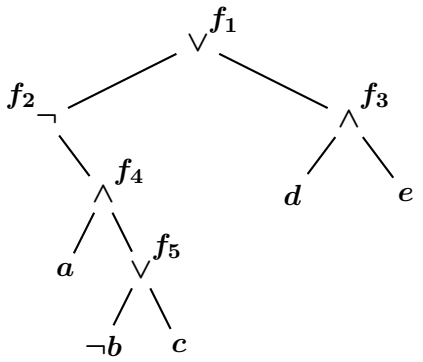
VU Programm- und Systemverifikation
Homework 3a: Propositional Logic / SAT

(10 points)

May 29, 2019

1. Convert the following formula into conjunctive normal form (CNF) by applying **Tseitin transformation only**.

$$\neg(a \wedge (\neg b \vee c)) \vee (d \wedge e)$$



$f_1 \wedge (f_1 \leftrightarrow (f_2 \vee f_3)) \wedge (f_2 \leftrightarrow (\neg f_4)) \wedge (f_4 \leftrightarrow (a \wedge f_5)) \wedge (f_5 \leftrightarrow (\neg b \vee c)) \wedge (f_3 \leftrightarrow (d \wedge e))$

where:

$f_1 \leftrightarrow (f_2 \vee f_3) \equiv (f_1 \vee \neg f_2) \wedge (f_1 \vee \neg f_3) \wedge (f_2 \vee f_3 \vee \neg f_1)$

$f_2 \leftrightarrow (\neg f_4) \equiv (\neg f_2 \vee \neg f_4) \wedge (f_2 \vee f_4)$

$f_3 \leftrightarrow (d \wedge e) \equiv (\neg f_3 \vee d) \wedge (\neg f_3 \vee e) \wedge (\neg d \vee \neg e \vee f_3)$

$f_4 \leftrightarrow (a \wedge f_5) \equiv (\neg f_4 \vee a) \wedge (\neg f_4 \vee f_5) \wedge (\neg a \vee \neg f_5 \vee f_4)$

$f_5 \leftrightarrow (\neg b \vee e) \equiv (f_5 \vee \neg\neg b) \wedge (f_5 \vee \neg e) \wedge (\neg b \vee e \vee \neg f_5)$

Upload a pdf file with your solution to TUWEL by June 2, 2019. [5 points]

2. Consider the package dependency graph shown in Figure 1. Nodes depict software packages; a solid arrow from package P_i to package P_j requires P_j to be installed, if P_i is installed; a dashed arrow from package P_i to package P_j prohibits P_i to be installed, if P_j is installed. We model information about installed packages using 13 boolean variables $x_1, \ldots, x_{13}$: for $i : 1 \leq i \leq 13$, if $x_i = \mathit{true}$, then package i is installed.

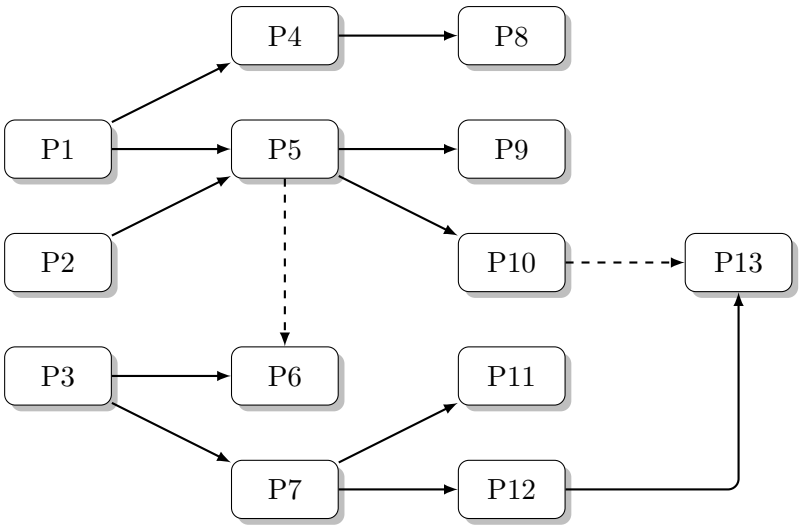


Figure 1: The package dependency graph: nodes depict software packages; a solid arrow from package P_i to package P_j requires P_j to be installed, if P_i is installed; a dashed arrow from package P_i to package P_j prohibits P_i to be installed, if P_j is installed.

Encode the constraints of the graph in Figure 1 as a CNF. Create the file `graph.cnf` and write down the CNF there in the DIMACS format.[1] Use exactly 13 variables: for a package P_i, use the boolean variable with the number i, for $1 \leq i \leq 13$.

You can check the correctness of your encoding with the following tests:

(a) For each solid arrow $i \to j$ in the graph, if the user adds the clauses `"i 0"` and `"j 0"` to `graph.cnf` (with the number of clauses fixed accordingly) and runs the solver on `graph.cnf`, the solver returns "SAT".

(b) For each pair (i, j) that does not have a solid arrow $i \to j$ in the graph, if the user adds the clauses `"i 0"` and `"-j 0"` to `graph.cnf` (with the number of clauses fixed accordingly) and runs the solver on `graph.cnf`, the solver returns "SAT".

(c) If the user adds the clauses `"1 0"` and `"2 0"` to `graph.cnf` (with the number of clauses fixed accordingly) and runs the solver on `graph.cnf`, the solver returns "SAT".

(d) If the user adds the clauses `"1 0"` and `"3 0"` to `graph.cnf` (with the number of clauses fixed accordingly) and runs the solver on `graph.cnf`, the solver returns "UNSAT".

Upload the file `graph.cnf` to TUWEL by June 2, 2019. Make sure that the file contains your name and ID (as a comment). Note that `graph.cnf` **must** contain only the CNF describing Figure 1, not including the changes specified in points (2a)–(2d). [5 points]

See solution.cnf in TISS.

[1] The DIMACS format: `http://www.satcompetition.org/2009/format-benchmarks2009.html`