields:

$$\varphi'' : \quad F_2(F_1(x_1)) \doteq G_2(x_2, G_1(x_1, x_3, x_4), F_3(x_2)) \rightarrow H_p(x_1, y) \doteq x_p.$$

The propositional skeleton is:

$$flat^E : \quad f_2 \doteq g_2 \rightarrow h_p \doteq x_p.$$

FC^E is the conjunction of the following functional constraints:

$$\begin{aligned} x_1 \doteq f_1 &\rightarrow f_1 \doteq f_2 \\ x_1 \doteq x_2 &\rightarrow f_1 \doteq f_3 \\ f_1 \doteq x_2 &\rightarrow f_2 \doteq f_3 \\ (x_2 \doteq x_1 \wedge g_1 \doteq x_3 \wedge f_3 \doteq x_4) &\rightarrow g_1 \doteq g_2 \end{aligned}$$

Note that there are no constraints for h_p since it only occurs once in φ'. Finally, φ^E is $FC^E \rightarrow flat^E$.

Exercise 16. *Integer Overflow*
We have argued in the slides of the third lecture ("First-order Logic and Theories") that most implementations of the binary search algorithms are broken. Implement the algorithm in a correct way which avoids that the program crashes for large arrays.

Solution 16.
The problem is line 6 where $(l + h)$ already can lead to an integer overflow:

$$m = (l + h)/2;$$

One possible solution: replace line 6 by

$$m = l + ((h - l)/2);$$

Exercise 17. *Integer Overflow and Program Termination*
Consider the C program shown below and recall that type `int32_t` is 32 bits in size.

```
#include <stdint.h>

int main(void) {
  int32_t x = 1;
  int32_t y = (1 << 16);
  while (x >= 0 && y >= 0) {
      y = y * y;
      x = x - y;
    }
}
```

- Does the program terminate?
- Does termination depend on the integer representation?

Solution 17.
If you don't see it from the C source, simply try it! For instance, use `gcc` and compile the C file, say `p32.c`, using `gcc -o p32 p32.c`. Change the type `int32_t` to `int64_t` and try again.

Exercise 18. *Ackermann-Péter function*
Augment Preßburger arithmetic with the following axioms to define the Ackermann-Péter function.

$$\forall y.\ ap(0, y) = y + 1 \qquad \text{(ap left zero)}$$
$$\forall x.\ ap(x + 1, 0) = ap(x, 1) \qquad \text{(ap right zero)}$$
$$\forall x, y.\ ap(x + 1, y + 1) = ap(x, ap(x + 1, y)) \qquad \text{(ap successor)}$$

Calculate $ap(x, x)$ for $x = 0, 1, 2, 3, 4$. Moreover, prove that the computation of $ap(x, y)$ terminates for every x, y from the non-negative integers and that $\forall x, y.\ ap(x, y) > y$ holds.

Hint: Use well-founded induction.

Solution 18.
Some parts have been taken from the book of Bradley and Manna. Some values for $ap(m, n)$can be found at `https://en.wikipedia.org/wiki/Ackermann_function`.

We give the values here: $ap(0, 0) = 1$, $ap(1, 1) = 3$, $ap(2, 2) = 7$, $ap(3, 3) = 61$, and $ap(4, 4) = 2^{2^{2^{2^{16}}}} - 3$.

We know that $(\mathbb{N}_0 \times \mathbb{N}_0, \sqsubseteq)$ is a well-founded and lexicographically ordered set with least element $(0, 0)$. Then the following holds for all $x, y \in \mathbb{N}_0$.

1. $(x + 1, 0) \sqsubset (x + 1, 1)$
2. $(x + 1, y) \sqsubset (x + 1, y + 1)$
3. $(x, ap(x + 1, y)) \sqsubset (x + 1, y + 1)$

Let $\mathcal{P}(x, y)$ denote $ap(x, y) > y$. The proof is by lexicographic well-founded induction on $(\mathbb{N}_0 \times \mathbb{N}_0, \sqsubseteq)$.

Base case: The least element is $(0, 0)$. Then $\mathcal{P}(0, 0)$ is true because (ap left zero).

Induction hypothesis. Pick arbitrarily a non-minimal element (x, y) and assume that $\mathcal{P}(x', y')$ is true for all $(x', y') \sqsubset (x, y)$.

Induction step. We want to show that $\mathcal{P}(x, y)$ is true. We distinguish the following three cases.

Case 1: $x = 0$. Then $ap(0, y) = y + 1$ by (ap left zero) and $\mathcal{P}(0, y)$ is true.

Case 2: $x \neq 0 \wedge y = 0$. Then $(x - 1, 1) \sqsubset (x, 0)$. By the induction hypothesis, $\mathcal{P}(x - 1, 1)$ is true and $ap(x, 0) = ap(x - 1, 1)$ by (ap right zero). Therefore, $\mathcal{P}(x, 0)$ is true.

Case 3: $x \neq 0 \wedge y \neq 0$. Then $(x, y-1) \sqsubset (x, y)$ and, forall $z \in \mathbb{N}_0$, $(x-1, z) \sqsubset (x, y)$. Then, $\mathcal{P}(x, y-1)$ is true by the induction hypothesis. Moreover, for all $z \in \mathbb{N}_0$, $\mathcal{P}(x-1, z)$ is also true by the induction hypothesis. Then $ap(x, y) = ap(x-1, ap(x, y-1))$ by (ap successor) The induction hypothesis then implies $ap(x, y - 1)) \geq y - 1$ and $ap(x - 1, z) \geq z$ for all $z \in \mathbb{N}_0$. Hence, $\mathcal{P}(x, y)$ is true.

We conclude that $\mathcal{P}(x, y)$ is true for all $(x, y) \in \mathbb{N}_0 \times \mathbb{N}_0$. □