1	2	3	4	Σ	Grade

6.0/4.0 VU Formale Methoden der Informatik
185.291 **May 20, 2022**

Kennz. (study id)	Matrikelnummer (student id)	Nachname (surname)	Vorname (first name)

1.) An undirected graph (V, E) is called a *degree-restricted graph* if for each vertex $v \in V$ it holds that the degree of v is either 1 or greater than 3.

Examples:

- $(\{a, b, c, d, e\}, \{[a, b], [a, c], [a, d], [a, e]\})$ is degree-restricted since vertex a has degree 4 and vertices $\{b, c, d, e\}$ have all degree 1.
- $(\{a, b, c, d\}, \{[a, b], [b, c], [c, d]\})$ is not degree-restricted since vertices b, c have degree 2.
- The complete graph K_4, i.e. a clique with 4 vertices, is also not degree-restricted since all vertices have degree 3.

Consider the following variant of the 3-coloring problem:

> **3-COLORABILITY-DEGREE-RESTRICTED (3COLD)**
>
> INSTANCE: A degree-restricted graph $G = (V, E)$.
>
> QUESTION: Does there exist a function μ from vertices in V to values in $\{0, 1, 2\}$ such that $\mu(v_1) \neq \mu(v_2)$ for any edge $[v_1, v_2] \in E$.

(a) The following function f provides a polynomial-time many-one reduction from **3COL** to **3COLD**: for an undirected graph $G = (\{v_1, \ldots, v_n\}, E)$, add for each vertex v_i

- four new vertices x_i^1, x_i^2, x_i^3, x_i^4, and
- edges $[v_i, x_i^1]$, $[v_i, x_i^2]$, $[v_i, x_i^3]$, $[v_i, x_i^4]$

to G, and let $f(G)$ be the resulting graph. Note that $f(G)$ is indeed degree-restricted, since the new vertices x_i^j all have degree 1 and vertices v_i have at least degree 4.
Show that G is a yes-instance of **3COL** $\Longleftrightarrow$ $f(G)$ is a yes-instance of **3COLD**.

(9 points)

(b) Let us assume the reduction from **3COL** to **3COLD** is correct. Argue briefly why we can then conclude that **3COLD** is NP-complete.

(6 points)

2.) We consider the function P which was introduced by Rózsa Péter in 1935.

Input: x, y, two *non-negative* integers
Output: The computed integer value for x, y
if $x == 0$ **then**
  | **return** $2y + 1$;
end
else if $y == 0$ **then**
  | **return** $\mathrm{P}(x - 1, 1)$;
end
else return $\mathrm{P}(x - 1, \mathrm{P}(x, y - 1))$;

Algorithm 1: The function P

(a) Let $\mathbb{N}_0$ denote the set of natural numbers *including* 0. Use well-founded induction to show

$$\forall x \, \forall y \, \big((x \in \mathbb{N}_0 \, \wedge \, y \in \mathbb{N}_0) \, \rightarrow \, \mathrm{P}(x, y) > 2x + 2y\big).$$

(12 points)

(b) Consider the clauses $C_1, \ldots, C_6$ in `dimacs` format (in this order from top to bottom, shown in the box) which are given as input to a SAT solver.

- Apply CDCL using the convention that if a variable is assigned as a decision, then it is assigned 'true'. Select variables as decisions in increasing order of their respective integer IDs in the `dimacs` format, starting with variable `1`.
- When the *first* conflict occurs, draw the complete implication graph, mark the first UIP, give the derivation of the learned asserting clause that corresponds to the first UIP, and stop CDCL. You do not have to solve the formula!

```
-1 4 0
-4 5 0
-2 -4 6 0
-3 -6 7 0
-7 9 0
-5 -6 -7 -9 0
```

(3 points)

3.) (a) Let p be the following IMP program loop, containing the integer-valued program variables i, x, y:

$$
\begin{array}{l}
x := 10; y := 10; \\
\textbf{while } i \geq 0 \textbf{ do} \\
\quad x := x - 4 * i; \\
\quad y := y + 4 * i; \\
\quad i := i - 1; \\
\textbf{od}
\end{array}
$$

Give a loop invariant and variant for the **while** loop in p and prove the validity of the total correctness triple $[i > 0]\ p\ [x + y = 20]$.

(10 points)

(b) Let p be the following IMP program loop, containing the integer-valued program variables i and x:

$$
\begin{array}{l}
\textbf{while } i \neq 3 \textbf{ do} \\
\quad x := 3 * x; \\
\quad i := i - 1; \\
\textbf{od}
\end{array}
$$

Provide a non-trivial pre-condition A, such that:

(i) the total correctness triple $[A]\ p\ [x \geq 27]$ is valid;

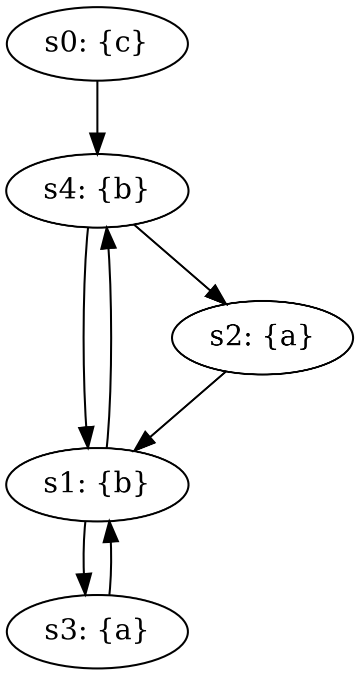
(ii) the partial correctness triple $\{A\}\ p\ \{x \geq 27\}$ is valid, but the total correctness triple $[A]\ p\ [x \geq 27]$ is not valid.

Trivial means equivalent to `true` or `false`, so your precondition A should not be equivalent to `true` or `false`. Justify your answer!

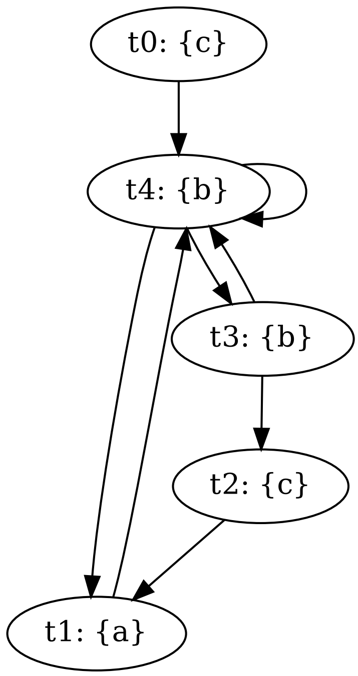
(5 points)

4.) (a) Consider the Kripke structures M_1 and M_2. The initial state of M_1 is s_0, the initial state of M_2 is t_0.

Kripke structure M_1**:**



Kripke structure M_2**:**

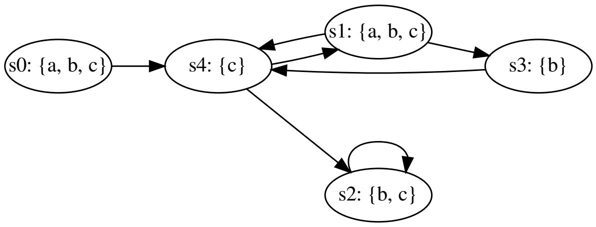


i. Check whether M_2 simulates M_1, i.e., provide a simulation relation that witnesses $M_1 \preceq M_2$, or explain why M_2 does not simulate M_1.

ii. Check whether M_1 simulates M_2, i.e., provide a simulation relation that witnesses $M_2 \preceq M_1$, or explain why M_1 does not simulate M_2.

(4 points)

(b) Consider the following Kripke structure M:



For each of the following formulae φ,

i. indicate whether the formula is in CTL, LTL, and/or CTL*, and
ii. list the states s_i on which the formula φ holds; i.e. for which states s_i do we have $M, s_i \models \varphi$?
(If φ is a path formula, list the states s_i such that $M, s_i \models \mathbf{A}\varphi$.)

φ	CTL	LTL	CTL*	States s_i
$\mathbf{G}(b \wedge c)$	☐	☐	☐	
$\mathbf{EFG}a$	☐	☐	☐	
$\mathbf{E}(\mathbf{G}c \wedge \mathbf{GF}b)$	☐	☐	☐	
$\mathbf{F}(a \wedge \mathbf{X}c)$	☐	☐	☐	
$\mathbf{E}[(\neg a)\ \mathbf{U}\ c]$	☐	☐	☐	

(5 points)

(c) **LTL tautologies**

An LTL formula is a *tautology* if it holds for every Kripke structure M and every path π in M. For each of the following formulas, prove that it is a tautology, or find a Kripke structure M and path π in M for which the formula does not hold and justify your answer.

i. $\neg(a\ \mathbf{U}\ b) \Rightarrow (\mathbf{G}\neg b \vee \mathbf{F}\neg a)$
ii. $(\mathbf{G}\neg b \vee \mathbf{F}\neg a) \Rightarrow \neg(a\ \mathbf{U}\ b)$

(6 points)